Interactive Partner Control in Close Interactions for Real-Time Applications

EDMOND S. L. HO, Hong Kong Baptist University
JACKY C. P. CHAN, City University of Hong Kong
TAKU KOMURA, University of Edinburgh
HOWARD LEUNG, City University of Hong Kong

This article presents a new framework for synthesizing motion of a virtual character in response to the actions performed by a user-controlled character in real time. In particular, the proposed method can handle scenes in which the characters are closely interacting with each other such as those in partner dancing and fighting. In such interactions, coordinating the virtual characters with the human player automatically is extremely difficult because the system has to predict the intention of the player character. In addition, the style variations from different users affect the accuracy in recognizing the movements of the player character when determining the responses of the virtual character. To solve these problems, our framework makes use of the spatial relationship-based representation of the body parts called interaction mesh, which has been proven effective for motion adaptation. The method is computationally efficient, enabling real-time character control for interactive applications. We demonstrate its effectiveness and versatility in synthesizing a wide variety of motions with close interactions.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Animations, Artificial, augmented, and virtual realities*

General Terms: Design, Performance

Additional Key Words and Phrases: Close interactions, motion capture, character animation, virtual partner

ACM Reference Format:

Ho E. S. L., Chan J. C. P., Komura T., and Leung H. 2013. Interactive partner control in close interactions for real-time applications. ACM Trans. Multimedia Comput. Commun. Appl. 9, 3, Article 21 (June 2013), 19 pages.
DOI: http://dx.doi.org/10.1145/2487268.2487274

21

This work was partially supported by grants from EPSRC (EP/H012338/1), EU FP7 TOMSY, and the Hong Kong Baptist University's Research Committee start-up grant.

Authors' addresses: E. S. L. Ho (corresponding author), RRS633, Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong; email: Edmond@comp.hkbu.edu.hk; J. C. P. Chan, Department of Computer Science, City University of Hong Kong, Kowloon Tong, Kowloon, Hong Kong; T. Komura, IF1.23, Informatics Forum, School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, UK; H. Leung, Department of Computer Science, City University of Hong Kong, Kowloon Tong, Kowloon, Hong Kong.

DOI: http://dx.doi.org/10.1145/2487268.2487274

1. INTRODUCTION

In the area of virtual environments such as computer games, the demand for controlling virtual characters to closely interact with user players is rapidly increasing due to the advance of full-body motion tracking controllers. Such controllers are needed especially in movements such as dance, sports, or martial arts with nonplayer characters.

Automatically coordinating the virtual characters with the human player during close interactions in real time is an extremely difficult problem. First of all, the system needs to guess the intention of the player character from his/her movements on-the-fly and automatically decide the response motion of the virtual character such that a natural and smooth coordination can be achieved. Movements can vary significantly between users and building a system that can robustly synthesize realistic response motions of virtual characters subject to a wide variety of input movements is not an easy task. Second, even after the system has correctly figured out the intention of the player, the system needs to adapt the full-body movement of the virtual character to that of the user-controlled character taking into account the forthcoming movements and multiple constraints due to contacts between the bodies and the environment such that its body does not collide nor penetrate through the player-controlled character.

In this article, we propose a new framework to automatically infer the intention of the user player and control the virtual character in response to the player's movements, which is especially useful for real-time applications such as full-body motion games. Our framework makes use of the spatial relationship-based representation of the body parts called interaction mesh [Ho et al. 2010], which has been proven effective for motion adaptation. The abstract nature of the interaction mesh greatly helps to recognize the movement of the players in real time despite the great variance of the kinematic movements between different players. The interaction mesh can also adapt well to the movement of the players even when they move in an unexpected way, even under a condition that the body parts of the characters are highly constrained.

We mainly focus on the applications which involve highly constrained interactions with many contacts. Such interactions were difficult to be handled by previous approaches. We implemented an interactive system in which the users play the role of a virtual character and a virtual partner responds to the action of that virtual character. Figure 1 shows the setup of our proposed framework. In particular, our proposed framework captures the motion of the user by an optical motion capture system at runtime. A pair of prerecorded reference poses will be retrieved from the motion database using the live captured pose as query. Then, the selected pose pair will be edited according to the movement of the user while maintaining the spatial relationships between the body parts of the characters. By this, the context of the interactions will be preserved as in the original motion. Finally, the synthesized poses of the user-controlled character and the virtual partner will be displayed on the screen as visual feedback to the user.

