Simulating Interactions of Characters

Taku Komura, Hubert P.H. Shum, and Edmond S.L. Ho

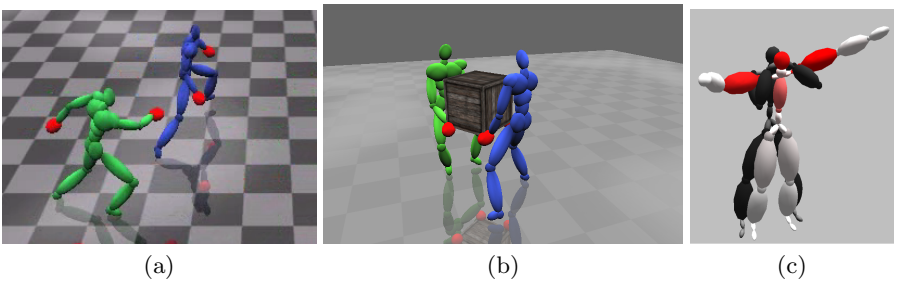
Institute of Perception, Action and Behaviour
School of Informatics, University of Edinburgh

Abstract. It is difficult to create scenes where multiple characters densely interact with each other. Manually creating the motions of characters is time consuming due to the correlation of the movements between the characters. Capturing the motions of multiple characters is also difficult as it requires a huge amount of post-processing of the data. In this paper, we explain the methods we have proposed to simulate close interactions of characters based on singly captured motions. We propose methods to (1) control characters intelligently to cooperatively / competitively interact with the other characters, and (2) generate movements that include close interactions such as tangling the segments with the others by taking into account the topological relationship of the characters.

Keywords: character animation, motion capture, crowd simulation.

1 Introduction

Scenes that multiple characters densely interact with each other are common in daily life. Such scenes include multiple people carrying luggage together, one person holding the shoulder of another injured person and helping him/her walk, a group of people fighting or playing sports such as wrestling, rugby, or ice hockey, and people dancing in a densely crowded hall sticking their body together. Controlling each character under such environment is difficult as the motion of one character affects the motions of all the people in the scene.



(a) (b) (c)

Fig. 1. The close interactions of human characters that we cover in this paper: (a) A character chasing another character taking into account the benefits in the future, (b) two characters carrying luggage together, and (c) two characters wrestling

A. Egges, A. Kamphuis, and M. Overmars (Eds.): MIG 2008, LNCS 5277, pp. 94–103, 2008.

Currently, the cost and time required to create such scenes are enormous. Such motions need to be either created manually or using the motion capture system.

Manually creating a scene of multiple characters is time consuming as each movement of the characters is correlated with those of the others. One amendment results in a number of updates in the scene. Suppose an animator is editing a scene where one character is knocking down another character. If we need to edit the punching motion of the attacker, by changing the position and timing of the punch landing onto the opponent, then we also need to edit the motion of the opponent being knocked down, by changing the way it gets hit and falls down onto the ground. If the two characters are repeatedly attacking / defensing, the amount of work becomes enormous.

Capturing the motions of multiple persons in the scene at the same time using a motion capture system is another solution to create such an animation; however, for scenes as fighting, this is difficult due to the intrusiveness of the motion capture system, occlusions, and the intensive interactions that affect the performance of capturing. Think of using an optical motion capture system to capture two boxers seriously sparring. In the beginning, they actually never seriously hit each other as markers are attached to various parts of their bodies. Athletes are sensitive to such an unusual condition and they cannot perform in the way they usually do. Although it is difficult, suppose the boxers overcome such pressure and start to spar at close distance seriously. A lot of markers are occluded by the arms, head and torso of each boxer as they are hitting each other in a very close distance. And finally it will be found out capturing serious intense sparring is almost impossible as the markers are flying away from their bodies when one fighter's arm hits/rubs the surface of the other's body. The situation will be similar or even worse when magnetic / mechanical motion capture systems are used, as they are even more intrusive than the optical system.

