Simulating Competitive Interactions using Singly Captured Motions

Hubert P.H. Shum* Taku Komura† Shuntaro Yamazaki‡
Institute of Perception, Action and Behavior, School of Informatics, University of Edinburgh
Digital Human Research Center, AIST, Japan

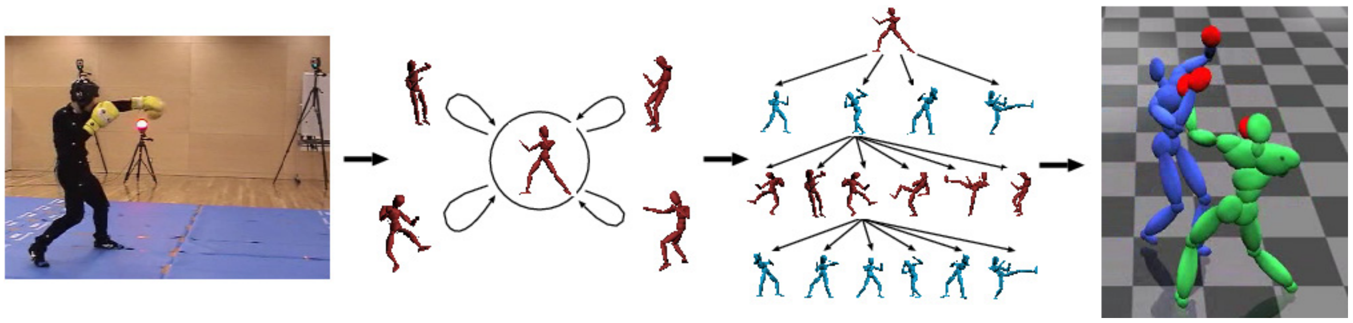

Figure 1: *The outline of the proposed method to simulate competitive interactions: (left most) capture the motions of avatars individually (left middle) generate the action level motion graph (right middle) evaluate the interaction by expanding the game tree (right most) simulate the competition by physically-based animation*

Abstract

It is difficult to create scenes where multiple avatars are fighting / competing with each other. Manually creating the motions of avatars is time consuming due to the correlation of the movements between the avatars. Capturing the motions of multiple avatars is also difficult as it requires a huge amount of post-processing. In this paper, we propose a new method to generate a realistic scene of avatars densely interacting in a competitive environment. The motions of the avatars are considered to be captured individually, which will increase the easiness of obtaining the data. We propose a new algorithm called the temporal expansion approach which maps the continuous time action plan to a discrete space such that turn-based evaluation methods can be used. As a result, many mature algorithms in game such as the min-max search and $\alpha - \beta$ pruning can be applied. Using our method, avatars will plan their strategies taking into account the reaction of the opponent. Fighting scenes with multiple avatars are generated to demonstrate the effectiveness of our algorithm. The proposed method can also be applied to other kinds of continuous activities that require strategy planning such as sport games.

CR Categories: I.3.6 [Methodology and Techniques]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Human Simulation, Motion Planning, Motion Capture

1 Introduction

Scenes where multiple characters densely interact with each other frequently appear in TV programs, movies and 3D computer games. Currently, the cost and time required to create such scenes are enormous. Such motions need to be created either manually or by using the motion capture system.

Manually creating a scene of multiple avatars is time consuming as each movement of the avatar is correlated with those of the others. Even a small modification to the motion of a single avator results in a number of updates in the scene. Suppose an animator is editing a scene one avatar is knocking down another avatar. If the avatar needs to edit the punching motion of the attacker, say, changing the position and timing of the punch landing onto the opponent. The animator then also needs 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 avatars are repeatedly attacking / defensing, the total amount of work becomes enormous.

Capturing the motions of multiple persons in the scene together 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 will avoid performing in the way they usually do. Although it is difficult, suppose the boxers overcome such pressure and start to spar at close distance seriously. Now 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 fighter'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.

In this paper, a more practical approach is proposed; we capture the motion data individually, and simulate the competitive interactions by AI techniques used to control computer-based players in

*e-mail: hubert.shum@ed.ac.uk
†e-mail: tkomura@inf.ed.ac.uk
‡e-mail: shun-yamazaki@aist.go.jp

strategy games such as chess. After the motions are captured, they are segmented and classified into semantic groups such as “straight punch”, “kick” or “parry” automatically. In order to control the characters in an intelligent way, it is necessary to make the avatar predict how the opponent will react to its action, and decide the next action based on its benefits in the future. 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. The outline of the method is shown in Figure 1.

