EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Motion Grammars for Character Animation

Kyunglyul Hyun, Kyungho Lee, and Jehee Lee

Seoul National University, Republic of Korea

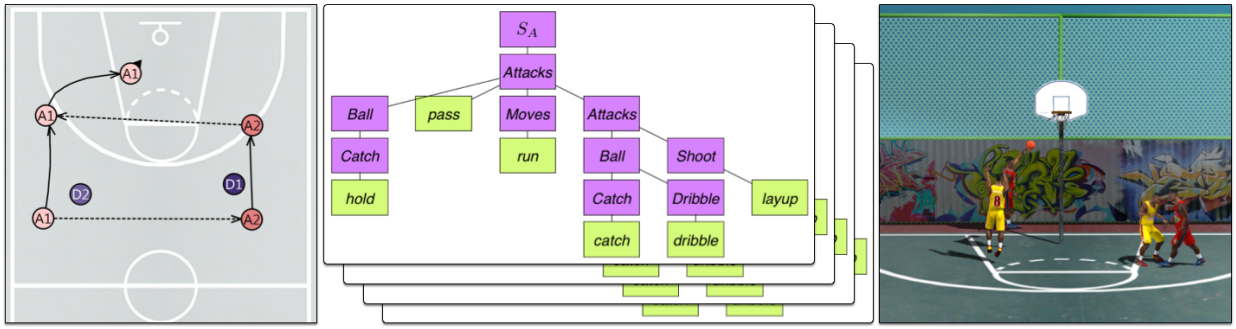

Figure 1: *Our motion grammar reconstructs a structurally-valid 3D animated scene from a sketch of the basketball tactics board.*

Abstract

The behavioral structure of human movements is imposed by multiple sources, such as rules, regulations, choreography, habits, and emotion. Our goal is to identify the behavioral structure in a specific application domain and create a novel sequence of movements that abide by structure-building rules. To do so, we exploit the ideas from formal language, such as rewriting rules and grammar parsing, and adapted those ideas to synthesize the three-dimensional animation of multiple characters. The structured motion synthesis using motion grammars is formulated in two layers. The upper layer is a symbolic description that relates the semantics of each individual's movements and the interaction among them. The lower layer provides spatial and temporal contexts to the animation. Our multi-level MCMC (Markov Chain Monte Carlo) algorithm deals with the syntax, semantics, and spatiotemporal context of human motion to produce highly-structured, animated scenes. The power and effectiveness of motion grammars are demonstrated in animating basketball games from drawings on a tactic board. Our system allows the user to position players and draw out tactical plans, which are animated automatically in virtual environments with three-dimensional, full-body characters.

1. Introduction

Natural human movements are often strongly structured. Behavioral structures may be imposed explicitly by rules, regulations, and choreography. Structures may arise implicitly from habits, cultural heritage, and emotion. For example, the rules of basketball dictate how many steps the player can take while holding the ball and prevent players from violations such as double dribbling. Basketball rules and regulations render highly-constrained structures of players' movements. Dance choreography is another example of structured human movements. Each category of dances has established the conventions of its form, motion, and rhythm. Dance choreographers dictate motion and form in a highly structured manner based on such conventions.

Motion capture technology is popular in computer graphics and thus large databases of high-quality human motion data are readily available on the web. A common goal of data-driven animation research is to animate computer-generated characters by using a collection of canned motion data. Efforts for achieving this goal have developed a toolbox of data-driven techniques that range from low-level data manipulation to higher-level planning of a sequence of actions in virtual environments. Understanding the syntax and semantics of human movements would allow even higher-level con-

trol over the motion of animated characters and make it look plausible.

Our goal is to identify the behavioral structure of human movements and create a novel sequence of movements that abide by structure-building rules. To do so, we exploit formal language technologies, such as rewriting rules and grammar parsing, developed in computer science and computational linguistics. The underlying assumption is that behavioral structures can largely be formulated as context-free grammars. In graphics applications, human motion data have often been stored and maintained in motion graphs [LCR*02, KGP02], which are equivalent to finite state machines and regular grammars in terms of their expressive power. The motion graph maintains a collection of motion fragments and encodes the transitioning possibilities among motion fragments. Transitioning through the graph generates a sequence of character's actions. Context-free grammars are more expressive than regular grammars, so the use of context-free grammars allow us to uncover richer hierarchical structures, regularities, interactions, and patterns that could not be expressed in motion graphs. Motion grammars can be easily incorporated into existing animation systems that are equipped with motion graphs or finite state machines of character's actions.

The theory of formal language does not generalize easily to deal with human movements. Human motion data are high-dimensional, continuous, and time-series, whereas a formal language is a string of discrete symbols. Human movements take place at specific spatial locations and specific time instances. The spatial and temporal contexts do not exist in formal languages. Our *motion grammar* is a set of context-free rewriting rules of quantized *motion symbols* and each rule is annotated with semantic context that may evaluate and trigger actions. We formulate a multi-character motion synthesis problem in two layers. The structure layer is a symbolic, tree-based description that relates the structure of each individual's movements and the interaction among them. The semantic layer provides spatial and temporal contexts to the animated scene and generates full-body animation. Our multi-level MCMC (Markov Chain Monte Carlo) algorithm deals with the syntax, semantics, and spatiotemporal context of human motion to produce plausible, highly-structured, animated scenes.

To demonstrate the power of motion grammars, we built an animation system that produces the animation of basketball plays from drawings on a tactics board. Most basketball coaches use clipboards to position players and draw out tactical plans. Our system reconstructs the full-body motion of multiple players dribbling, passing, and shooting balls. Motion grammars are used to describe basketball regulations and the behavioral patterns of offensive and defensive players.

2. Related Work

Motion capture is a major source of realism in modern character animation. Data-driven approaches commonly segment motion capture data into short fragments and splice them in a novel sequence. Animation techniques based on motion capture can roughly be classified into three categories. A large class of data-driven methodologies edit individual fragments subject to new requirements. There exist another class of methodologies that focus on temporal sequencing and spatial alignment of motion data in space and time. The first category of techniques tend to make smooth, continuous changes over individual motion fragments or a family of parameterized motions, while the problems in the second category are discrete and combinatorial. The last category of techniques tried to solve both continuous editing and discrete planning simultaneously.

The continuous editing of motion is usually formulated as constrained optimization, which minimizes the deviation from the original motion data subject to user-specified constraints and requirements. This formulation has been effective for a wide range of problems, such as retargeting motion to new characters [Gle98], interactive manipulation [LS99], blending a family of similar motions [RCB98], statistical modeling [MCC09, MC12], and incorporating physics-based objectives and constraints [LHP05]. The idea has further been explored to deal with multiple interacting characters in the context of interactive manipulation [KLLT08, KHKL09, KSKL14].

