COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.376

SPECIAL ISSUE PAPER

A finite state machine based on topology coordinates for wrestling games

Edmond S. L. Ho* and Taku Komura

Institute of Perception, Action and Behaviour School of Informatics, University of Edinburgh Informatics Forum, 10 Crichton Street, Edinburgh, UK

ABSTRACT

This paper proposes a new framework to simulate the real-time attack-and-defense interactions by two virtual wrestlers in 3D computer games. The characters are controlled individually by two different players—one player controls the attacker and the other controls the defender. A finite state machine of attacks and defenses based on topology coordinates is precomputed and used to control the virtual wrestlers during the game play. As the states are represented by topology coordinates, which is an abstract representation for the spatial relationship of the bodies, the players have much more degree of freedom to control the virtual characters even during attacks and defenses. Experimental results show the methodology can simulate realistic competitive interactions of wrestling in real time, which is difficult by previous methods. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS

character animation; motion capture; interactive games

***Correspondence**

E. S. L. Ho, Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh, Informatics Forum, 10 Crichton Street, Edinburgh, UK. E-mail: S.L.Ho@sms.ed.ac.uk

1. INTRODUCTION

Wrestling is a major field in 3D computer games. The motions in wrestling involve close contacts between the characters such as squeezing and locking. Simulating such motions is not easy as they involve a great amount of close contacts and collision avoidance. In fact, in most of the wrestling games, such motions are designed carefully in advance by the animators and the game players have little control over the characters once complex tangling motions start. This is completely different from real wrestling—wrestlers have lots of degrees of freedom to escape from the attackers, and attackers need to carefully select their motions to lock the defender. Such complex interactions of the wrestlers are rarely simulated in the existing wrestling games.

In this paper, we make use of the topology coordinates [1] in order to simulate such complex interactions in real time. A finite state machine of attacks and defenses based on topology coordinates is precomputed and used to present the game players the next possible moves for attacking/escaping from the opponent. The player who has an opportunity to attack is shown the possible actions on the list of icons. The characters can also be controlled kinematically by using inverse kinematics (IK).

Our interface provides large degrees of freedom to the game players while minimizing the complexity of control, which can increase the attractiveness of wrestling games.

2. RELATED WORK

We first briefly review the recent wrestling games and their interfaces. Next, we review researches of two topics that are related to character control in wrestling games—real-time character control and close interactions of multiple characters.

Wrestling games have been attracting millions of game players around the world and has become one of the major categories in computer games. In the old games, [2,3] the attacks were limited to hits such as punches, kicks, or chops. Therefore, the users could only repeatedly press the buttons to give large damage to the opponent character.

The recent advanced games [4] allow the characters to conduct complex tangling attacks such as back drops, rear-choke hold and full nelson attacks. In order to launch such motions, the user is supposed to press the button at

the correct pose and timing, or select the part to attack by a pointing device.

Motions that involve tangling are usually just replayed during run-time as real-time editing of such motions can easily result in collision and penetration of the body segments. The attractiveness of the games will be greatly enhanced if the game players have access to the details of the tangling motions during the game play.

Real-time character control: Here we review a number of techniques which are useful for real-time control of the virtual wrestlers. We first review techniques of IK which is a basic technique to edit character motions, and then their extensions to handle dynamics and tangling motions.

IK [5–9] is a basic technique that is often used for real-time control of characters. Methods to control characters of arbitrary morphology [10,11] have also been proposed. IK methods can be divided into (1) CCD-based approaches, [12] (2) analytical approaches, [13] (3) particle-based approaches, [10,11] (4) quadratic programming based approaches [5,6,9], and (5) example-based approaches [14,15].

Among these approaches, the quadratic programming based approach has an advantage to simulate wrestling motions as it can enclose constraints based on dynamics [16–18] and topological relationships [1] into the solver. Our approach is built on top of Ho and Komura, [1] which propose to handle tangling motions by adding constraints into an optimization-based IK solver.

Multiple Character Interactions: The simulation of interactions between multiple characters has many applications such as computer games, virtual environments, and films. Liu *et al* [19]. simulate the close dense interactions of two characters by repetitively updating the motion of each character by spacetime constraints. Lee and Lee [20] simulate the boxing match by using reinforcement learning. Treuille *et al* [21]. also use reinforcement learning to simulate the pedestrians avoiding each other. Shum *et al*. [22] use min–max search to find the optimal action in a competitive environment. They also propose a real-time approach based on an automatically produced finite state machine [23]. These researches do not handle very close interactions such as holding or wrestling. Ho and Komura [24] generate wrestling motions by finding the topological relationship of characters from the template postures and use PD control to simulate movements where the topological relationship is kept the same. If we want to simulate a scene where the topological relationship of characters changes in time, we cannot apply such a method. A method to dynamically update the postures and the topological relationships is required. Reference 25]Komura::TVCG2009 propose to evaluate the similarity of character postures based on the topological relationships. When equivalent postures are found, the postures are linearly interpolated at the level of generalized coordinates. However, no method has enabled game players to interactively change the topological relationship of virtual characters in real time. We propose such an approach in this paper.