In order to simulate realistic interactions of the avatars, we propose to use a table that pairs actions [Lin et al. 2005]. In this table, the appropriate actions that need to be launched when the opponent avatar 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. If the animator wants to associate a certain attack with a certain defense, he/she can add such entries into the table.

The users can easily specify how the scene should appear by tuning parameters. Every avatar is guided by an objective function, and it is possible to set up a scenario of the competition, or control the way the avatar 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 avatars when they successfully attack or defend. By giving higher scores to both avatars when they follow a path while fighting, both the avatars will tend to do so.

We have simulated various competitive interactions of the avatars 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 also adjust the parameters of the avatars to simulate different styles of fights, including outboxing and infighting. Our method can also be used for other competitive scenes such as chasing or playing sports.

2 Related Work

Motion editing and synthesis has become a huge research area which has many applications in computer graphics, robotics and biomechanics. Recently, a lot of data-driven techniques to edit / retarget [Gleicher 1998; Lee and Shin 1999; Abe et al. 2004] or synthesize a new sequence of character’s motion using precaptured motion data [Arikan and Forsyth 2002; Lee et al. 2002; Kovar et al. 2002; Kovar and Gleicher 2004; Mukai and Kuriyama 2005; Safonova and Hodgins 2007] are proposed. The Motion Graph approach [Arikan and Forsyth 2002; Lee et al. 2002; Kovar et al. 2002] is a method to interactively reproduce continuous motions of characters based on a graph that is automatically generated from captured motion data. Since the Motion Graph produces a lot of edges and nodes without any context, it becomes difficult to control the character as the user wishes. Recently, therefore, works to resolve such problems by introducing a hierarchical structure are proposed [Lau and Kuffner 2005; Kwon and Shin 2005].

Most of these researches handle characters in the scene individually, and need extensions to handle dense interactions, such as pushing or pulling, among several characters. For such kind of effects, methods that combine simulations and motion capture data are known to be effective [Arikan et al. 2005; Zordan et al. 2005; Komura et al. 2005b; Komura et al. 2005a]. In case of fighting scenes, however, the offended characters must not just be pushed away, but need to defend and counterattack.

Although there is a lot of research work for motion editing and synthesis, less work has been done on simulating scenes in which more than two characters continuously interact with each other.

Liu et al. [Liu et al. 2006] creates such scenes by alternately computing the motions of individual character by using spacetime constraints. This method is effective for creating animation that includes sustained constraints such as a parent pulling the hand of a child. However, for events that have many interactions such as fighting, this method cannot be used.

Lau et al. [Lau and Kuffner 2006] precomputes the optimal motions to move to locations around the avatar based on a finite state machine; they simulated the motions of avoiding each other when running or walking in an open space. The interactions between avatars that can be simulated are limited by such an approach and they cannot handle simulations of dense interactions such as fights.

Lee et al. [Lee and Lee 2004] simulates a scene of two boxers fighting with each other by using a precomputing approach. The boxers are trained by reinforcement learning so that they know the optimal way to approach and hit the target. The boxers are trained alone to find the optimal motions to approach and make a hit. There is no concept of continuous interactions during the training stage, although this is one of the most important issues for activities of multiple avatars. For example, there is no context for defense motions, and therefore, the fighters will only try to approach and hit without taking into account the opponent’s action. Graepel et al. [Graepel et al. 2004] also uses reinforcement learning to train the computer-based player of fighting games. The system observes how the players fight with the computer-based players and learns the optimal policy to fight. The system requires many hours of training to learn the optimal policy, and such policy needs to be trained again when the style of the fight is changed.

Park et al.[Park et al. 2004] synthesizes a scene of avatars dancing and playing Taekwondo based on captured data. They capture the motions of multiple persons and generate a Hidden Markov Model based on the statistics of the motions. Since the motions of several people have to be captured together, the difficulty of capturing the motions of dense interaction limits the availability of the data, and as a result, the interaction that can be synthesized is limited in such an approach. In this research, since we simulate the interactions of the characters, we do not have limitations due to the capturing.

3 Data Acquisition and Analysis

Here we explain the process that we capture the motions of actors individually, segment them into shorter semantic actions, classify them into different categories, and finally compose a data structure called action-level motion graph.

Firstly we capture the motions of boxers and kickboxers shadow boxing alone. Here we define the term “motion” as the raw-captured data, and the term “action” as a semantic segment of the motion we captured. In the field of fighting, an action can be an attack (such as a “left straight”, “jab” or a “right kick”), a defense (such as “parries”, “blocking” or “ducking”) , a transition (such as “stepping to the left”, “stepping forward” or “back step”), or reactive motions when hit / pushed away, or the combination of these.