The combinatorial planning of action sequences often requires an efficient data structure to store and search motion data. The most popular structure is a motion graph [LCR*02, KGP02], which is essentially a finite state machine encapsulating the connectivity among motion fragments. The concept of motion graphs has further been elaborated to cope with families of parameterized motions [SO06]. Good segmentation and clustering of motion fragments are key ingredients of building effective motion data structures [BSP*04, BCvdPP08]. Provided that such a structure is built, synthesizing novel motion sequences entails combinatorial searching through the connectivity among motion fragments. Temporal sequencing of motion fragments has been addressed by using state-space search [LCR*02, KGP02, SH07], dynamic programming [AFO03], min-max search [SKY12], and policy learning [MP07, TLP07]. State-space search methods are closely related to the path planning of three-dimensional characters in highly-constrained and dynamically-changing environments [CKHL11, LLKP11].

Naturally, there have been efforts to integrate continuous optimization and combinatorial planning into a single framework. Motion data and their connectivity can be projected into a single continuous configuration space via dimensionality reduction [SL06] and can be retrieved from the configuration space by using multivariate regression [LWB*10]. Combinatorial planning problems can be reformulated as continuous optimization in the projected space. Alternatively, careful design of user interfaces can allow two heterogeneous types (continuous and discrete) operations to occur seamlessly in interactive manipulation of motion data [KHKL09].

Animating multiple interacting characters is a very challenging problem because the computational complexity of naïve approaches scales exponentially with respect to the number of characters. Many recent studies exploited *motion patches* to address the problem. While Lee et al [LCL06] originally invented motion patches to animate characters in complex virtual environments, subsequent studies adopted motion patches to deal with interactions among characters and coordinate their actions in the spatiotemporal domain [SKSY08, HKHL13]. Won et al [WLO*14] presented a pre-visualization system that generates full-body fight scenes of

multiple characters from a high-level graphical scene description. Our work also addresses motion synthesis with single and multiple characters, but tackles a new aspect of the problem with emphasis on understanding, formulating, and animating the structured behavior of individuals and their interactions using formal grammars.

Formal languages and context-free grammars have previously been explored in the computer graphics community for shape analysis and procedural geometric modeling [LWW08, BWS10] and in robotics for the control of robotic manipulators [DS13]. Grammar induction from human motion data has been used to understand the static structure of human movements [PLL11, GFA10, GFA07]. To the best of our knowledge, we for the first time demonstrate structured, full-body motion synthesis for practical applications using motion grammars.

3. Motion Grammar

The theory of formal languages is well-established in computer science. There exists a hierarchy of formal language classes. A set of languages generated by regular grammars form the smallest class. Context-free grammars generate a larger class of languages and the class spanned by context-sensitive grammars is even larger. Regular grammars are simple, easy-to-implement and thus have frequently been used in many text processing applications. However, their expressive power is too limited to deal with the general structures of programming languages and natural languages. Context-sensitive grammars are the most powerful among them, but computing with context-sensitive grammars is extremely demanding. For example, parsing general context-sensitive grammars is NP-hard. Context-free grammars have long been considered a trade-off between two extremes and still tremendously popular in designing programming languages and processing natural languages. Context-free grammars are not powerful enough to express the whole complexity of human movements and high-level behavioral patterns. However, we believe that context-free grammars capture a large portion of human behavioral structures that is significant enough to have practical uses.

The context-free grammar has a finite number of terminal and non-terminal symbols, and includes a set of rules which rewrites the original string of symbols. Context-free means that each individual rule replaces a single non-terminal symbol in the string with another string of terminal and non-terminal symbols. One of the non-terminal symbols serves as a starting symbol and fully expanding non-terminal symbols (until no non-terminals remain in the string) generates a string of terminal symbols. Parsing is a reverse process. It begins with a string of terminal symbols and searches for an ordering of rewriting rules that generates the input string. The result of parsing is a parsing tree. The internal nodes of the tree correspond to non-terminal symbols and its leaf nodes correspond to terminal symbols. The grammar is deterministic if any valid string has a unique parsing tree. Otherwise, the grammar is non-deterministic.

3.1. Instantiation, Semantics, and Plausibility

The motion grammar is a context-free grammar on motion sequences. Terminals are symbolic representations of unit actions, such as "a half cycle of walk" and "jump shoot". Each non-terminal symbol has one or more production rules to substitute itself with series of symbols. For example, non-terminal symbol "Dribble" can produce an arbitrarily long sequence of dribbling motions that may include many dribbling skills by recursively applying the production rules. Each terminal symbol is associated with many motion clips that perform a specific unit action. The multiplicity of motion clips includes spatial, temporal, and stylistic variations of acting out the action and thus provides rich connectivity between actions.

Instantiation. The action represented by a terminal symbol is *instantiated* in the virtual environment by picking a motion clip from many available choices associated with the symbol and locating the clip in the environment at a certain time. Instantiating a string of terminal symbols, which we call an *action string*, selects a sequence of associated motion clips. Splicing them and smoothing out the seams make an extended clip, which will be situated in the virtual environment. Let X be an action string and $Y(X)$ be an extended motion clip instantiated by X. X is a symbolic representation of the structure of a character's action, while $Y(X)$ is a concrete realization of the action in the spatiotemporal context. The actual form of $Y(X)$ is a sequence of a character's full-body poses varying over time.

Semantic Rules. The animated *scene* includes one or more characters, each of which produces a course of actions X_i and its instantiation $Y_i = Y(X_i)$. Situating and orchestrating them in the common environment requires careful coordination and synchronization. We use three types of semantic rules to orchestrate multiple characters. Each individual terminal symbol can be annotated with a set of semantic rules, which are inherited to motion instances derived from the symbol. The semantic rule of the first type locates a character at desired position and/or direction $(\mathbf{p}_0, \theta_0) \in R^2 \times R$ at time t in the environment.

$$f_{\text{pos}} = \|\text{pos}(Y_i, t) - \mathbf{p}_0\|, \tag{1}$$

$$f_{\text{dir}} = \|\text{dir}(Y_i, t) - \theta_0\|, \tag{2}$$

where $\text{pos}(Y_i, t)$ and $\text{dir}(Y_i, t)$ are the position and facing direction, respectively, of the root segment (pelvis) projected on the ground surface at the t-th frame of Y_i. The rule of the second type synchronizes the action of two (or more) characters.

$$f_{\text{synch}} = \big|(t_j - t_i) - \Delta t\big|. \tag{3}$$

For example, if one character passes a ball at frame t_i so that the other character catches the ball at frame t_j, Δt is the estimated time of the ball flying between the characters. The last type coordinates and aligns the motion of a character with respect to other characters and environment objects.

$$f_{\text{dist}} = \begin{cases} 0, & \text{if } d_{\text{near}} \le \|\text{pos}(Y_i, t) - \mathbf{p}_0\| \le d_{\text{far}}, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

$$f_{\text{line}} = \text{dist}\big(\text{pos}(Y_i, t), \text{line}(\mathbf{p}_1, \mathbf{p}_2)\big). \tag{5}$$

For example, the distance rule f_{dist} encourages a defensive player to stay near, but not too close to an offensive player. The line rule f_{line} locates the defensive player on the line between the offensive player and the goal rim.