3. OVERVIEW

We first prepare a finite state machine of two characters wrestling that is based on topology coordinates. The animator first prepares a number of representative postures of wrestling attacks by keyframing the characters by topology coordinates. Using these postures, the system generates a finite state machine of wrestling. Each node represents a state of two characters tangled with each other and transition represents the change in the way the characters are tangled, which results in either newly forming a tangle between the body components or desolving existing ones.

During runtime, the topology coordinates of the two virtual wrestlers are computed based on their postures, and the corresponding node in the finite state machine is found. The players are then shown the list of possible attacks that they can launch from the current posture. The user also has a choice to control the characters directly by IK to get away from the attacks and holds by the opponent player.

4. METHODOLOGY

We first briefly review the topology coordinates, [1] which is the basic technique used to control the tangling motions. Next, how we produce the finite state machine based on the topology coordinates is explained. Finally, we explain how the topology coordinates can be applied to the control of virtual wrestlers in computer games.

4.1. Topology coordinates

Topology coordinates enable characters to tangle their bodies with other characters without conducting global path planning methods. The topology coordinates are composed of three attributes, the writhe, center, and density. The first attribute *writhe* counts how much the two curves are twisting around each other. Writhe can be calculated by using Gauss Linking Integral (GLI) [26] by integrating along the two curves γ_1 and γ_2 as

$$\mathrm{GLI}(\gamma_1, \gamma_2) = \frac{1}{4\pi} \int_{\gamma_1} \int_{\gamma_2} \frac{\mathrm{d}\gamma_1 \times \mathrm{d}\gamma_2 \cdot (\gamma_1 - \gamma_2)}{\|\gamma_1 - \gamma_2\|^3} \qquad (1)$$

where $\times$ and $\cdot$ are cross product and dot product operators, respectively. The GLI computes the average number of crossings when viewing the tangle from all directions.

Curves can twist around each other in various ways. In order to further specify the status of the two chains, we introduce the other two attributes, *center* and *density*. Examples of changing these attributes for a pair of strands are shown in Figure 1. The center, which is composed of two scalar parameters, explains the center location of the twisted area, relative to each strand. The density, which is a single scalar parameter, explains how much the twisted area is concentrated at one location along the strands. When the density is zero, the twist is spread out all over the two strands.

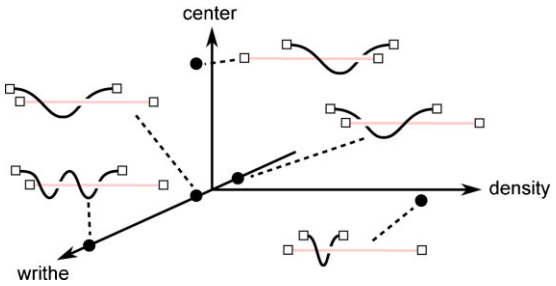

Figure 1. The three axes in topology space : writhe, center, and density. The center, which specifies the central location of the twist relative to each strand, is actually composed of two scalar parameters, although it is represented by a single axis in this figure. The density tells which strand is playing the major role to compose the twist.

When the density value is either very large or very small, we can say one strand is playing a major role to compose the twist, as it is twisting around the other strand which is kept relatively straight (Figure 1). When the density turns from negative to positive, or *vice versa*, the strand playing the major role switches.

4.2. Controlling characters by topology coordinates

We represent the bone structure of the virtual wrestlers by a set of line segments. Therefore, now we will mathematically define the topology coordinates of serial chains. Let us assume we have two chains S_1 and S_2, each composed of n_1 and n_2 line segments, connected by revolute, universal or gimbal joints (Figure 2). In this case, we can compute the total writhe by summing the writhes by each pair of segments:

$$w = \mathrm{GLI}(S_1, S_2) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} T_{i,j} \tag{2}$$

where w represents the writhe, $T_{i,j}$ is the writhe between segment i on $S1$ and j on $S2$.

Let us define a $n_1 \times n_2$ matrix $\mathbf{T}$ whose (i, j)th element is $T_{i,j}$, and call this the *writhe matrix*. The writhe matrix explains how much each pair of segments from S_1 and S_2 contribute to the total writhe value. Various twists of two serial chains and the corresponding writhe matrices are shown in Figure 3.

The topology coordinates can be updated by changing the distribution of the elements in the writhe matrix using basic operations such as rotation, translation, and scaling. Rotating the elements results in changing the density. Translating the elements results in changing the center. Scaling the whole matrix results in changing the writhe. Let us define these operations by $R(M, d)$, $Tr(M, c)$, and $S(M, w)$, respectively, where M is the input matrix and (d, c, w) are topology coordinates, each of which representing the density, center and writhe, respectively.

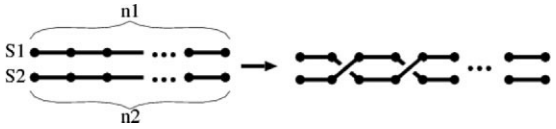

Figure 2. Twisting a chain of line segments around each other.