We have developed an automatic motion analyzer to segment and classify raw motions into actions. This is done by first partitioning a long motion sequence into segments at the center of double support phase. However, in case the sum of squares of the acceleration of the joints is large, we do not segment the motion at that moment as it can be expected that the body is going through a continuous action. Finally, we classify the actions according to the trajectories of the joints with large acceleration.

We build a Motion Graph [Arikan and Forsyth 2002; Lee et al. 2002; Kovar et al. 2002] in the action level rather than the frame level, as in [Gleicher et al. 2003; Lau and Kuffner 2005; Kwon and Shin 2005]. This is done by extracting the starting poses and the ending poses of the actions and grouping similar poses together. Let us call this data structure the action-level Motion Graph. Figure 2 illustrates the action level Motion Graph. Planning based on the action level Motion Graph is similar to the human way of thinking, as people also use attack/defense/transition actions as the building blocks when fighting.

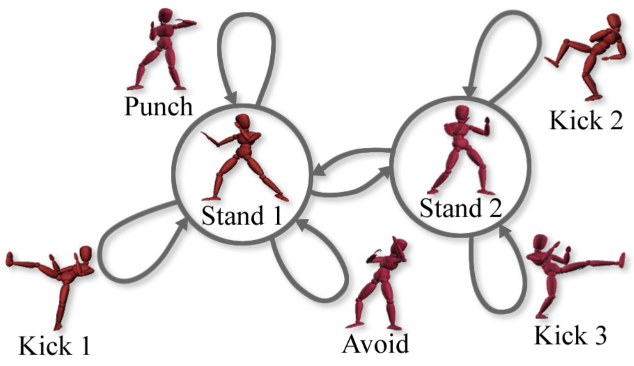

Figure 2: *An action level motion graph that is generated from the boxing motion.*

4 Evaluating the Actions

In order to evaluate the actions, we propose an objective function that can be used for various competitive interactions including fighting, chasing and sports. We evaluate the following three factors: (1) its relative position with the opponent or global position / orientation (**location function**), (2) the immediate gain and loss caused by the launch of that action (**scoring function**), and (3) the quality of the action from the viewer's point of view, such as the duration of the action and the frequency of usage (**control function**). The objective function can change according to the individuality of the avatar or the style of fighting.

The **location function** is designed so that the avatar's relative distance / facing angle with the opponent is within the preferred range, and its global location / orientation is following the scenario given by the user. In boxing, an infighter prefers to keep short distance with the opponent. In that case, higher scores are given to actions that bring the avatar closer to its enemy. On the contrary, for an outboxer or a passive fighter who prefers to escape from the opponent, high scores are given to actions that increase the distance between them. Regarding the orientation, for competitive environments, it is always preferable that the avatar faces the opponent. Therefore, we evaluate the action using the direction of the head at the last frame of the action. Finally, sometimes the user prefers the avatars to follow some paths based on the scenario of scene. We can add such terms into the location function so that the avatar is given higher scores for actions that bring it to the desired location. The location function can be written as

$$F^{loc} = w_\theta \theta^2 + w_r(r - r_d)^2 + w_{orient}(\theta_o - \theta_d)^2 + w_l(p - p_d)^2 \quad (1)$$

where θ, r are the relative orientation and distance from the opponent, respectively, r_d is the preferred distance of the avatar from the opponent, θ_o is the orientation of the avatar around the vertical axis in the world coordinate system, θ_d is its desired value, p is the location in the world coordinate system, p_d is its desired value, and $w_\theta, w_r, w_{orient}, w_l$ are the weight constants for each term.

The **scoring function** evaluates how effective the action is to compete with the opponent. It is the weighted sum of the damage the avatar gives to the opponent and that the avatar receives from the opponent by selecting that action. It can be written as

$$F^{score} = w_D^+ D^+ - w_D^- D^- \quad (2)$$

where D^+ is the damage that the fighter gives the opponent, D^- is the damage received, and w_D^+, w_D^- are positive weight constants for each term. The weight constants are changed according to the competing style of the avatar. D^+ and D^- are proportional to the velocity of the attacking segment at the moment it is landing to the opponent. For boxing, in case the fighter is an outboxer who is less aggressive, w_D^+ is set smaller and w_D^- is set larger. In case a fighter is running out of time and is losing the fight, it has to fight more aggressively regardless of the risk of being hit; in that case, w_D^+ is increased and w_D^- is decreased.

The **control function** ($F^{control}$) can be designed according to the application. If the designer prefers some specific good looking motions, they can favor those by giving higher marks to them when they are successful. We have tuned it to favor short and seldom used actions, as shorter motions will risk the body less, and seldom used actions will enhance the visual effect.