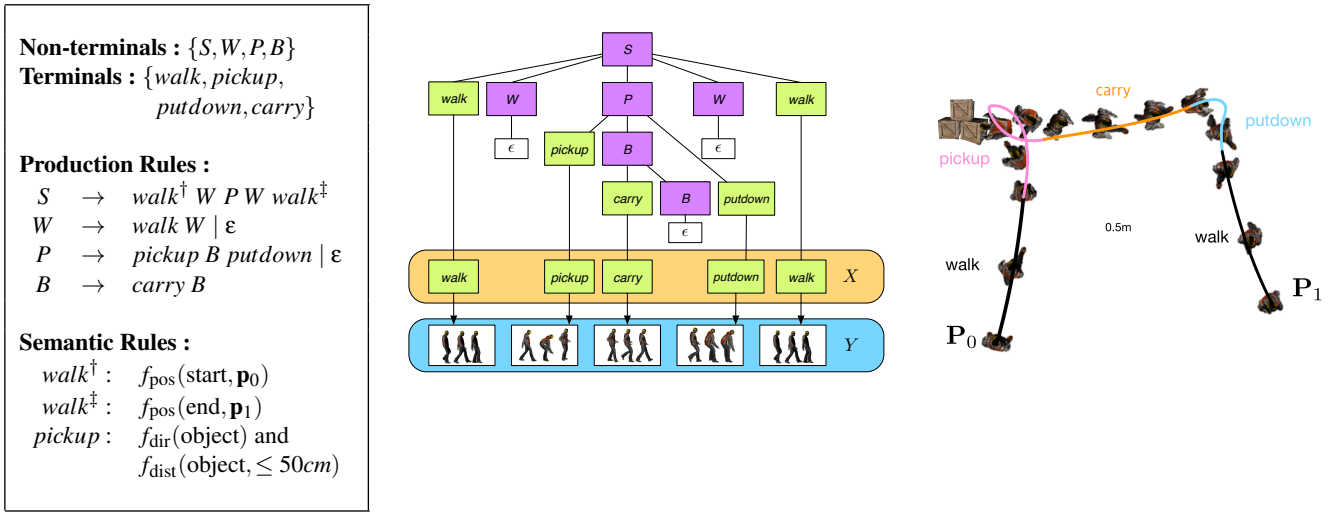

Figure 2: *A simple example. (Left) The motion grammar for carrying an object. (Middle) A parse tree to generate an action string and a series of motion clips corresponding to the string. (Right) The motion sequence spliced and situated in the virtual environment.*

Plausibility. The structural plausibility of an action string X_i is:

$$P(X_i) \propto g_{\text{parse}} \cdot \exp\big(-\frac{g_{\text{user}}}{w_1}\big), \tag{6}$$

where g_{parse} is a binary boolean function, of which value is 1 if the string X_i has a parse tree and 0 otherwise. g_{user} is a user-control term, which we will discuss in Section 4, and w_1 is its weight value. The semantic plausibility of action instances $Y = \{Y_i\}$ is:

$$P(Y) \propto \exp\big(-\sum_{k \in \mathcal{S}(Y)} \frac{f_k}{c_k} - \frac{g_{\text{smooth}}}{w_2} - \frac{g_{\text{collision}}}{w_3} - \frac{g_{\text{effort}}}{w_4}\big), \tag{7}$$

where $\mathcal{S}(Y)$ is a set of semantic rules associated with Y, c_k's are weight values, and g_{smooth} is the smoothness of motion concatenation in Y. The smoothness is measured by the weighted sum of pose and velocity mismatches at the boundaries of two consecutive motion clips [LCR*02]. $g_{\text{collision}}$ is a binary boolean function that penalizes interpenetration between characters. The function value is 1 if there is any collision, and 0 otherwise. g_{effort} is an optional term to choose a better animation among many plausible animation samples based on a secondary goal, which is either the total distance the characters travelled or their traveling time in the animation.

3.2. A Simple Example

A simple example grammar is given in Figure 2(left). The motion grammar consists of a set of non-terminal symbols $\{S, W, P, B\}$, a set of terminal symbols $\{walk, pickup, putdown, carry\}$, a starting symbol S, production rules, and semantic rules. The animation from the grammar shows a character starting to walk from $\mathbf{p}_0 \in R^2$ towards an object, picking it up, carrying it, putting it down, and walking towards the target location $\mathbf{p}_1 \in R^2$. The production rules generate a valid sequence (in other words, structure) of actions, while the semantic rules provide spatial context to the motion sequence. The semantic rules specify the start and target locations, and also declare a condition that the character can pick up an object if it faces the object and within $50cm$ from the object. Note that the grammar does not specify the position of the object to pick up. Its position may be provided by the user to control the scenario.

The production rules expand the parse tree (Figure 2(middle)) to generate an action, which describes how many steps the character will take to get to the object, how many steps it will take while carrying the object, and how many steps it will take again to get to the target location after putting down the object. A series of motion clips instantiated from the action string describe the details of individual walking steps, such as stride length and steering angle. Many walking motion clips with different strides and steering angles are available in our motion repertoire, and thus the character's moving path depends on both the structure (e.g., the number of steps) of the action string and its instantiation (e.g., a series of motion clips with different stride lengths and steering angles). The structure and its instantiation interact with each other in a way that the character may either take a few steps with long stride or more steps with shorter stride.

Our multi-level MCMC algorithm in Section 5 samples the space of plausible animations with respect to the production and semantic rules. The most plausible animation from the samples will be chosen (Figure 2(right)). The final step is motion editing to remove any mismatches in character coordination and synchronization. The animation is a collage of canned motion clips, so motion clips may not fit precisely with each other and in the virtual environment. We use a Laplacian motion editing technique [KHKL09] to get rid of any residual mismatches, foot sliding, and interpenetration.