Rather than directly manipulating the writhe matrix of the characters, we first compute an ideal, desired writhe matrix and try to minimize the difference of the character's writhe matrix and the desired writhe matrix. The desired writhe matrix T_d that corresponds to topology coordinates (d, c, w) is computed by sequentially applying $R()$, $Tr()$ and $S()$ to a matrix $\mathbf{I}$, which is a $n_1 \times n_2$ matrix who has values evenly distributed at the $(n_2 + 1)/2$th column if n_2 is odd, or at both the $n_2/2$ and $n_2/2 + 1$th column if it is even:

$$\mathbf{T} = S\left(Tr\left(R\left(\mathbf{I}, d - \frac{\pi}{4}\right), \mathbf{c}\right), w\right) \tag{3}$$

where

$$\mathbf{I} = \begin{cases} \begin{pmatrix} 0\cdots, \frac{1}{n_1}, \cdots, 0 \\ \vdots \\ 0\cdots, \frac{1}{n_1}, \cdots, 0 \end{pmatrix} (n_2 \text{ is odd}) \\ \begin{pmatrix} 0\cdots, \frac{1}{2n_1}, \frac{1}{2n_1}, \cdots, 0 \\ \vdots \\ 0\cdots, \frac{1}{2n_1}, \frac{1}{2n_1}, \cdots, 0 \end{pmatrix} (n_2 \text{ is even}) \end{cases} \tag{4}$$

and $\frac{\pi}{4}$ is an offset to adjust the density d due to its definition [1].

Once the desired writhe matrix T_d is computed, the character is guided to the desired posture by updating the generalized coordinates so that the writhe matrix T of the character becomes similar to the desired writhe matrix T_d. This problem is solved by quadratic programming:

$$\min_{\Delta\mathbf{q_1}, \Delta\mathbf{q_2}, \boldsymbol{\delta}} \|\Delta\mathbf{q_1}\|^2 + \|\Delta\mathbf{q_2}\|^2 + \|\boldsymbol{\delta}\|^2 \text{ s.t.} \tag{5}$$

$$\Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} + \frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\mathbf{q_2} \tag{6}$$

$$|T_{i,j} + \Delta T_{i,j}| \le \sigma (1 \le i \le n_1, 1 \le j \le n_2) \tag{7}$$

$$\mathbf{T} + \Delta\mathbf{T} - \mathbf{T_d} + \boldsymbol{\delta} = \mathbf{0} \tag{8}$$

$$\mathbf{r} = \mathbf{J_1}\Delta\mathbf{q_1} + \mathbf{J_2}\Delta\mathbf{q_2} \tag{9}$$

where $(\mathbf{q_1}, \mathbf{q_2})$ are the generalized coordinates of the two chains, $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$ are their updates to be made at this iteration, $\Delta\mathbf{T}$ is the update of the writhe matrix, σ is a threshold, that is set to 0.2 in our experiments to avoid the segments to approach too close to each other, $\boldsymbol{\delta}$ is a vector of slack parameters introduced to minimize the difference of the desired writhe matrix and that of the controlled characters, Equation (9) represents the other kinematical constraints which can be linearized with respect to $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$ when the movement is small, such as the movements of any parts

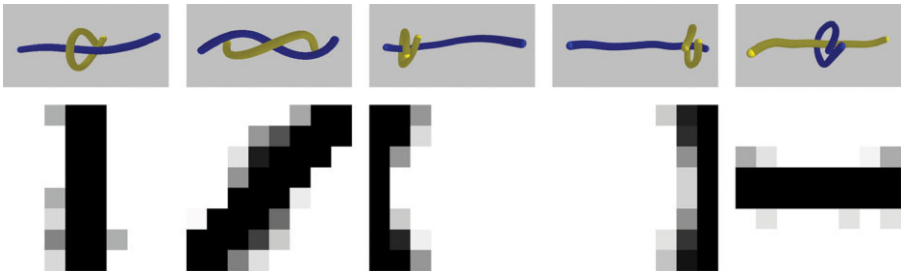

Figure 3. Tangles with different density and center (upper), and the distribution of elements with large absolute values in the corresponding writhe matrices (lower). The darkness represents the amplitude of the absolute value.

of the body in Cartesian coordinates or the center of mass. $\mathbf{J_1}, \mathbf{J_2}$ are the Jacobians of this constraint, and $\mathbf{r}$ is the linearized output of this constraint. The updated generalized coordinates $(\mathbf{q_1} + \Delta\mathbf{q_1}, \mathbf{q_2} + \Delta\mathbf{q_2})$ correspond to the target topology coordinate at the next time step, $(w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$. By solving Equation (5) at every time step, we simulate the interactions of two virtual wrestlers.

4.3. Finite state machine for wrestling

In this subsection, we explain how a finite state machine of two characters wrestling are produced from the keyframe postures designed using the topology coordinates.

The animator first prepares postures of two characters wrestling by updating the topology coordinates of the body segments. Scroll-bars that let the user adjust the writhe, center and density values are provided to generate different configurations. A view of the interface for editing the postures by the topology coordinates is shown in Figure 4.

Once the postures are designed, we start to produce the FSM. First, the designed postures are added as new states of the FSM. The topological status of the designed postures are evaluated based on the concept of rational tangles [25]. The rational tangles between all the routes that connect the end effectors are computed. Next, we find all the shortest paths from the designed postures to the untangled states in which the two characters simply stand next to each other. The limbs are untangled one by one, and each of such states are added into the state machine as well. Let us call these states intermediate states. Many intermediate states might be shared between different attacks.