The outputs of the three criteria are combined together by calculating their sum. As a result, the objective function J can be written in the following form:

$$J = F^{loc} + F^{score} + F^{control}. \quad (3)$$

This objective function can be used for various competitive interactions such as fighting, chasing, and sports. For simulating interactions such as chasing, we can increase the preferred distance for the avatar running away, and shorten it for the chaser. For other sports, for example basketball, we can set up a scoring function that lets the avatar shoot more when the probability to get a score is higher (for example, when there is no opponent in front of it), a location function that guides the avatar to the sweet spot, and a control function that makes the avatar select the action that appears more impressive.

5 Temporal Expansion Approach

In this section, the method called temporal expansion approach is explained. This is a method to control the avatars intelligently by expanding the game tree. Once each avatar's motion data are prepared, their interactions are simulated. Using this method, we can (1) simulate different level of intelligence by changing the depth of the expansion, and (2) balance the computational load and the intelligence requirement.

5.1 Game Tree Expansion

For controlling the avatars, we adopt methods used for AI players in strategy games such as chess. If we want to control them intelligently, only considering the immediate benefit is not enough. For example, in chess, a movement that shows the greatest effect in one ply (such as taking a valuable piece like a castle or a bishop) is not necessarily the best choice for winning at the end. AI algorithms expand the game tree and evaluate the static position after a few plies, to make a choice that benefits the player in long term. Here we apply the same approach. The difference between chess and fighting is that the choices made by the players are not alternate, and they depend on the duration of the action done by each player. An example of such a game tree is shown in Figure 3.

In this game tree, we assume that the depth along the vertical axis represents the time passed. The red and blue circles represent the

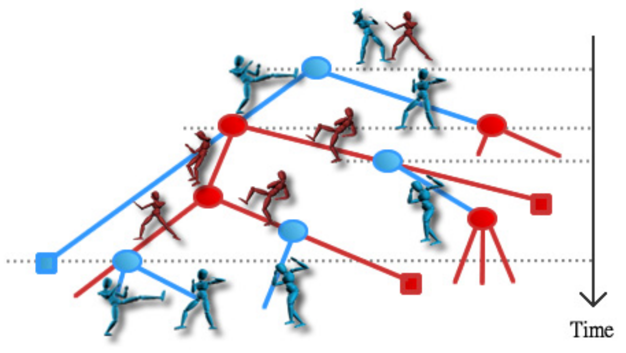

Figure 3: *An expanded game tree of fighting. The distance along the vertical axis represents time. The white and black circle nodes represent the moments fighter A and B launch new actions, respectively. Each edge represents the action that has been selected by the fighter and the square represents the end of each action*

moment fighter A and B launch new actions, respectively. Each edge represents the action that has been selected by the fighter, and the square represents the moment that the action ends. The square is omitted and replaced with a circle node in case a successive motion is started by the same fighter.

After an avatar launches an action, if this action is short enough, the avatar might be able to launch another action before the opponent ends the current action. In such a case, the actions are not going to be alternate, but two series of actions are consecutively launched by the same avatar.

In some cases, the fighter's action might be interrupted by the opponent by being punched or kicked. In such a case, the fighter might be either knocked down onto the ground immediately, or just lose balance and walk a few steps to recover the balance and resume the fight. In either case, the response motion will be decided based on the current state of the body and the impulse added to the body. Since we assume the avatar being hit does not have a choice for the action, there is only one edge going out from the node when such response motion is launched.

When expanding the game tree, it is necessary to specify the time up to when we make the prediction. Here we limit by the number of plies. Once we reach this limit, we stop expanding the tree and evaluate the sequence of motions to proceed to the min-max algorithm.

5.2 Min-Max Selection

Once the game tree is expanded, we use the min-max algorithm to select the best motion that maximizes the avatar's chance to win. The min-max algorithm gives the optimal move for zero-sum games such as tic-tac-toe, chess and go, and is also suitable for competitions in this research.