4. Basketball Tactics Board

Although our motion grammar is a general method that generalizes easily to deal with various applications, this paper focuses on a

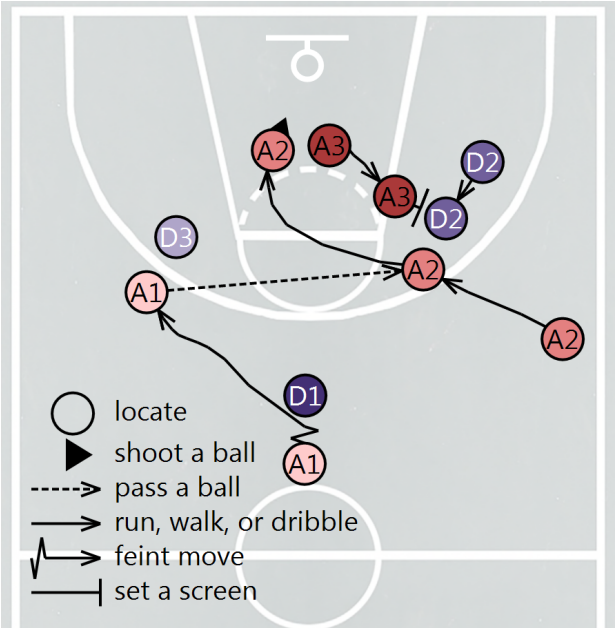

Figure 3: *A sketch-based interface for drawing basketball tactics diagrams*

single application domain (i.e., basketball) to discuss the grammar-based modeling process in-depth. The basketball play is highly-structured and the players move in a highly-coordinated manner. While the structures of basketball plays are derived from basketball rules and regulations, the semantic rules emerge from the game context that includes game strategies, coordinated tactics, and adversarial interaction between offensive and defensive players. We found that the motion grammar is an effective tool for describing the structure and semantics of basketball plays (see Appendix for the motion grammar with production and semantic rules).

Given the motion grammar for basketball plays, we can generate three-dimensional, full-body basketball animation from a diagram sketch on the tactics drawing board (Figure 3). Since the diagram is simple, sparse, and abstractive, reconstructing 3D animation from the diagram is an ill-conditioned problem. The motion grammar is a key to the reconstruction of structurally-valid, semantically-meaningful plays from a sparse sketch. Our sketch-based interface for the tactics board includes five elements:

Circle: The circle indicates the location of a player. Offensive players are shown in red-tones and defensive players are shown in blue-tones.

Solid arrow: A solid arrow from a circle to the goal rim indicates a player shooting a basketball.

Solid curve with arrow: A solid curve with an arrow end describes a moving path of a player who may walk, run, or dribble along the path.

Dashed line with arrow: A dashed line with an arrow end describes one player passing a basketball to the other player.

Zigzag line: The zigzag lines indicate feint moves and rapid pivot turns.

T-end: A moving path with a T-end indicates an offensive player setting a screen to block a defensive player.

The tactics diagram in Figure 3 implicates the structure and semantics of the play; the offensive player $A2$ outside the three-point line runs toward the free throw line to catch a pass from player $A1$, and dribbles the ball toward the goal rim to make a shoot. In the meantime, player $A3$ sets a screen to block the defensive player $D2$. A regular expression $\hat{X} = (walk|run)^*(catch)(dribble)^*(shoot)$ describes the role of the player $A2$ in the tactics. Any action string X fits to the diagram if the regular expression $\hat{X}$ matches a substring of X. Here, the substring may not be continuous in X. The actual computation is simple. We drop the asterisk symbols from the expression $\hat{X}$ and then the following equation evaluates how well X matches the description of an individual player.

$$g_{\text{user}} = \text{length}(\hat{X}) - \text{LCS}(X, \hat{X}), \tag{8}$$

where $\text{LCS}(X, \hat{X})$ is the length of the longest common subsequence of X and $\hat{X}$.

The diagram also generates a set of auxiliary semantic rules. The circles locate the characters at desired positions (f_{pos}). The direction of a character when shooting or passing a ball is also implied in the diagram (f_{dir}). Synchronization rules (f_{synch}) are set between reciprocal actions, for example, "passing a ball" and "catching a ball", or "setting a screen" and "pushing against the screen". Although it is not clearly specified in the diagram who is holding the ball at the beginning of the animation, the ball holder can be inferred by tracing the diagram backward from either shoot or pass elements. The diagram is infeasible if it has more than one ball holders or has a link traversing backward in time.

5. Motion Synthesis

The basketball tactics diagram implicates basketball plays governed by a motion grammar $\mathcal{G}$ and auxiliary semantic rules $\{f_k\}$ derived from the diagram. An animated basketball play is a tuple $Z = \big(\{X_i\}, \{Y_i\}, \{\mathcal{T}_i\}\big)$, where X_i is an action string of the i-th player, Y_i is an instance of X_i, and $\mathcal{T}_i$ is a parse tree of X_i. A set of play scenes form a probability distribution, and a play scene is likely to emerge with probability $P(Z) \propto P(X)P(Y)$ in Equation (6) and (7), where $X = \{X_i\}$ and $Y = \{Y_i\}$. Conceptually, synthesizing an animated scene from a diagram is to choose the most probable scene from the distribution with respect to our plausibility measures. The probability distribution of animated scenes is high-dimensional and very complex. The dimensionality of an animated scene includes the complexity of full-body poses, the dimension of time, and the coordination of multiple characters. Therefore, searching the most probable scene in such a high-dimensional space is computationally demanding.

We formulate the scene reconstruction in the framework of Markov Chain Monte Carlo (MCMC) methods, which are a class of algorithms for sampling from a high-dimensional probability distribution from which direct sampling is difficult. Specifically, the Metropolis-Hastings algorithm is an MCMC method for obtaining a sequence of random samples that approximate the target probability distribution. We modified and generalized the Metropolis-Hastings algorithm to cope with the complexity of animated scenes,

including the hierarchical structure of parse trees, the symbolic description of actions, and their instances in the virtual environment. Our formulation separates the structure and semantics of basketball plays into two layers. The structure layer is symbolic and tree-structured, while the semantic layer is continuous and spatiotemporal. Our novel multi-level MCMC algorithm can deal with the heterogeneous (structure vs semantics, symbolic vs continuous, and space vs time) nature of the scene reconstruction problem.

The Metropolis-Hastings algorithm begins with an arbitrary initial sample $Z^{(0)}$ and performs random walk to generates a Markov chain of random samples $\{Z^{(1)}, Z^{(2)}, \cdots\}$. The random walk requires a proposal density function $Q(Z';Z)$, which suggests a new proposal Z' given the previous sample Z. The proposal is accepted with the ratio

$$\alpha_{Z\to Z'} = \min\left(1, \frac{P(Z')Q(Z;Z')}{P(Z)Q(Z';Z)}\right). \tag{9}$$

If the new sample is rejected, the next sample is the previous sample Z. Repeating this process generates a chain of samples one by one. Theoretically, the chain of samples converge to the target probability distribution with an arbitrary proposal density function. Selecting Q, however, influences heavily the computational performance of the algorithm in practice. Therefore, defining an effective proposal density function is a key to the application of an MCMC algorithm.