Therefore, we take a more practical approach; we capture the motion data individually, and simulate the interactions by controlling the characters using AI techniques. In that case, the problems of occlusions and post-processing become much easier to handle. We can also produce combinations of actions which are difficult to be captured due to safety. In this paper, we introduce our previous work to generate animation of multiple characters closely interacting with each other. We simulate competitive interactions such as chasing, fighting and playing sports, as well as cooperative interactions such as carrying luggage together. Three methodologies are presented:

1. **Simulating Competitive Interactions using Singly Captured Motions (Section 2).** First we explain a method to generate a realistic scene of characters interacting in a competitive environment with a method similar to controlling computer-based players in chess. We expand the game tree and evaluate the interactions of characters taking into account the rewards in the future. This method is effective to simulate scenes such as fighting and chasing.
2. **Simulating Interactions of Avatars in High Dimensional State Space (Section 3).** Although the above approach is effective to simulate

realistic interactions between the characters, it is difficult to apply it to real-time applications such as 3D computer games. This is because game-tree expansion requires exponential computational time. Here we explain a method to create and utilize a finite state machine which is effective to control computer-controlled characters in real-time.

3. **Creating tangled motions (Section 4).** In the previous approaches, we mainly focused on instantaneous and impulsive interactions. In some cases, we need to handle interactions where a character tangles its body segments to those of the others, such as when piggy-backing another person, giving shoulder to an injured person to walk, or playing wrestling. Here we explain a method to create such motions by taking into account the topological relationships of the characters.

2 Simulating Competitive Interactions Using Singly Captured Motions

In this section, we explain the overview of our method to simulate interactions of characters by expanding the game-tree and evaluating the status of the characters in the future. For the details of the techniques, the reader is referred to [8]. By using this method, the characters make intelligent choices taking into account their rewards in the future. As a result, the interactions between the characters appear more realistic than when using emergent approaches such as flocking.

The outline of the method is shown in Figure 2. We first capture the motions of subjects individually using optical motion capture systems (Figure 2, left-most). The captured motions are segmented and classified into semantic groups such as "straight punch", "kick" or "parry" automatically. Using the annotated motions, we generate a data-structure called action-level motion graph (Figure 2, left middle), which is a motion graph structure whose actions are annotated. In order to control the characters in an intelligent way, it is necessary to make the character predict how the opponent will react to its action, and decide the next action based on its benefits in the future. In order to do this, we expand the game tree, and assess the status in the future using an evaluation function (Figure 2, right middle). Finally, the character selects an action that maximizes its rewards in the future, and launches its action (Figure 2, right most).

We propose a new algorithm called the temporal expansion approach which maps the continuous action planning to a discrete space so that turn-based evaluation methods such as min-max algorithms and $\alpha - \beta$ pruning can be used.

In order to simulate realistic interactions of the characters, we propose to use a table that pairs actions. In this table, the appropriate actions that need to be launched when the opponent character is undergoing some specific actions are listed. For example, for each entry of the attack, the appropriate defense motions together with the best timing to launch them are listed.

The users can easily specify how the scene should appear by tuning parameters. Every character is guided by an objective function, and it is possible to

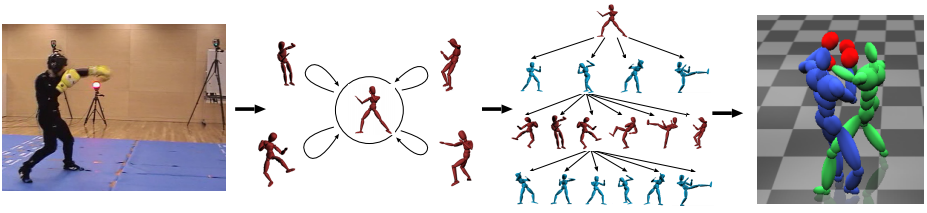
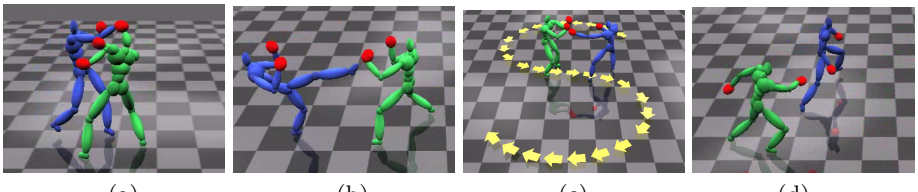