When using min-max algorithm, we need to define a zero-sum static function that evaluates the leaf nodes of the game tree. In order to define such zero-sum function for the leaf nodes, we first need to evaluate every transition from one node to another in a zero-sum manner according to Equation 3. Say the positive score of the evaluation function represents the benefits of fighter A and the negative score represents the benefits of fighter B. Then, a function f_z that evaluates the transition can be defined by

$$f_z(e) = J_A^e - J_B^e. \tag{4}$$

where J_A^e is the result of evaluating transition e by Equation 3 from avatar A's point of view, and J_B^e is that from avatar B's point of view. Say we want to evaluate a leaf node L_n of the game tree, which can be reached from the root by descending a route defined by l (Figure 4, left). The zero-sum evaluation of the leaf node is then defined as

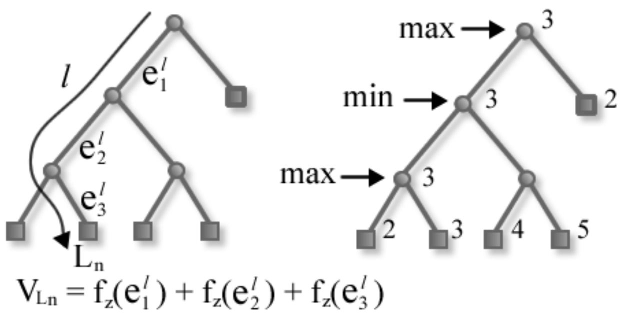

Figure 4: *Evaluation of the end node is done by summing the score of all the edges from the root to the leaf node. Then the score of the internal nodes are computed by using min-max method from the leaf nodes toward the root.*

follows:

$$V_{L_n} = \sum_i f_z(e_i^l) \tag{5}$$

where e_i^l is the i-th edge from the root when descending l.

Now, we recursively evaluate the internal nodes of the game tree starting from the leaf nodes and moving up towards the root node using the min-max algorithm (Figure 4, right). Suppose we are to evaluate the score of an internal node which is m levels deep from the root node, and we know the scores of all the nodes at level $m+1$. Let us represent this node by $\mathbf{N^m}$, and its children by $\mathbf{N_1^{m+1}}, \ldots, \mathbf{N_k^{m+1}}$, and their scores by $S(\mathbf{N^m})$ and $S(\mathbf{N_1^{m+1}}), \ldots, S(\mathbf{N_k^{m+1}})$, respectively. We can compute $S(\mathbf{N^m})$ using $S(\mathbf{N_1^{m+1}}), \ldots, S(\mathbf{N_k^{m+1}})$ as follows:

$$S(\mathbf{N^m}) = \begin{cases} max\{S(\mathbf{N_1^{m+1}}), \ldots, S(\mathbf{N_k^{m+1}})\} & \text{if } \mathbf{N^m} \text{ is A's node} \\ min\{S(\mathbf{N_1^{m+1}}), \ldots, S(\mathbf{N_k^{m+1}})\} & \text{if } \mathbf{N^m} \text{ is B's node} \end{cases} \tag{6}$$

As the scores of all the leaf nodes are already computed by Equation 5, we can recursively compute the scores of the internal nodes towards the root. The score of the root node can be calculated by recursively applying Equation 6.

5.3 Pruning Non-Plausible Choices

In order to reduce the computational cost and avoid non-plausible interactions to appear, we prune the mal choices when expanding the game tree. Although there are a huge number of combinations of the actions of the two fighters, many of them never happen as they obviously cause disadvantages to the avatar. Such mal combinations of actions are listed below:

- **Attacks out of distance** : there is no meaning to launch an attack if the opponent is farther away than the reaching distance. (Figure 5(a))
- **Defending when the opponent is not attacking** : in case the opponent is neither attacking nor showing any sign of attack, defense motion must not be launched(Figure 5(b))

- **Actions of penetrating the opponent** : collisions of the fighters are examined and in case they end up brutally overlapping with each other, such pairs of actions are invalid. (Figure 5(c))

These combinations are excluded from consideration unless there is no other action to choose, as the resulting animation will appear unnatural. We take advantage of the above mentioned features to reduce the computational cost of strategy making. According to our experimental results, we can prune half of the available choices in average using these pruning policies. This would reduce the computational cost by $O((\frac{1}{2}m)^n)$, where m is the number of available actions and n is the depth of the of the game tree.

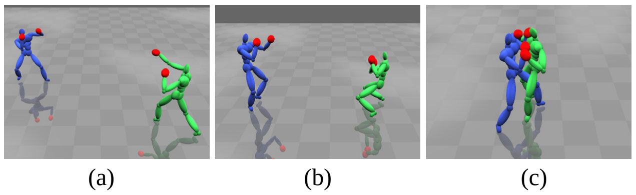

Figure 5: *The combinations which are considered invalid. (a) Attacks out of distance, (b) defending when there is no attack, and (c) the bodies penetrating each other too much*

5.4 Creating the Offense / Defense Table

In this section the offense / defense table [Lin et al. 2005] is explained, which pairs the offense and defense motions to let the avatar effectively launch the appropriate defense motion to counteract the attack by the opponent. This table illustrates the spatiotemporal relationship of attacks and dodges, to incorporate tactical maneuvers of defense into the scene.