Random Walks. We introduce two types of random jumps to generate a Markov chain. The first type $Q_I(Y';Y)$ substitutes a motion clip with another motion clip in Y, leaving the action strings X and its parse trees $\mathcal{T}$ remain unchanged. Note that the scene may have multiple characters, each of which has an action string, its instance, and a parse tree. The proposal density Q_I selects one of the characters and picks one motion clip among a series of the character's motion clips. A new motion clip is instantiated randomly from the same motion symbol that generated the original motion clip. The probability of suggesting a particular motion clip for substitution is proportional to the error induced by the motion clip in $-\log\big(P(Y)\big)$ from Equation (7). The motion clip will be more likely to be chosen if it violates semantic rules, involves in a collision, or the connection to its previous or next motion clip is not smooth.

The second type $Q_S(\mathcal{T}';\mathcal{T})$ makes a larger, structural change to the scene by replacing a subtree of a parse tree $\mathcal{T}_i \in \mathcal{T}$ with a new subtree. The root of any subtree is a non-terminal symbol T. We generate a new subtree randomly by applying production rules recursively starting from the root symbol and updating action strings accordingly. Motion clips are re-instantiated from new action symbols and therefore Y is changed as well. The scope of the change varies depending on the size of the subtree. The change can be as little as replacing a single action symbol with another, and as big as replacing the whole scene with a random scene. The proposal density should make a good balance among jumps of various sizes. The probability of suggesting a particular subtree for substitution is proportional to the error induced by the subtree. The error $E(T)$ is defined recursively:

$$E(T) = \Big(\prod_{T'\in \text{child}(T)} E(T')\Big)^{\frac{1}{n+1}} \tag{10}$$

which is the geometric mean of errors of the child nodes, and n is the number of child nodes. The rationale of using the geometric mean is as follows. If the errors in the child nodes are equally high or equally low, the parent node is equally likely to be suggested for substitution as its descendants. If error-prone nodes and error-less nodes are mixed in the descendants, the parent node is less likely to be suggested than its descendants. In this case, the proposal density tends to suggest error-prone descendant nodes selectively rather than make a big jump of substituting the parent node. We used the $(n+1)$-th root instead of the n-th root to favor small jumps moderately over larger jumps.

Algorithm 1 Multi-level MCMC

1: $Z^{(0)} \leftarrow (X^{(0)}, Y^{(0)}, \mathcal{T}^{(0)})$
2: **for** $i \leftarrow 1$ **to** N **do**
3: **repeat**
4: $\mathcal{T}' \sim Q_S(\mathcal{T}';\mathcal{T}^{(i-1)})$
5: $X' \leftarrow \text{string}(\mathcal{T}')$
6: **until** $P(X') > 0$
7: $\bar{Y}^{(0)} \leftarrow \text{instantiate}(X')$
8: **for** $j \leftarrow 1$ **to** M **do**
9: $Y' \sim Q_I(Y';\bar{Y}^{(j-1)})$
10: $\beta \leftarrow \min(1, \frac{P(Y')Q_I(\bar{Y}^{(j-1)};Y')}{P(\bar{Y}^{(j-1)})Q_I(Y';\bar{Y}^{(j-1)})})$
11: **if** random() $< \beta$ **then**
12: $\bar{Y}^{(j)} \leftarrow Y'$
13: **else**
14: $\bar{Y}^{(j)} \leftarrow \bar{Y}^{(j-1)}$
15: **end if**
16: **end for**
17: $Z' \leftarrow (X', \bar{Y}^{(M)}, \mathcal{T}')$
18: $\alpha = \min(1, \frac{P(Z')Q_S(\mathcal{T}^{(i-1)};\mathcal{T}')}{P(Z^{(i-1)})Q_S(\mathcal{T}';\mathcal{T}^{(i-1)})})$
19: **if** random() $< \alpha$ **then**
20: $Z^{(i)} \leftarrow Z'$
21: **else**
22: $Z^{(i)} \leftarrow Z^{(i-1)}$
23: **end if**
24: **end for**
25: **return** $\{Z^{(1)}, \ldots Z^{(N)}\}$

Multi-level MCMC Algorithm. Our algorithm begins with initial scene $Z^{(0)}$ that is structurally-plausible (i.e., $P(X^{(0)}) > 0$ in Equation (6)). It means that the action strings have valid parse trees and match the input tactics diagram. The algorithm has two nested loops. The outer loop is for structural jumps, while the inner loop is for optimizing the instantiation of the scene. The structural jumps are allowed only between structurally-plausible configurations (line 3–6). The semantic-layer MCMC instantiates a new scene from the structure suggestion X' and optimizes the scene with respect to semantic rules and quality measures in Equation (7) (line 7–16). random() is a function drawing a random number in $[0,1)$. The structure suggestion is either accepted or rejected based on a structure-layer MCMC method (line 17–23).

Bootstrapping. The bootstrapping is a process of finding a structurally-plausible initial configuration $(X^{(0)}, \mathcal{T}^{(0)})$ before our multi-level MCMC algorithm starts. Since action strings are always

Action	Clips	Action	Clips	Action	Clips
screen	9	hold	6	feint	10
block	20	guard	121	push	9
pass	41	fake_shot	11	pivot_l	10
pivot_r	14	dribble	146	shoot	27
catch	41	walk	64	run	72
layup	15			Total	598

Table 1: *The number of motion clips for each action symbol*

derived by using production rules and parse trees, $X^{(0)}$ is valid if $g_{\text{user}} = 0$. The bootstrapping is yet another MCMC algorithm that searches for a valid configuration with probability $P(X)$ and proposal density $Q_S(\mathcal{T}';\mathcal{T})$.

Parallel Tempering. Although MCMC algorithms are theoretically guaranteed to converge even in high dimensional space, the more difficult problem is to make it converge faster. The speed of convergence is closely related to determining proper size of jump proposals especially with complex structures such as motion grammars. When jumps are too big, proposals are easily rejected and thus new samples are rarely accepted. Using small jumps only, on the other hand, leads to trapping in local extrema. Instead of going through laborious parameter tuning, we can run different chains simultaneously using a parallel tempering scheme [Gey91]. We generate multiple copies of Markov chains with target distributions $P_n(Z) \propto P^{1/T_n}(Z)$ of different temperatures. The chain of high temperatures allows big jumps, while the chain of low temperature tends to search samples near local extrema with small jumps. An important feature of parallel tempering is exchanging samples at different temperatures based on the Metropolis criterion.

$$\alpha_{n\leftrightarrow n+1} = \min\left(1, \frac{P_n(Z_{n+1})P_{n+1}(Z_n)}{P_n(Z_n)P_{n+1}(Z_{n+1})}\right) \tag{11}$$

This swapping process gradually sends good samples to cool chains to stabilize the optimization process and bad examples to hot chains to explore new possibilities more aggressively. The simultaneous generation of multiple chains is easily implemented on multi-core/multi-thread architectures to parallelize the computation.