Fig. 2. The outline of the proposed method to simulate competitive interactions: (1) capture the motions of characters individually (2) generate the action level motion graph (3) evaluate the interaction by expanding the game tree (4) simulate the competition by physically-based animation

set up a scenario of the competition, or control the way the character competes by tuning the parameters of its objective function. For example, in case of fighting, it is possible to simulate various fighting styles, such as being more passive, aggressive, or preferring kicks than punches by changing the scores given to the characters when they successfully attack or defend. By giving higher scores to both characters when they follow a path while fighting, both the characters will tend to do so.

2.1 Experimental Results

We have simulated various competitive interactions of the characters to show the effectiveness of our method. We have created examples of boxing matches. The strength of each fighter can be adjusted by changing the depth of the game tree expanded. We can also adjust the parameters of the characters to simulate different styles of fights, including outboxing and infighting (Figure 3 (a),(b)).



(a) (b) (c) (d)

Fig. 3. Some of the screen shots of the simulated fights: (a) Infighters fighting at very close distance, (b) outboxers at long-range distance, (c) the fighters following a path while fighting, and (d) one avatar chasing another

Next, a scene two fighters moving along a predefined path while fighting was simulated (Figure 3 (c)). The path is modeled as a series of check points.

Finally, a scene where an avatar chases another was simulated (Figure 3 (d)). The movements of both avatars are based on the running-around motion. The preferred distance of the chaser is set short and that of the avatar who is running away long. Moreover, based on the scoring function, high score is given to the

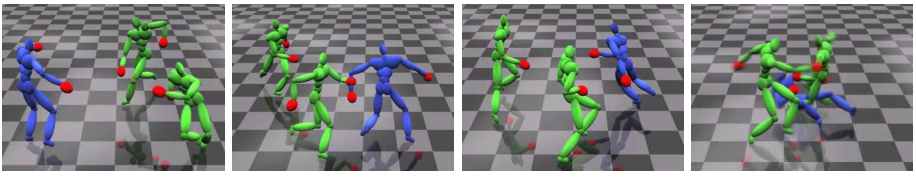


Fig. 4. Two green avatars chasing the blue avatar. The green avatars cooperate with each other to catch the blue avatar.

chaser when it catches the other avatar. As a result, the chaser tries to approach its opponent while the opponent tries to get away. We also simulated a scene where two avatars chase one avatar. In this case, the game tree is composed of nodes and edges which represent the actions of three avatars. The chasers cooperate with each other to catch the avatar who is running away (Figure 4).

3 Simulating Interactions of Avatars in High Dimensional State Space

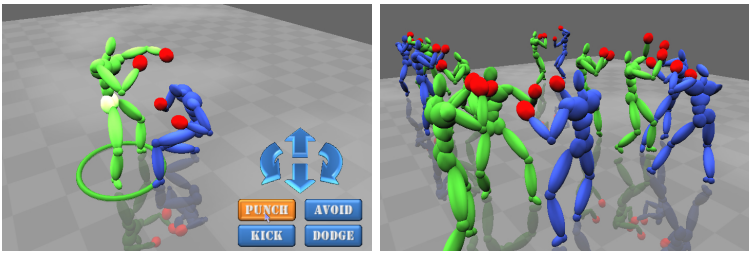
In this section, we explain our method to control non-player-characters (NPCs) of 3D computer games to intelligently interact with human-controlled characters in real-time. For the details of the techniques, the reader is referred to [7].