The table (1) enables to take into account subtle factors of fights/interactions which cannot be expressed by Equation 3, and (2) provides interface to animators who want to pair specific attacks with defenses.

Regarding (1), it is known in boxing that sway back motion is effective for avoiding upper cuts and hooks, and head slip is good for avoiding straight punches. There are various factors such as the direction the punch is approaching from, and whether the defender can see the attacker all through the motion, that support these basic techniques. Such kind of subtle factors cannot be evaluated by Equation 3. By using the offense/defense table, we will be able to see more effective defense motions to deal with the attacks, as those appearing in real matches.

Regarding (2), by using the offense/defense table, the animator has an interface to embed manually designed plausible close-contact interactions into the scene. By pairing special attacks with special defenses, it is possible to make such interactions appear which might be difficult to show in case the actions are evaluated purely by Equation 3.

In our implementation, the attacks and defenses are initially matched based on two attributes: the height and directions. Every attack and defense is classified into three categories “high”, “middle”, and “low” according to the height it is attacking / defending. They are further classified into categories “left”, “right”, “middle” and “upwards” according to the direction of the attack or to which direction of attacks the defense is valid for. Finally, adjustments are made, according to knowledge of the experts.

6 Physical Interactions Between Avatars

Since there are a lot of collisions between the bodies, we add repulsive forces to the segments when a collision occurs, and the segments are pulled back to the original trajectories by PD control [Zordan and Hodgins 2002]. The bodies are modeled by spheres and cylinders to reduce the computational cost of collision detections.

We adopt Jakobsen’s [Jakobsen 2001] technique to use particle systems to simulate the rigid body dynamics; since the location of segments can be constrained in this method, we fix the supporting foot onto the ground to avoid it from sliding.

When the attacker correctly hits the opponent, a reactive motion to step away is launched to simulate the effect of being hit. According to the posture of the body and the direction and strength of impulse, we simulate the initial reaction based on rigid body dynamics and then blend the motion with the reactive motion selected from the motion database [Arikan et al. 2005; Zordan et al. 2005].

7 Experimental Result

An optical motion capture system was used to capture the motions of one actor at a time. The frame rate was set to 60 postures per second. We have captured the shadow boxing motion of an energetic kick boxer for 7 minutes, that of a tired boxer for 7 minutes, and a running-around motion for 1.5 minutes. They were automatically segmented into 279, 240 and 215 actions, respectively. These motion sets were used to control the virtual avatars. Each avatar model has 6 degrees of freedom for the translation and orientation of the root, and 72 degrees of freedom for the joint orientation. Using these data sets, various experiments based on the temporal expansion approach were conducted. The weight constants of the objective function for each experiment are shown in Table 1.

Firstly, we simulated a fight between two avatars using the actions of the energetic boxer. Although both avatars use the same action set, we can simulate different levels of intelligence by altering the depth of the game tree expansion. We simulated a less intelligent fighter by setting the intelligence level to two, and a smart fighter by setting the level to four. The intelligent fighter always wins the match as its decision is based on further expansion of the game tree.

Secondly, we simulated a match between an energetic fighter and a tired fighter. Since the motions of the tired fighter are slow, the tired avatar keeps being hit by the energetic fighter when the intelligence levels are the same. However, the tired avatar becomes stronger than the energetic fighter when it expands the game tree much deeper than that of the energetic fighter.

Thirdly, different styles of fighting were simulated by adjusting the objective function. It is known infighters prefer to fight in close distance, and hence uses short range attacks such as upper cuts and hooks more frequently. As a result, they become more aggressive as the duration of such attacks are short, and stopping the attacks will endanger the fighter as he/she will be in the reaching distance of the opponent. On the other hand, outboxers prefer to keep distance from the opponent and use long range attacks such as straight punches and kicks more often. They also move around more as they need to keep distance with the opponent. In order to simulate such effects, we first classified the attacks into short and long range ones. Then, an aggressive infighting style is modeled by setting the preferred distance to short, and giving higher score to successful short range attacks (Figure 6 (a)). The outboxing style is modeled by setting the preferred distance to long and giving higher score to successful long range attacks (Figure 6 (b)).