6. Results

Our motion grammar for basketball plays is given in Appendix. The grammar has two sets of production rules; one for offensive players and the other for defensive players. The grammar has 19 non-terminal symbols and 17 terminal symbols. The offensive players start with non-terminal symbol S_A to expand their action strings, while the defensive players start with S_D. Although we use two starting symbols in the example, the grammar can be adapted for individual player positions with a slight modification. Our basketball grammar is deterministic and in the LR(1) class of grammars. It means that any action string can be parsed sequentially by descending through productions recursively, picking the next production to expand using a single token of lookahead without backtracking.

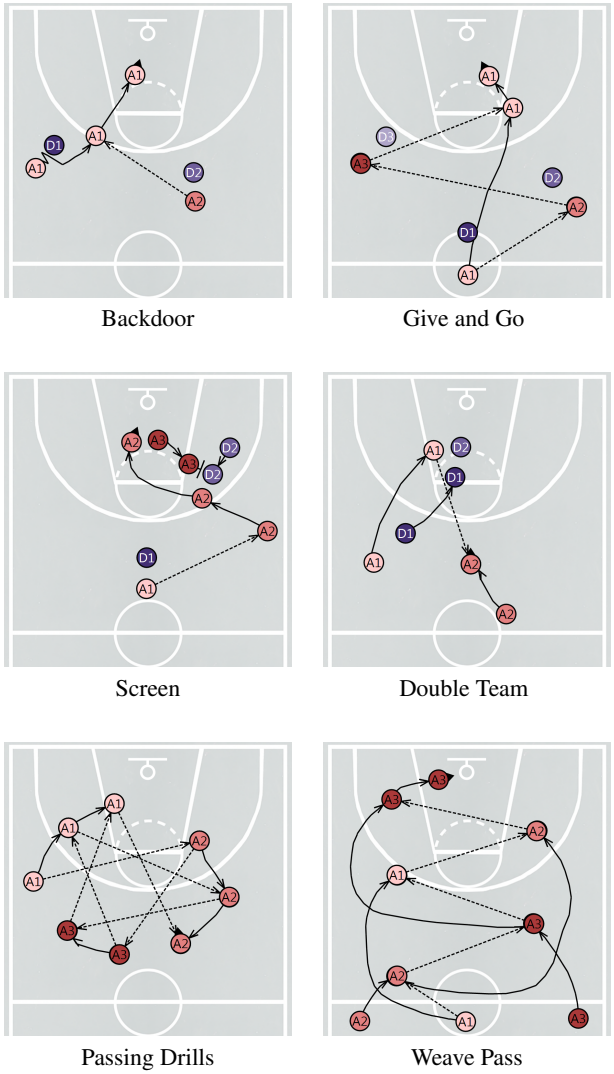

Figure 4: *Diagram sketches*

Even though our MCMC algorithm does not require deterministic grammars, it is convenient to have a deterministic parser when we want to use simpler motion synthesis algorithms based on A*-search, dynamic programming, or reinforcement learning.

The motion grammar can be easily incorporated into existing animation systems that are equipped with the motion graph or the finite state machine of the character's action. The motion grammar facilitates data-driven motion analysis and synthesis in many ways.

Validation: Given any motion data, the grammar parser can check if the motion is feasible with respect to the grammar governing its behavior. Grammar parsing reveals the hierarchical process how a sequence of actions are structured.

Action Suggestion: Assuming that a sequence of actions have already been performed, the parser can suggest a set of structurally-valid, plausible candidates for the next action. This allows us to choose a series of actions one-by-one sequentially

as if we use a motion graph. It means that the motion grammar can replace a motion graph seamlessly in existing animation systems.

Interactive Editing: Interactive manipulation of motion data entails continuous deformation of motion paths and large deformation often leads to structural changes on a sequence of actions [KHKL09]. Discrete, structural editing of motion data may introduce a new action in the middle of the sequence to cope with user manipulation, remove an action from the sequence, or reorder the actions to form a novel sequence. Motion grammars can parse valid structural changes and suggest new plausible sequences.

Motion Planning and Synthesis: Given a collection of motion fragments, there exist a number of motion planning and synthesis algorithms that can generate a sequence of motion fragments by splicing them together to satisfy user-specified goals and constraints. The motion grammar can facilitate any synthesis problem that entails splicing of actions.

Motion Data. We collected about 100 minutes' worth of basketball motion data, which captured two to four players simultaneously in each session. We segmented and labelled raw motion data manually to identify 598 motion clips, each of which is associated with a terminal symbol (Table 1). The motion clips are annotated with context information, including keyframes where important events occur (e.g., the moment of releasing a ball for shooting/passing and the moment of catching a ball), reciprocal relations between motion clips (e.g., pass and catch, shoot and block), and the relative pose of interacting players (e.g., the position and direction of a catcher relative to the passer, and the position and direction of a shooter relative to the goal rim). Our motion data do not include hand motion. The gaze direction, hand shapes, and ball trajectories are added to match the scene context in the post-processing phase.

Tactics Diagram. A number of offensive and defensive tactics/drill plans are available in basketball textbooks and coaching guides in the form of diagram sketches. We reproduce 3D animated scenes for representative diagram sketches (Figure 4) including offensive tactics (e.g., Pass and Shoot, Backdoor, Screen, and Give and Go), defensive tactics (e.g., Double Team), and drills (e.g., Weave Pass). Our algorithm works well with any diagram sketches as long as a vocabulary of actions are readily available. The interface system is very easy to use. The user can sketch an arbitrary diagram using a set of predefined elements (e.g., location, move, pass, shoot, screen, and feint). Then, the system checks the validity of the diagram and generates a 3D animated scene in a fully automatic manner.

Performance. Reconstructing play scenes from scratch took 3 to 10 minutes depending on the number of characters, the complexity of interaction among them, and the length of the animation (Table 2). Performance statistics are measured on a desktop PC equipped with an Intel Core i7-4820K CPU (8 cores, 3.7 GHz) and and 32 GB main memory, except for the 20-core parallel tempering example, which runs on another machine with slower Intel Xeon E7-4870 CPUs (2.4 GHz). The semantic layer sampling (the inner loop in the algorithm) takes more computation than the structure-layer sampling (the outer loop). The computation times for bootstrapping and motion editing in the post-processing phase are negligible comparing to the MCMC sampling. Parallel tempering effectively parallelizes the sampling procedure. As shown in Figure 5, the benefits of parallel tempering are twofold. It achieves a significant performance gain over single-chain sampling and, more importantly, finds a better solution effectively avoiding local extrema. The 6-core tempering converges to a better solution than a single Markov chain, and the 20-core tempering converges to a even better solution.