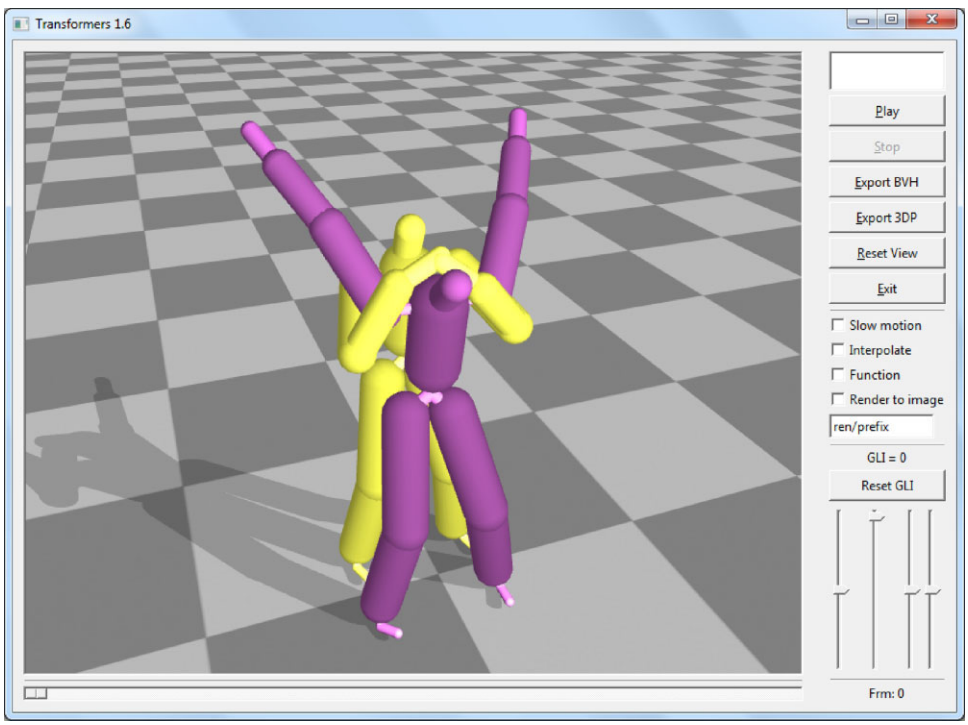

Figure 4. An interface to edit the postures of the wrestling characters by the topology coordinates. The animator specifies the limbs to be tangled and adjust their topology coordinates by the scroll bar at the right.

Finally, the system creates the FSM by connecting the nodes (postures) with similar topological states. Inspired by the work in Reference [25], we connect the postures/states if the absolute differences of the writhe between every pair of routes are less than a threshold of 0.5. An example finite state machine of one character attacking another from behind is shown in Figure 5.

The FSM mode starts only when the characters are close enough and one of the player launches an attacking action. When the characters are separate from each other in the beginning, they are not in the state of the FSM. Both characters freely move around according to the user input until either of them reach a state the user can launch a new attack. As shown in Figure 6, the possible choices are shown on the screen and the user selects one of them by the mouse.

There are also states that the defender can start an attack—if the player controlling the defender reacts faster than the player controlling the attacker to launch such an attack, the status of the fight can be switched. The user who has selected an action earlier becomes the attacker.

4.4. Real-time control of wrestlers

In this subsection, we explain how to simulate the interactions of two wrestling characters, each controlled by different game players. We assume one character is attacking the other character by moves that involve a lot of tangling.

Let us first give an overview of the computation of the characters' motions. The motion of each character is computed sequentially by two different quadratic programming problems. The attacker's movement is guided by the topology coordinates—once the attack is specified, the attacker tries to tangle its body with the defender's body according to the target configuration. The attack can be switched in the middle if the player thinks a different attack is more effective under the current configuration. The defender needs to escape from the attacks by controlling the body segments involved in the tangling process. The player uses a pointing device to move a body segment and the posture of the defender is computed by IK based on quadratic programming.

Now the method to control the attacker is explained. The attacker's motion is determined based on the defender's current posture and the target topology coordinates in the next time step. Let us assume the configurations of the attacker and defender are represented by $\mathbf{q_1}$ and $\mathbf{q_2}$, respectively. Solving Equation (8) is not a good idea to compute the motion of the attacker, as the updates of the attacker's movements takes into account the movements of the defender in the next time step. This makes the defender difficult to escape from the attacker. Therefore, we solve the following problem for computing the motion of the attacker:

$$\min_{\Delta\mathbf{q_1},\delta} \|\Delta\mathbf{q_1}\|^2 + \|\boldsymbol{\delta}\|^2 \text{s.t.} \tag{10}$$

$$\Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} \tag{11}$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma(1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{12}$$

$$\mathbf{T} + \Delta\mathbf{T} - \mathbf{T_d} + \boldsymbol{\delta} = \mathbf{0} \tag{13}$$

$$\mathbf{r_1} = \mathbf{J_1}\Delta\mathbf{q_1} \tag{14}$$

where $\mathbf{r_1}$ represents the kinematic parameters of the attacker.