	$\mathbf{w}_\theta$	$\mathbf{w_r}$	$\mathbf{r_d}$	$\mathbf{w_l}$	$\mathbf{w}_\mathbf{D}^+$	$\mathbf{w}_\mathbf{D}^-$
General Boxer	10^1	10^1	$0.8m$	0	10^5	10^5
Path Follower	10^1	10^1	$0.8m$	10^1	10^5	10^5
Infighter	10^1	10^1	$0.5m$	0	$10^5/10^1$ †	10^5
Outboxer	10^1	10^1	$2.0m$	0	$10^1/10^5$ †	10^5
Chaser	10^1	10^1	$0.1m$	0	10^5	0
Runaway	10^1	10^1	$3.0m$	0	0	10^5

† *The weight of short range and long range attack respectively*

Table 1: *The weight values used to simulate various effects*

Then, a scene two fighters moving along a predefined path while fighting was simulated (Figure 6 (c)). The path is modeled as a series of check points. We made use of the desired position in Equation 1 such that higher score is given to an action that guides the avatars to the next check point. The primary objective of this animation is to let the avatars fight while the secondary objective is to let them follow the path. Therefore, the weight of location function should be smaller than that of scoring function in Equation 3.

Finally, a scene where an avatar chases another was simulated (Figure 6 (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 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 score of the each action is computed based on the status of two avatars. The chaser's score is computed by the chaser's action and the current status of the avatar running away. The score of the avatar running away is computed by its action and the status of the chasing avatar that is closer to it. When evaluating the leaf nodes of the game tree, the scores of edges by the chasers are summed. As a result, the chasers cooperate with each other to catch the avatar that is running away (Figure 7).

The computation time depends on the size of the action set, the connectivity of the motion graph, and the complexity of the objective function. In general, using a computer of Pentium 4 Dual core (3GHz) and 1GB of RAM, it takes 5 minutes to create a video of 30 seconds when expanding the game tree for three levels to determine every action of the avatars. The readers are referred to the supplementary video for further details.

8 Discussions

Using our method, it is possible to simulate the dense competitive interactions of multiple avatars based on individually captured motions. The process is fully automatic except adding semantic tags to the classified motions. Such tags are necessary for coupling the attacks and defenses. If we have a number of tagged motions already, this process can be skipped as the newly captured motions will be grouped into the corresponding group.

There are some drawbacks in our system. Firstly, the invalid combinations for pruning the subtree during the temporal expansion must be determined by the expert who knows the nature of the interactions well. Secondly, we cannot currently handle continuous contacts such as those appearing in wrestling. However, such continuous contact does not happen often in martial arts such as Karate, kick-boxing, Taekwondo, or other sports such as basketball, soccer or rugby. Therefore, our method is useful for most competitions.

When applying our method to generate an animation of mass crowd fighting, expanding the game tree for all the avatars in a single tree is computational costly and quite a waste as avatars far away cannot actually interact. We can handle such cases by first finding out the small of group of people having interactions and expand different game trees for each group. We can monitor and switch in/out the members of the group in case the distance from each other becomes smaller or larger.

The proposed method is deterministic; the action to be selected is determined only based on the min-max score computed over the interaction graph. The system can be easily switched to a probabilistic system. We can set the probability that the avatar selects each action according to the min-max score, and use the Russian Roulette approach to determine the action.

9 Conclusions

In this paper, we have presented a method to simulate competitive scenes in which multiple avatars are densely interacting with each other using singly captured motions. We have proposed a method called temporal expansion approach to determine the strategy of the avatar. We have shown that various styles of fighting can be created by changing the parameters of the game tree such as its depth and the evaluation function. We have also proposed a method to create and use the offense / defense table in order to simulate realistic interactions. For future work, we are planning to apply the methodology for collaborative activities such as carrying luggage together. As a result, we will be able to apply it to other fields such as robotics to control two robots simultaneously to conduct a collaborative task.

Acknowlegement

This project was partly supported by a CERG grant from the Research Grant Council of Hong Kong (RGC Reference No.:CityU1149/05).

References

ABE, Y., LIU, C. K., AND POPOVIĆ, Z. 2004. Momentum-based parameterization of dynamic character motion. *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 173–182.

ARIKAN, O., AND FORSYTH, D. 2002. Motion generation from examples. *ACM Transactions on Graphics 21*, 3, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. *Proceedings of 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 59–66.

GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. 2003. Snap-together motion: assembling run-time animations. *ACM Trans. Graph. 22*, 3, 702.

GLEICHER, M. 1998. Retargetting motion to new characters. *Computer Graphicsi Proceedings, Annual Conference Series*, 33–42.

GRAEPEL, T., HERBRICH, R., AND GOLD, J. 2004. Learning to fight. *Proceedings of Computer Games: Artificial Intelligence Design and Education (CGAIDE 2004)*, 193–200.