The method based on game-tree expansion explained in the previous section requires exponential computational cost. As a result, it is difficult that method for controlling NPCs in 3D computer games.

Reinforcement learning enables real-time optimal control of characters. It has been used to control pedestrians to avoid other obstacles/characters walking in the streets [3,10], control a boxer to approach and hit the target [4], make the transition of actions by the user-controlled character smooth [5] and train a computer-controlled fighter in computer games [1]. However, there are two problems that we face when we try to use reinforcement learning to control human characters intelligently when they are interacting with another character.

First of all, the state space is too large. The state space increases exponentially proportional to the number of parameters. Parameters such as the action the character is undertaking, its body position and orientation, and the timing to launch the action are going to form the state space. The number of parameters is going to double if there are two characters. As a result, it is difficult for existing adaptive learning techniques such as Q-learning [11] to explore the whole state space to search for optimal policies.

Another problem is that the way the people behave change according to various factors such as their mood, habits, and preferences of actions; however, previous animation techniques used "on-policy" [9] reinforcement learning methods, which require the system to be retrained in case the reward function is changed. For example, in boxing, there are boxers called infighters who prefer to fight aggressively in short distance, and use punches such as upper cuts and hooks more. On the contrary, there are outboxers, who prefer to stay away from the



(a) (b)

Fig. 5. The interactions of articulated characters are generated by maximizing the reward function defined by the relative pose between characters, the effectiveness of actions, and/or user-defined constraints. This framework of synthesizing character animation is efficient and flexible enough to make a variety of practical applications including (a) interactive character control using high-level motion descriptions such as punches, kicks, avoids and dodges and (b) real-time massive character interactions by a large number of automated characters

opponent and as a result, prefer to use straight punches which are effective in long distance. If we train a virtual boxer by an on-policy reinforcement learning approach, it will not be able to compete well with other fighters who have different styles of fighting. The system needs to be pre-trained for various types of fighters, and the policy needs to be switched according to the type of the opponent, which will be very computationally costly.

Here we make use of the fact that the subspace of meaningful interactions is much smaller than the whole state space of two characters. We efficiently collect samples by exploring the subspace where dense interactions of the characters exist and favoring samples which have high connectivity with other samples. Using the samples collected, a finite state machine (FSM) called Interaction Graph is composed. The Interaction Graph is a Motion Graph of two characters. In order to control the character in an optimal way, a min-max search / dynamic programming is conducted on the Interaction Graph.

We can simulate various activities by two characters such as fighting, chasing, playing sports, or carrying luggage together. Our method can plan strategic movements for Non-Player Characters (NPCs) in 3D computer games. For example, we can control virtual fighters in boxing games (Figure 5(a)), or the background crowd moving or fighting with each other in computer animations (Figure 5(b)), or characters collaboratively working, such as carrying a box (Figure 1(b)).

3.1 Outline of the Method

The procedure of our method can be divided into the preprocessing stage and run-time stage. The **preprocessing stage** proceeds as follows:

1. Capture the motions of a single person conducting the target motion and generate the action level motion graph structure out of the motion data

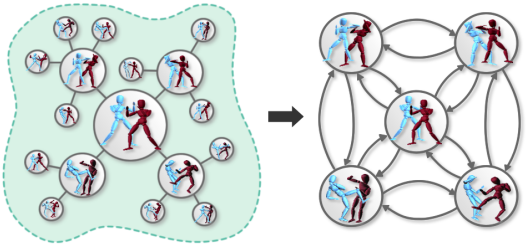


Fig. 6. The outline of the preprocessing stage: (left) exploring the state space by favoring states with more interactions and transitions to existing samples (right) an Interaction Graph created from the collected samples