This technique adds an effect of physiological delay for launching a response motion with respect to the defender's movements, which increases the realism of the interaction. It also adds an essence of a game play to the interaction between the two virtual wrestlers as the attacker may not be able to achieve the target topology coordinates if the defender is controlled well.

Next, the defender's movement is computed by solving the following IK problem:

$$\min_{\Delta\mathbf{q_2}} \|\Delta\mathbf{q_2}\|^2 \text{s.t.} \tag{15}$$

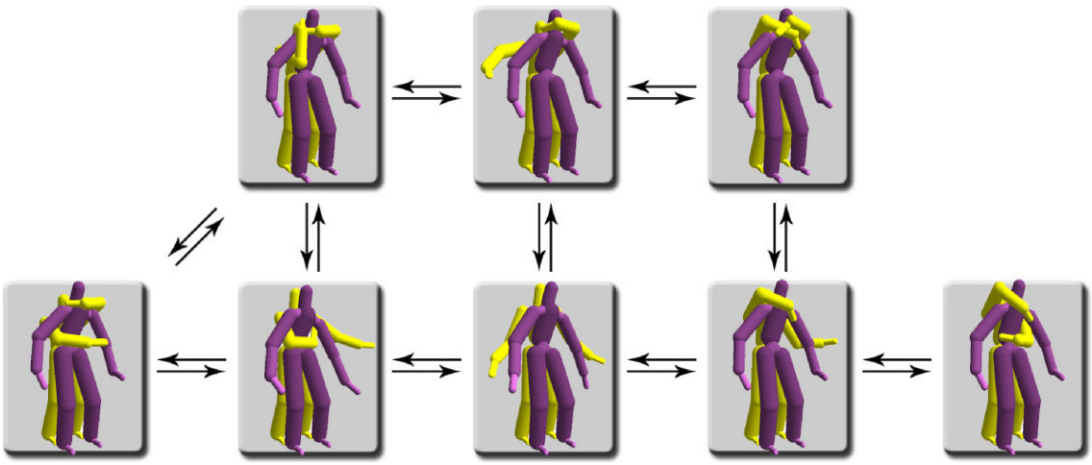

Figure 5. A finite state machine of two people wrestling when one character at the back of the other.

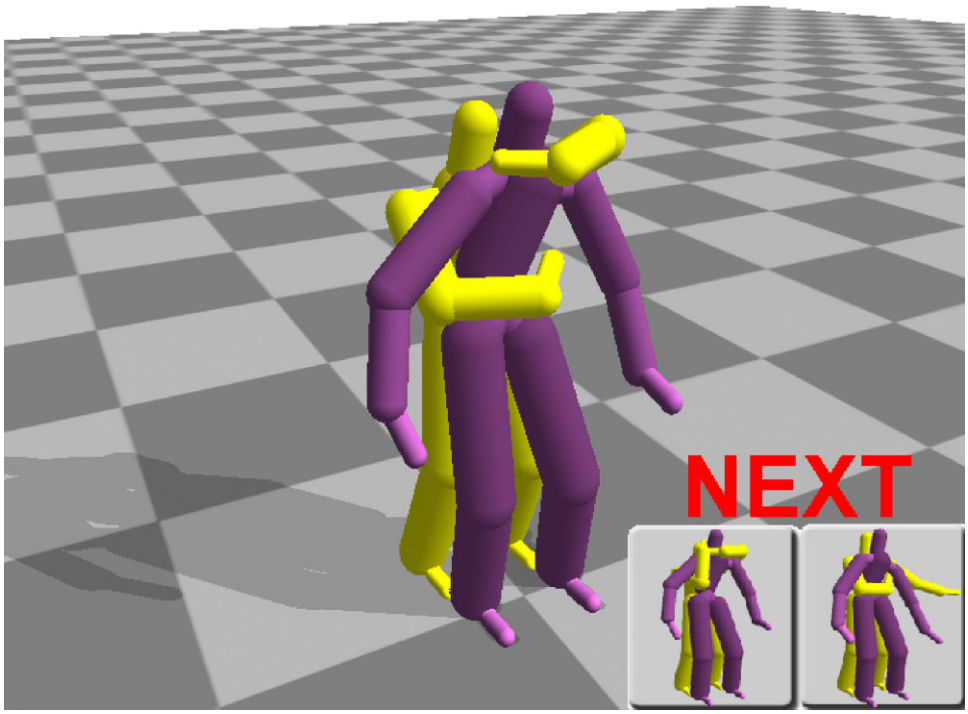

Figure 6. Once an intermediate state of the FSM is reached, the possible transitions are shown to the user.

$$\Delta \mathbf{T} = \frac{\partial \mathbf{T}}{\partial \mathbf{q_2}} \Delta \mathbf{q_2} \quad (16)$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma (1 \leq i \leq n_1, 1 \leq j \leq n_2) \quad (17)$$

$$\mathbf{r_2} = \mathbf{J_2} \Delta \mathbf{q_2} \quad (18)$$

where $\mathbf{r_2}$ represents the kinematic parameters of the defender, based on the input from the pointing device and any other kinematic constraints. The GLI between every segment pairs are still considered to avoid penetrations.