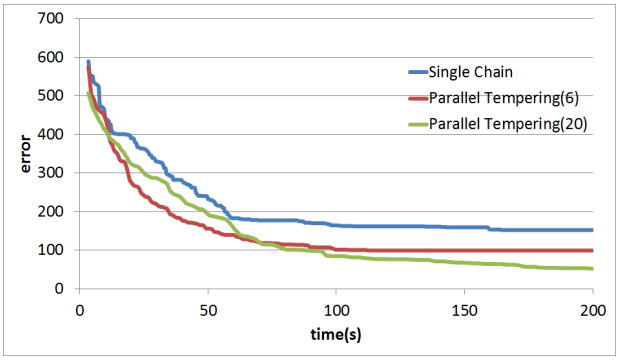

Figure 5: *A single Markov chain vs parallel tempering. The X-axis is the computation time in seconds, and the Y-axis is the error* $-\log(P(Y))$. *The performance is averaged over 10 trials for each of a single chain, 6-core tempering, and 20-core tempering.*

The optimization performance based on MCMC sampling depends heavily on the choice of the initial configuration. If the user changes the input diagram slightly, the animation can be updated incrementally by taking the previous result as the initial configuration for the subsequent optimization. Our algorithm allows the input diagram to be manipulated interactively and updates the corresponding animation at interactive rates with up to two players. The incremental update for a small change is in average two orders of magnitude faster than the full reconstruction from scratch, so it takes only several seconds to update the example scenes incrementally.

7. Discussion

We have presented the motion grammar as a general method for designing the behavior model of characters and generating animated scenes from a simple sketch. The rigorous formulation of the behavior model made it possible to synthesize the coordination and interaction among multiple characters, which are structurally-valid as well as semantically-meaningful. Even though our focus has been on animating basketball plays in this paper, our approach could be easily extended to deal with other types of scenes where there is a requirement for the structured behavioral patterns and the coordination of multiple interacting characters, such as sports, dancing, and social interactions.

Our motion grammar is a subset of the comprehensive grammar for basketball plays, since our motion grammar focused on modeling scenarios that are likely to appear in tactics plans. For example, consider a scenario that a ball is loose on the court and players compete for the ball. Such a scenario can occur in real basketball

Diagram	# of characters	# of motion instances	# of animation frames	Computation time (second)	
				Outer loop	Inner loop
Pass and Shoot	4	33	179	25	217
Screen	5	18	132	28	215
Backdoor	4	20	136	17	160
Give and Go	6	36	158	45	309
Double Team	4	22	170	25	244
Passing Drills	3	34	235	103	485
Weave Pass	3	26	245	86	436

Table 2: *Results and performance.*

plays, but it is unlikely to consider such an unusual scenario in a tactics plan. An advantage of our grammar-based approach is its flexibility and scalability. Even though our motion grammar is not comprehensive in the present form, it is easy to add new production and semantic rules incrementally to deal with a larger domain of basketball plays, situations, and scenarios.

The most time-consuming part of the grammar-based modeling is motion data segmentation and labelling. Although there have been significant technical advances recently in data-driven animation, automating motion segmentation and labelling are still challenging. Many game developer companies have already built domain-specific motion databases and finite state machines for character animation using a number of labelled motion clips. The motion grammar can be built on top of those readily-available motion databases to have more structure than a finite state machine can offer.

Recent basketball video games provide rich details of characters' motion using large, high-quality motion databases captured from professional players. The character animation algorithms in video games are usually simple and very efficient. The efficiency comes at a cost of compromising the quality of animation. Typically, game characters are allowed to slide or change its position or direction suddenly in an implausible manner. Game developers design finite state machines and character control mechanisms very carefully to make such artifacts less obtrusive. The goal of this work is different than what game developers are pursuing. Our MCMC algorithm is a general solution method to address the problem without compromising structural and semantic constraints.

Won et al [WLO*14] also addressed the animation of tightly-coupled multiple interacting characters from a sparse, high-level description, though their generate-and-rank method is quite different from ours and their problem domain (scripts-based description for fight scenes) is also different from our basketball tactics animation. Technically, the generate-and-rank method based on a motion graph and uniform random sampling is complementary to our motion grammars, which would provide more structures and better descriptions on their fight scenes. Uniform random sampling for motion synthesis can be replaced with our multi-level MCMC sampling to achieve significant performance improvements. Conversely, their generate-and-rank method can be a valuable supplement to our MCMC algorithm, which generates a chain of samples. Although generated samples are the best samples with respect to the plausibility measure, they are similar to each other due to the Markov property of random walks. The generate-and-rank algorithm would add diversity to the system by providing the user with multiple distinct representative samples. Our motion grammar also improves the flexibility of description. While the relative position and orientation of characters participating in each interaction event are fixed in their work, our motion grammar allows the relationships among characters to be flexibly described in rules so that a range (from tightly-coupled to loosely coupled) of interactions can be dealt with in our work.

An interesting direction for future research is inferring motion grammars automatically from a corpus of motion data. In computational linguistics, grammar induction has been an active research topic to learn a formal grammar from a corpus of texts. Grammar induction is also called grammatical inference. Learning weak structures and habitual patterns from loosely organized dance choreography and idling pedestrians seems feasible [PLL11]. However, learning stronger structures, such as basketball rules, directly from a database of basketball motions would be challenging even with state-of-the-art grammar induction algorithms. Probably, semantic reasoning and extra negative samples (violating basketball rules) would allow us to infer motion grammars from canned motion data.

Acknowledgements

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No.2011-0018340 and No. 2007-0056094). The authors would like to thank Junho Park for his help with experiments.

References

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

[BCvdPP08] BEAUDOIN P., COROS S., VAN DE PANNE M., POULIN P.: Motion-motif graphs. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer Animation* (2008), pp. 117–126. 2

[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 *Proceedings of Graphics Interface 2004* (2004), GI '04, pp. 185–194. 2

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4 (2010), 104:1–104:10. 3

[CKHL11] CHOI M. G., KIM M., HYUN K. L., LEE J.: Deformable motion: Squeezing into cluttered environments. *Computer Graphics Forum (Eurographics) 30*, 2 (2011), 445–453. 2

[DS13] DANTAM N., STILMAN M.: The motion grammar: Analysis of a linguistic method for robot control. *IEEE Transactions on Robotics 29*, 3 (2013), 704–718. 3

[Gey91] GEYER C. J.: Markov chain monte carlo maximum likelihood. *Proceedings of the 23rd Symposium on the Interface: Computing Science and Statistics* (1991), 156–163. 7

[GFA07] GUERRA-FILHO G., ALOIMONOS Y.: A language for human action. *Computer 40*, 5 (2007), 42–51. 3

[GFA10] GUERRA-FILHO G., ALOIMONOS Y.: The syntax of human actions and interactions. *Journal of Neurolinguistics*, 5 (2010), 500–514. 3

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), SIGGRAPH '98, pp. 33–42. 2

[HKHL13] HYUN K., KIM M., HWANG Y., LEE J.: Tiling motion patches. *IEEE Transactions on Visualization and Computer Graphics 19*, 11 (2013), 1923–1934. 2