2. Explore the combined state space of two characters by simulating the interactions of the two characters and expanding the motion tree (Figure 6 left)
3. Generate the Interaction Graph of the two characters and find the most appropriate movements of the characters at each node by dynamic programming or min-max search. (Figure 6 right)

Then during **run-time**:

1. At each state, the corresponding character selects the precomputed optimal action
2. If the animator/user wants to change the policy/strategy of the control, the information in the lookup-table is recomputed by re-running dynamic programming or min-max search. This can be done in a few seconds, and can be run in background while simulating the interactions

3.2 Experimental Results

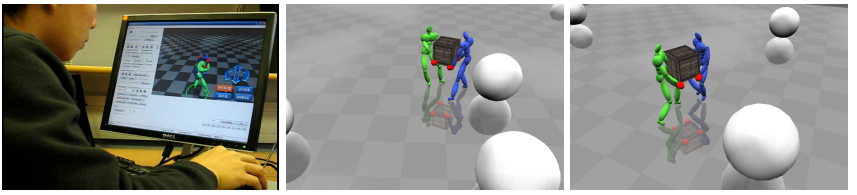
We have simulated scenes of fighting as examples of competitive interactions and scenes of carrying luggage together as examples of collaborative interactions.

In order to show the real-time performance of our system, we have implemented a game-style interface which the user can control an avatar to fight with the computer-controlled avatar (Figure 7 (a)).

For the example of carrying luggage, another interface to move the avatars to arbitrary directions to avoid being hit by the ball was implemented. Screen shots of this example are shown in Figure 7 (b),(c).

4 Creating Tangled Motions

In this section, we explain our method to simulate close interactions that requires human characters to tangle its limbs with those of the others. We propose a method to use Gauss Integrals (GI), which is a concept proposed in knot theory. For the details of the techniques, the reader is referred to [2].



(a) (b) (c)

Fig. 7. (a) Using the game style interface, the user can control an avatar to fight with the computer-controlled avatar by the Interaction Graph. (b),(c) Screen shots of the avatars controlled to avoid the ball while holding a box.

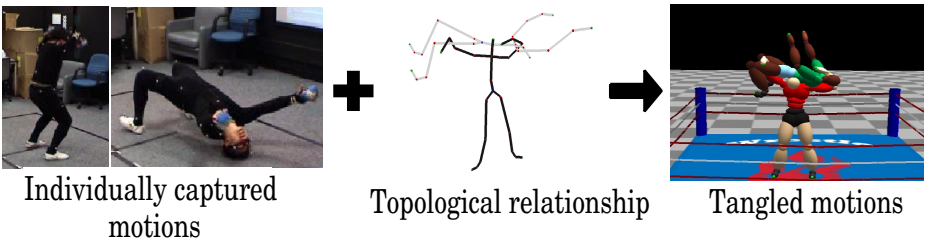


Fig. 8. The outline of the proposed method to simulate interactions of characters that involve tangling

Animations of two characters tangled with each other often appear in battle or fighting scenes in films or games. However, creating such scenes is difficult due to the limitations of the tracking devices and the complex interactions of the characters during such motions. Here we propose a new method to generate animations of two persons tangled with each other based on individually captured motions. We use wrestling as an example.

We propose to use GI to calculate the tangled status of bodies. Since GI can only calculate the relationship of two strands, we propose a method to apply it to express the tangled status of multibody structures such as humans. Once the relationship is specified, the motions of the characters are imported and edited automatically, so that constraints due to penetration or geometrical constraints such as keeping the support feet onto the ground are satisfied. The tangle relationships are also monitored so that the segments do not get untangled. As a result, a scene of two characters interacting with each other can be generated.

Our method can be used to create motions such as holds and chokes in wrestling, a helper holding a shoulder of an injured person to walk or a person piggy-backing another person. The motions created using this method can be used for applications such as computer games and 3D computer animation.