JAKOBSEN, T. 2001. Advanced character physics. *In Game Developers Conference Proceedings*, 383–401.

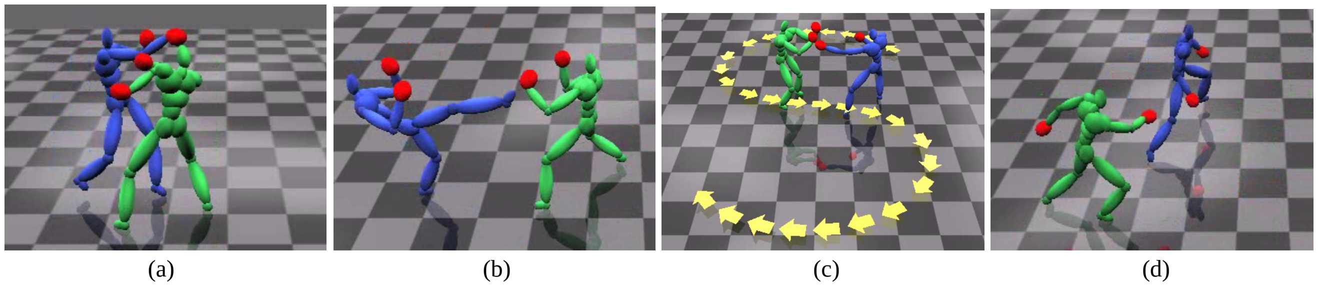

Figure 6: *Some of the screen shots of the simulated fights: (a) Infighters fighting at very close distance, (b) outboxers at long-range distance, (c) the fighters following a path while fighting, and (d) one avatar chasing another.*

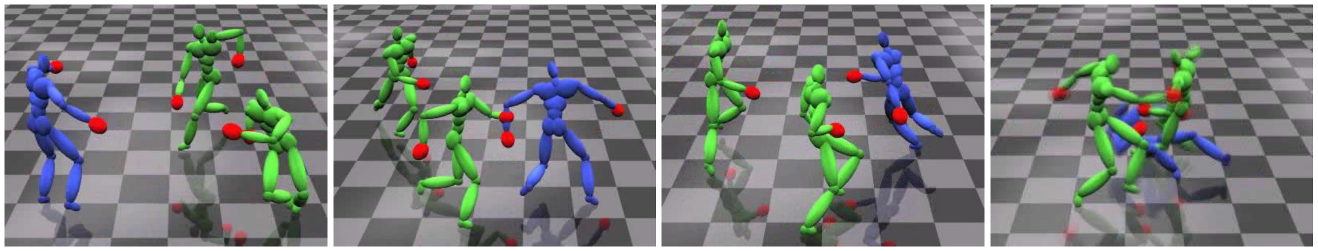

Figure 7: *Two green avatars chasing the blue avatar. The green avatars cooperate with each other to catch the blue avatar.*

KOMURA, T., HO, E. S., AND LAU, R. W. 2005. Animating reactive motion using momentum-based inverse kinematics. *Journal of Computer Animation and Virtual Worlds (special issue of CASA 2005) 16*, 3, 213–223.

KOMURA, T., LEUNG, H., AND KUFFNER, J. 2004. Animating reactive motions for biped locomotion. *ACM Virtual Reality Software and Technology*, 32–40.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics 23*, 3, 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Transactions on Graphics 21*, 3, 473–482.

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 29–38.

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 271–280.

LAU, M., AND KUFFNER, J. J. 2006. Precomputed search trees: Planning for interactive goal-driven animation. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 299–308.

LEE, J., AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. *Proceedings of 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 79–87.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of SIGGRAPH'99*, 39–48.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3, 491–500.

LIN, C.-K., PENG, J.-Y., AND TAI, W.-K. 2005. Exploring offense-defense relationship for chinese martial arts. *Proceedings of CASA 2005*, 177–182.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2006. Composition of complex optimal multi-character motions. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 215–222.

MUKAI, T., AND KURIYAMA, S. 2005. Geostatistical motion interpolation. *ACM Trans. Graph. 24*, 3, 1062–1070.

PARK, S. I., KWON, T., SHIN, H. J., AND SHIN, S. Y. 2004. Analysis and synthesis of interactive two-character motions. *Technical Note, KAIST, CS/TR-2004-194*.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics 3*.

ZORDAN, V. B., AND HODGINS, J. K. 2002. Motion capture-driven simulations that hit and react. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*.

ZORDAN, V. B., MAJKOWSKA, A., CHIU, B., AND FAST, M. 2005. Dynamic response for motion capture animation. *ACM Transactions on Graphics 24*, 3, 697–701.