We show experimental results in which the virtual partner can adapt well to the movements of the player, despite the variations between the movement performed by the player and the motion database. The system is also applicable to a wide range of other applications such as environments in which the interactions are more instantaneous, such as pedestrian relations in the streets.

1.1 Contributions

We propose a unified framework for interactively synthesizing movements by virtual partners in response to the user-controlled character. The proposed method infers the intention of the user by making use of prerecorded close interactions in the database and selects the appropriate one to facilitate the motion synthesis process. We further enhance the interaction mesh adaptation framework for adapting the prerecorded motions to the movement of the user in real time for synthesizing virtual partner

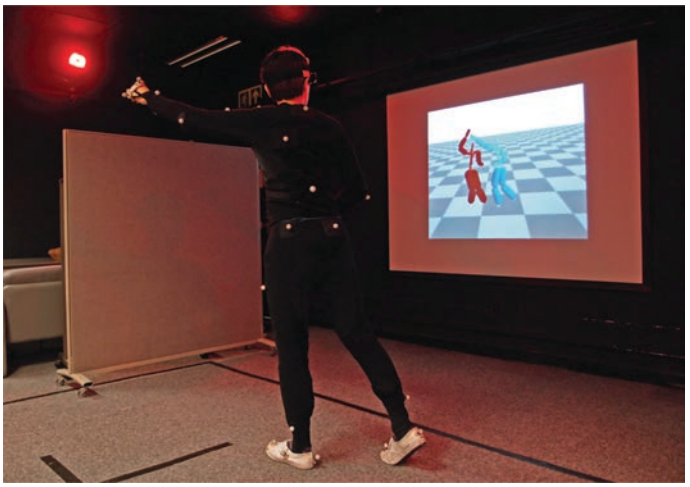

Fig. 1. The setup of our proposed framework. Our method provides the user with real-time visual feedbacks for interactive partner motion synthesis. In this example, the user is controlling a virtual character (blue) to dance with the virtual partner (red).

motions. By this, our method can handle a wide variety of close interactions between characters such as those in partner dancing and fighting, which is difficult to be handled by previous methods.

2. RELATED WORK

In this section, we review the related research in synthesizing close interactions for humanoid characters and robots. First, we review the work in synthesizing close interactions in the computer graphics and animation community in Section 2.1. Next, we introduce the work proposed in the robotics community for controlling partner robots in response to the movements of human in Section 2.2. While there is a broad range of research in human-robot interactions, we are particularly interested in the work on handling close interactions which share the same interests with our work.

2.1 Synthesizing and Controlling Close Interaction for Computer Animation

Synthesizing close interactions of humanoid characters has been an active research area in the computer graphics and animation community. Liu et al. [2006] proposed a method to create scenes such as one character avoiding another and a mother holding the hand of a child by using spacetime optimization. However, the method requires the user to specify the constraints to control how the characters move, which is not suitable for generating the partner's response to the user-controlled character automatically. Lee and Lee [2004] simulated boxing interactions by using reinforcement learning. Treuille et al. [2007] also used reinforcement learning to simulate pedestrians avoiding each other. Shum et al. [2007] proposed a method to compute the optimal action in a competitive environment by using min-max search. They also proposed a real-time approach based on an automatically produced finite state machine [Shum et al. 2008]. However, these methods do not handle very close interactions such as dancing.

Ho and Komura [2009a, 2009b, 2011] proposed to use tangles from the Knot theory to analyze and synthesize close interactions (such as wrestling and dancing) between two characters. While their method can avoid collisions and penetrations between the body segments of the characters, the method becomes less effective when the body parts are not tangled, which limits the variety of interactions/motions that can be handled. Ho et al. [2010] proposed to use an interaction mesh for editing

and retargeting close interactions and showed that different kinds of motions such as judo, dancing and fighting can be handled. In this research we will apply the interaction mesh motion adaptation framework for editing close interactions in real time.

While various methods for generating close interactions have been proposed, less attention has been paid on synthesizing virtual characters responding to the actions performed by user-controlled characters for interactive applications. Hsu et al. [2004] showed an example of generating the motions of a virtual dance partner according to the control motion of the leader. Their method selects an optimal sequence of synchronized partner dance motion segments based on the input motion. However, the whole control motion sequence is required for selecting the best matched partner dance motion segments from the database. In addition, the optimal sequence of motion segments is selected by solving a computational costly optimization problem, which makes the method not applicable for real-time applications.