4.1 Methodology

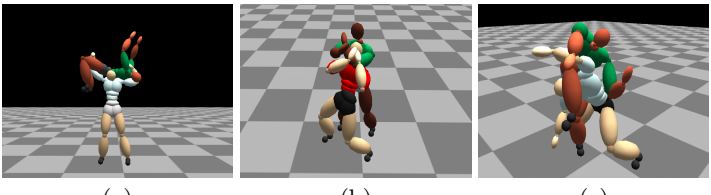
The overview of our methodology is as follows:

1. The user captures the motions of the two characters individually using a motion capture system
2. The user specifies the topological relationship of the characters by composing a template posture with our 3D character posing interface. The postures are examined by the system and the segments composing the tangles are detected by calculating the GLI of the segments.
3. The motion data of both characters are edited according to the topological relationship specified in Step 2.

The flowchart of the algorithm is shown in Figure 8.

4.2 Experimental Results

We have simulated a number of wrestling motions including the Argentine Back-Breaker (Figure 9 (a)), the Rear-Chokehold (Figure 9 (b)), and the Octopus Hold (Figure 9 (c)).



(a) (b) (c)

Fig. 9. (a) Argentine Back Breaker, (b) The Rear-Chokehold and (c) the Octopus Hold simulated using our method

5 Summary and Future Work

In this paper, we introduced methods that we have previously proposed to simulate the close interactions of multiple characters. We have covered interactions such as chasing, fighting, carrying luggage, and wrestling. We believe the following topics are the important areas to further explore:

Simulating scenes where a large number of characters interact, such as one person fighting with many background characters, characters fall onto others like domino in panic, and multiple characters pass luggage to the person standing next to it.

Creating a motion graph based on topology: In motion graphs, usually the Euclidean distances of the state vectors based on joint angles or the joint positions

are used to evaluate the similarity of different postures. Such kind of distance measures can cause troubles when we animate motions such as wrestling. We can make use of the topological relationship of the bodies in evaluating the similarity of postures of two characters, and compose a motion graph based on this distance measure. It is expected that less penetration of the segments will occur.

Parameterizing the interactions: Currently, we select the actions from a large set of motions. This increases the state space as there can be a set of similar motions which are have the same effect to the scene. We can parameterize the actions [6] from a small set of actions by using interpolation and produce various interactions out of them.

References

1. Graepel, T., Herbrich, R., Gold, J.: Learning to fight. In: Proceedings of Computer Games: Artificial Intelligence Design and Education (CGAIDE 2004), pp. 193–200 (2004)
2. Ho, E.S.L., Komur, T.: Wrestle alone: Creating tangled motions of multiple avatars from individually captured motions. In: Proceedings of Pacific Graphics 2007, pp. 427–430 (2007)
3. Ikemoto, L., Arikan, O., Forsyth, D.: Learning to move autonomously in a hostile world. Technical Report No. UCB/CSD-5-1395, University of California, Berkeley (2005)
4. Lee, J., Lee, K.H.: Precomputing avatar behavior from human motion data. In: Proceedings of 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 79–87 (2004)
5. McCann, J., Pollard, N.S.: Responsive characters from motion fragments. ACM Transactions on Graphics (SIGGRAPH 2007) 26(3) (August 2007)
6. Mukai, T., Kuriyama, S.: Geostatistical motion interpolation. ACM Trans. Graph. 24(3), 1062–1070 (2005)
7. Shum, H.P.H., Komura, T., Yamazaki, S.: Simulating interactions of avatars in high dimensional state space. In: ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D) 2008 (2008)
8. Shum, H.P.H., Komura, T., Yamazaki, S.: Simulating competitive interactions using singly captured motions. In: Proceeedings of ACM Virtual Reality Software Technology 2007 (2007)
9. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
10. Treuille, A., Lee, Y., Popovic, Z.: Near-optimal character animation with continuous control. ACM Transactions on Graphics 26(3) (2007)
11. Watkins, C.: Learning from Delayed Rewards. PhD thesis, Cambridge University (1989)