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics (SIGGRAPH) 21*, 3 (2002), 473–482. 2

[KHKL09] KIM M., HYUN K., KIM J., LEE J.: Synchronized multi-character motion editing. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (2009), 79:1–79:9. 2, 4, 8

[KLLT08] KWON T., LEE K. H., LEE J., TAKAHASHI S.: Group motion editing. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (2008), 80:1–80:8. 2

[KSKL14] KIM J., SEOL Y., KWON T., LEE J.: Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 83:1–83:10. 2

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion Patches: Building blocks for virtual environments annotated with motion data. *ACM Transactions on Graphics (SIGGRAPH) 25*, 3 (2006), 898–906. 2

[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) 21*, 3 (2002), 491–500. 2, 4

[LHP05] LIU C. K., HERTZMANN A., POPOVIĆ Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (SIGGRAPH) 24*, 3 (2005), 1071–1081. 2

[LLKP11] LEVINE S., LEE Y., KOLTUN V., POPOVIĆ Z.: Space-time planning with parameterized locomotion controllers. *ACM Transactions on Graphics 30* (2011), 23:1–23:11. 2

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

[LWB*10] LEE Y., WAMPLER K., BERNSTEIN G., POPOVIĆ J., POPOVIĆ Z.: Motion fields for interactive character locomotion. *ACM Transactions on Graphics (SIGGRAPH ASIA) 29* (2010), 138:1–138:8. 2

[LWW08] LIPP M., WONKA P., WIMMER M.: Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (2008), 102:1–102:10. 3

[MC12] MIN J., CHAI J.: Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (SIGGRAPH ASIA) 31*, 6 (2012), 153:1–153:12. 2

[MCC09] MIN J., CHEN Y.-L., CHAI J.: Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics 29* (2009), 9:1–9:12. 2

[MP07] MCCANN J., POLLARD N.: Responsive characters from motion fragments. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3 (2007). 2

[PLL11] PARK J. P., LEE K. H., LEE J.: Finding syntactic structures from human motion data. *Computer Graphics Forum 30* (2011). 3, 9

[RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and Adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18* (1998), 32–40. 2

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3 (2007). 2

[SKSY08] SHUM H. P. H., KOMURA T., SHIRAISHI M., YAMAZAKI S.: Interaction patches for multi-character animation. *ACM Transactions on Graphics (SIGGRAPH ASIA) 27*, 5 (2008), 114:1–114:8. 2

[SKY12] SHUM H. P. H., KOMURA T., YAMAZAKI S.: Simulating multiple character interactions with collaborative and adversarial goals. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (May 2012), 741–752. 2

[SL06] SHIN H. J., LEE J.: Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds 17*, 3-4 (2006), 219–227. 2

[SO06] SHIN H. J., OH H. S.: Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the ACM SIGGRAPH/Eurographics 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 (SIGGRAPH) 26*, 3 (2007). 2

[WLO*14] WON J., LEE K., O'SULLIVAN C., HODGINS J. K., LEE J.: Generating and ranking diverse multi-character interactions. *ACM Transactions on Graphics (SIGGRAPH ASIA) 33*, 6 (2014), 219:1–219:12. 2, 9

Appendix

Starting symbols : $\{S_A, S_D\}$

Non-terminals : $\{S_A, Moves, Move, Attacks, Ball, Shoot, Catch, Feint, Runs, Pivot, PivotL, PivotR, Fake, Fakes, Dribble, Dribbles, S_D, Defenses, Defense\}$

Terminals : $\{run, walk, idle, screen, hold, catch, feint, pivot\_left, pivot\_right, dribble, pass, shoot, layup, fake\_shot, guard, block, push\}$

Production Rules :

Attack		
S_A	$\rightarrow$	$Moves\ Attacks$
S_A	$\rightarrow$	$Attacks$
$Moves$	$\rightarrow$	$Move \mid Move\ Moves$
$Move$	$\rightarrow$	$run \mid walk \mid idle \mid screen$
Ball play		
$Attacks$	$\rightarrow$	$Ball\ pass\ Moves\ Attacks$
$Attacks$	$\rightarrow$	$Ball\ Shoot \mid \varepsilon$
$Ball$	$\rightarrow$	$Catch\ Pivot\ Dribble$
Catch		
$Catch$	$\rightarrow$	$hold \mid catch \mid Feint\ catch$
$Feint$	$\rightarrow$	$feint\ Runs$
$Runs$	$\rightarrow$	$\varepsilon \mid run\ Runs$
Pivot		
$Pivot$	$\rightarrow$	$\varepsilon \mid PivotL\ Fake \mid PivotR\ Fake$
$PivotL$	$\rightarrow$	$pivot\_left \mid pivot\_left\ PivotL$
$PivotR$	$\rightarrow$	$pivot\_right \mid pivot\_right\ PivotR$
Fake		
$Fake$	$\rightarrow$	$\varepsilon \mid Fakes$
$Fakes$	$\rightarrow$	$fake\_shot \mid fake\_shot\ Fakes$
Dribble		
$Dribble$	$\rightarrow$	$\varepsilon \mid Dribbles\ Pivot$
$Dribbles$	$\rightarrow$	$dribble \mid dribble\ Dribbles$
Shoot		
$Shoot$	$\rightarrow$	$layup \mid shoot$
Defense		
S_D	$\rightarrow$	$Defenses$
$Defenses$	$\rightarrow$	$Defense \mid Defense\ Defenses$
$Defense$	$\rightarrow$	$run^\dagger \mid walk^\dagger \mid guard$
$Defense$	$\rightarrow$	$block \mid push$

Semantic Rules :

$pass$:	$f_{\text{dir}}(\text{receiver}),$ $f_{\text{synch}}(\text{receiver}, catch, \Delta t)$
$layup$:	$f_{\text{dir}}(\text{rim})$ and $f_{\text{dist}}(\text{rim}, \leq 2.5m)$
$shoot$:	$f_{\text{dir}}(\text{rim})$ and $f_{\text{dist}}(\text{rim}, > 1.5m)$
$feint$:	$f_{\text{dist}}(\text{opponent}, < 1m)$
$fake\_shot$:	$f_{\text{dist}}(\text{opponent}, < 1m)$
$screen$:	$f_{\text{dir}}(\text{receiver}),$ $f_{\text{synch}}(\text{opponent}, push, 0)$
$run^\dagger, walk^\dagger$:	$f_{\text{dir}}(\text{opponent}),\ f_{\text{line}}(\text{opponent}, \text{rim})$
$guard$:	$f_{\text{dist}}(\text{opponent}, > 50cm \text{ and } \leq 2m)$
$block$:	$f_{\text{dir}}(\text{opponent}),$ $f_{\text{line}}(\text{opponent}, \text{rim}),$ $f_{\text{synch}}(\text{opponent}, \{shoot, fake\_shot\}, 0)$