The attacker's motion is computed first by solving Equation (10) and the attacker's posture is updated. Then, the defender's motion is computed by solving Equation (15). Once both character's motion is updated, the time counter is increased.

The key point of controlling the defender is to move the body such that it can efficiently escape from the attacks. Such movements are those which can reduce the writhe value by little motion. For example, when the attacker starts to shift to a configuration of a rear-choke hold (Figure 7a), which is an attack to squeeze the neck from behind, the most efficient way to escape is to knee down at the last moment, which requires little movement. On the other hand, if the attacking player can predict such escaping moves of the defender and switch to another move that can make use

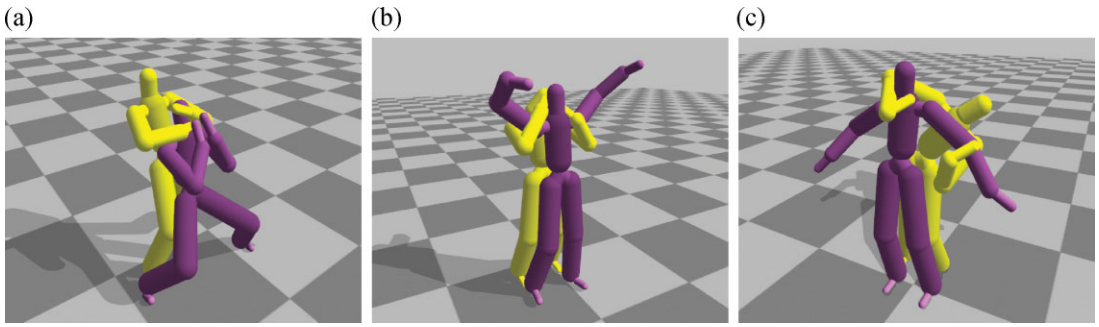

Figure 7. Various wrestling interactions created by the proposed method.

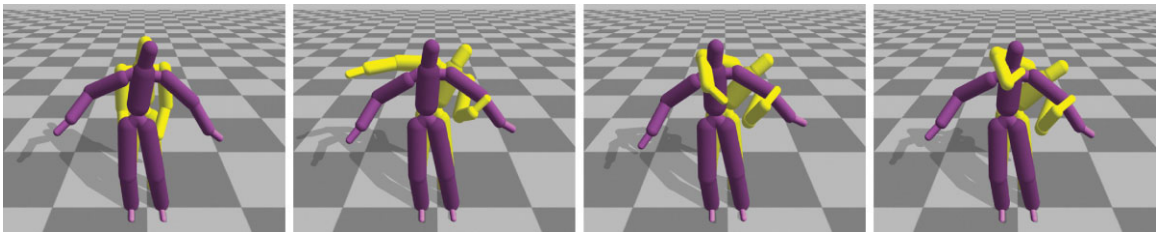

Figure 8. Without controlling the defender (in purple), the attacker (in yellow) can tangle with the defender easily by changing the topology coordinates.

of such movements, the player can efficiently tangle the attacker's body to the defender.

Sometimes the attacker has difficulty to conduct the attack due to the bad posture of the attacker. In order to deal with such cases, we also allow the attacker to switch to a control mode similar to the defender—specifying the motion by IK. In such case, the body is moved such that the already tangled parts are kept tangled as it is. In order to increase the DOF of the control, only the writhe constraint is satisfied but not those of the center and density.

5. EXPERIMENTAL RESULTS

We have simulated a number of interactions between two virtual wrestlers both controlled by game players. Those include rear-choke hold (Figure 7a), Full Nelson hold (Figure 7b) and a number of different squeezing motions from the back. The virtual wrestlers start from separate postures and the attacker approaches to the defender to tangle its body with it.

In our demos, we used a human character model of 42 DOF. All DOFs are used when controlling the characters. For the human character model, when one character is controlled, we can obtain an interactive rate of 40 frames/second when using a Pentium IV 3.2 GHz PC. When two characters are simultaneously controlled, we can still obtain a frame rate of 30 frames/second. We use ILOG CPLEX 9.1 [27] as our quadratic programming solver.

The movement of the attacker is controlled by specifying the topology coordinates. In the first animation, the attacker performed an attack by tangling (i) it's right arm with the neck of the defender and (ii) it's left arm with the left arm of the defender. Without controlling the defender, this attack can be performed easily by changing the topology coordinates of the attacker as shown in Figure 8.

In the second example, the defender tries to escape from the attacks shown in Figure 8. In our implementation, the player can specify kinematic constraints on the defender by dragging the body segments using a pointing device. In Figure 9, the player dragged the left hand (colored in red) of the defender. However, the defender fails to escape from the attack as it was not controlled quick enough. In Figure 10, the defender escapes from the attack successfully by moving the torso quickly and vigorously.

The attacker can also switch to another wrestling move in the middle of the attack. In the third example, the defender escapes from the attack by blocking the right arm of the attacker in the early stage of the interaction. Then the attacker switches to another wrestling move by tangling its

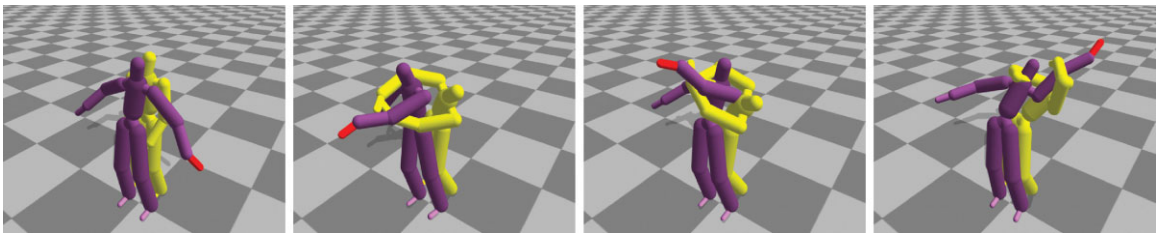

Figure 9. The defender (in purple) cannot escape from the attack if the player does not control it quick enough.

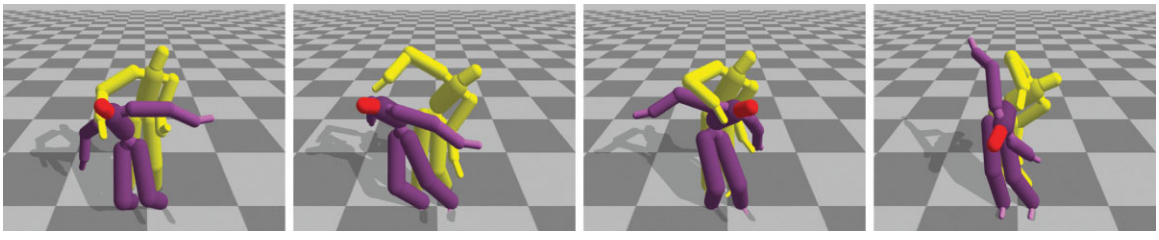

Figure 10. The defender (in purple) escapes from the attack.

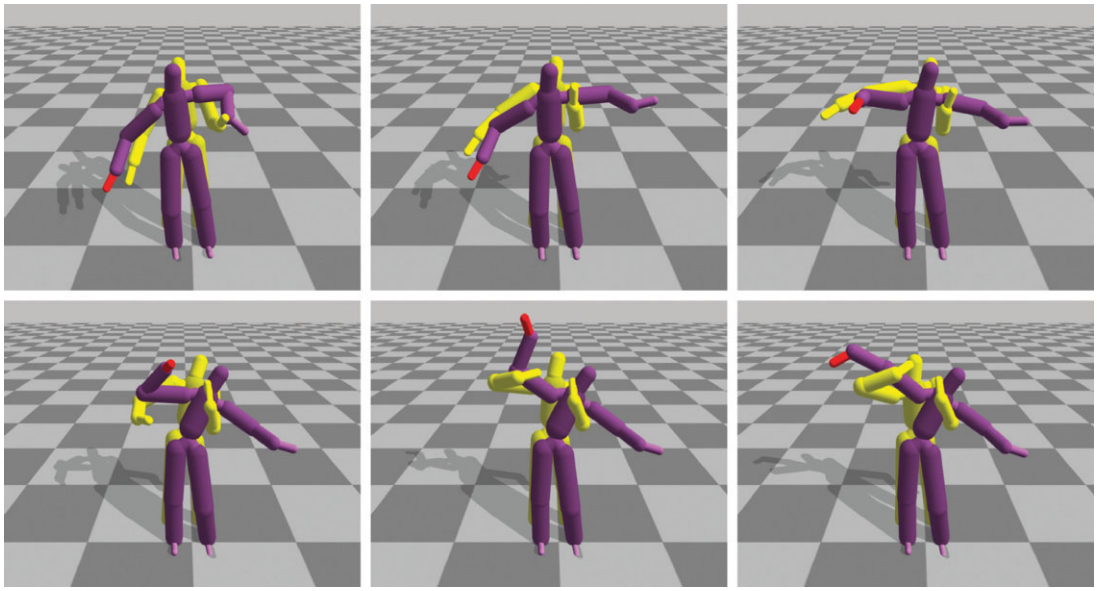

Figure 11. The attacker (in yellow) switched the attack in the middle in order to lock the defender (in purple).

right arm with the right arm of the defender and its left arm with the torso and neck of the defender. Finally the attacker successfully locks with the defender. The screenshots of this interaction are shown in Figure 11.

6. DISCUSSIONS AND FUTURE WORK

The introduction of an FSM in the level of topology coordinates is a new abstraction that increases the degrees of freedom of the players while keeping the movements of the characters manageable. In the previous wrestling games, the movements of the characters are strictly fixed and the FSM are prepared at the level of actions which results in monotonic game plays. We further divide the FSM into the level of topology coordinates which gives freedom to the players switching to different attacks in the middle while making use of the already tangled parts. Such an FSM can be easily handled by topology coordinates and the transition motion will vary according to the posture of the characters and the motion of the defender whose body is controlled by IK. This will make the game more attractive as the movements of the characters differ from time to time. Note that this can only be done by using topology coordinates as controlling wrestling characters at the level of joint angles can easily cause a lot of collisions and penetrations of the body components.

By computing the movements of the two characters by two independent quadratic programming problems, we can increase the reality of the interactions and the essence of the game play. Our experimental results show that the application of topology coordinates greatly increases the attractiveness of wrestling games from the following view points:

(1) The players, especially the defender has great access to the kinematics of the virtual wrestlers which increases the variety of movements.
(2) Although the kinematics are controlled interactively, the topology coordinates automatically guide the characters to avoid collisions and penetrations.

The methodology is not limited to wrestling games—it can be applied to other games that involve close interactions such as dancing. For examples, we can produce an FSM of two dancing characters and make use of it to let a real dancer wearing a motion capture system to dance with a virtual partner who reacts to the movements of the real dancer. The FSM can be used to evaluate the current state of the dance and predict the possible moves in the future.

In the future, we would like to conduct a user study to examine our user interface and seek for better ways to control the characters.

REFERENCES

1. Ho ESL, Komura T. Character motion synthesis by topology coordinates. *Computer Graphics Forum* 2009; **28**(2): 299–308.
2. *Nintendo Co. Ltd. Pro Wrestling*, 1986.
3. *Sculptured Software Inc. WWF Super WrestleMania*, 1992.
4. *THQ Inc. WWE: Smackdown vs Raw*. Available at: http://www.smackdownvsraw.com

5. Whitney DE. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems* 1969; **10**: 47–53.
6. Nakamura Y, Hanafusa H. Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control* 1986; **108**: 163–171.
7. Phillips CB, Zhao J, Badler NI. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Computer Graphics* 1990; **24**(2): 245–250.
8. Zhao J, Badler NI. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics* 1994; **13**(4): 313–336.
9. Yamane K, Nakamura Y. Natural motion animation through constraining and deconstraining at will. *IEEE Transactions on Visualization and Computer Graphics* 2003; **9**(3): 352–360.
10. Jakobsen T. Advanced character physics. *In Game Developers Conference Proceedings*, 2001; 383–401.
11. Hecker C, Raabe B, Enslow RW, DeWeese J, Maynard J, van Prooijen K. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics* 2008; **27**(3): 1–11.
12. Kulpa R, Multon F, Arnaldi B. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum* 2005; **24**(3): 343–351.
13. Lee J, Shin SY. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of SIGGRAPH'99*, 1999; 39–48.
14. Rose CF, Peter-pike I, Sloan J, Cohen MF. Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum* 2001; **20**: 239–250.
15. Kovar L, Gleicher M. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics* 2004; **23**(3): 559–568.
16. Yamane K, Nakamura Y. Dynamics filter—concept and implementation of on-line motion generator for human figures. *Proceedings of IEEE International Conference on Robotics and Automation*, 2000; 688–694.
17. Kagami S, Kanehiro F, Tamiya Y, Inaba M, Inoue H. Autobalancer: an online dynamic balance compensation scheme for humanoid robots. In *Algorithmic and Computational Robotics: Proceedings of the Fourth Workshop on Algorithmic Foundations of Robotics*, DonaldK BR, Lynch K, Rus D (eds), 2001; 329–340.
18. Shin HJ, Kovar L, Gleicher M. Physical touch-up of human motions. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, 2003; 194–203.
19. Liu CK, Hertzmann A, Popović Z. Composition of complex optimal multi-character motions. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006; 215–222.
20. Lee J, Lee KH. Precomputing avatar behavior from human motion data. *Proceedings of 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; 79–87.
21. Treuille A, Lee Y, Popovic' Z. Near-optimal character animation with continuous control. *ACM Transactions on Graphics* 2007; **26**(3): 7:1–7:7.
22. Shum HPH, Komura T, Yamazaki S. Simulating competitive interactions using singly captured motions. *Proceedings of ACM Virtual Reality Software Technology 2007*, 2007; 65–72.
23. Shum HPH, Komura T, Yamazaki S. Simulating interactions of avatars in high dimensional state space. *ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D) 2008*, 2008; 131–138.
24. Ho ESL, Komura T. Wrestle alone: creating tangled motions of multiple avatars from individually captured motions. In *Proceedings of Pacific Graphics 2007*, 2007; 427–430.
25. Ho ESL, Komura T. Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics* 2009; **15**(3): 481–492.
26. Pohl WF. The self-linking number of a closed space curve. *Journal of Mathematics and Mechanics* 1968; **17**: 975–985.
27. ILOG Inc. *ILOG CPLEX 9.1 User's and Reference Manual*, 2005.

AUTHORS' BIOGRAPHIES

Edmond S. Ho received the BSc degree (first-class honors) in Computer Science in 2003 from the Hong Kong Baptist University, and the MPhil degree in 2006 from the City University of Hong Kong. He is currently a PhD student in the School of Informatics, University of Edinburgh. His research interests include physically based animation, and human motions analysis and synthesis.

Taku Komura is currently a Lecturer in the School of Informatics at University of Edinburgh. He received his PhD (2000), MSc (1997), and BSc (1995) in Information Science from the University of Tokyo. His research interests include human motion analysis and synthesis, physically based animation, and real-time computer graphics.