Previous work in synthesizing a virtual dance partner according to the movements of the user [Tsuruta et al. 2007; Deng et al. 2011; Tang et al. 2011] mainly focus on selecting the best matched motion sequence from the database. These methods do not edit the motion of the virtual partner according to the motion of the user but simply display the recorded motions which do not contain any body contact between the characters. As a result, close interactions cannot be handled or artifacts such as collisions and penetrations between body parts of the characters are likely to be presented in the synthesized motions.

2.2 Synthesizing Partner's Response Motion in Robotics

While there is limited work in generating a virtual partner's response to a user-controlled character in the computer animation community, researchers in robotics have been working on controlling robots automatically to react to the actions performed by humans. In particular, ballroom dance partner robots [Takeda et al. 2005, 2007b, 2007a; Nakayama et al. 2009; Sakai et al. 2007] and fighting robot [Nakamura and Yamane Laboratory 2005] have been developed to interact with humans. The details will be discussed in the following sections.

2.2.1 *Partner Dance Robots.* In order to effectively coordinate the movement of the partner dance robot with the human dancer, the future dance steps of the human dancer are estimated by using a Hidden Markov Model (HMM) based on the human intention detected from the force sensors attached to the upper body of the robot [Takeda et al. 2005, 2007b; Nakayama et al. 2009]. By this, the robot acts as the follower in partner dance to interact with the human dancer. The dance step size can further be adjusted according to the interaction between the robot and human [Takeda et al. 2007a] in order to make the dance motion look more natural. On the other hand, planning the movements of the robot who plays the leading role in partner dance has been proposed [Sakai et al. 2007]. In Sakai et al. [2007], an HMM is again used for estimating the next step of the follower and collision avoidance is taken into account when planning the movements of the robot.

Our proposed method is different from the previous work in dance partner robots as follows: First, we intend to generate full-body motion of the humanoid characters/robots. In Takeda et al., [2005, 2007a, 2007b]; Nakayama et al. [2009], and Sakai et al. [2007], the robot has very limited degrees of freedom. The lower body is composed by three wheels and a rigid frame. As a result, these methods cannot be directly applied to full-body motion control or a new motion planning/generator has to be proposed. Second, their methods assume that the human dancer will select one of the steps in the predefined ballroom dance (i.e., waltz) step transitions in the step prediction. Since different types of dance have different step transitions, this assumption limits the method to work with a specific type of dance. On the other hand, our framework is designed to handle different kind of motions.

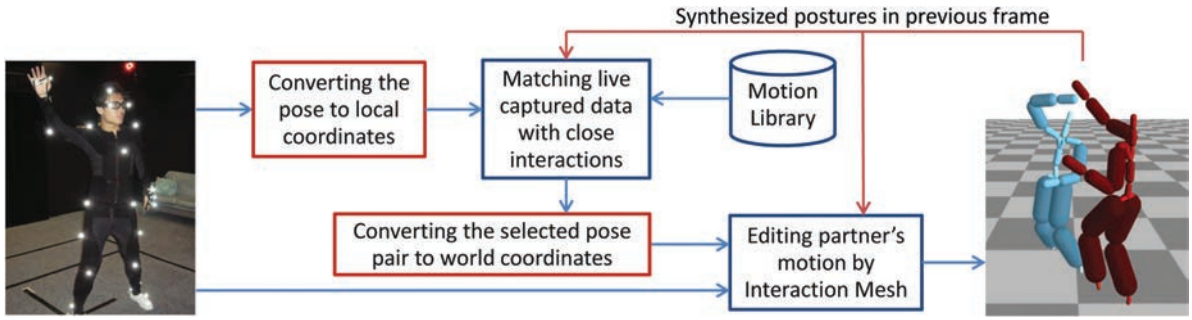

Fig. 2. Overview of the proposed framework.

2.2.2 *Fighting Robots.* Other than dance partner robots, the Nakamura laboratory [Nakamura and Yamane Laboratory 2005] studied automating their high-mobility robot [Sugihara et al. 2005] to fight and interact with a human [Lee et al. 2009]. In these previous works, the movement of the player is captured by a motion capture system at runtime. The *mimesis loop* which is based on an HMM is then applied for recognizing the behavior of the user by analyzing the live captured motions. Next, the *self-behavior*, which is the behavior of the robot, is generated based on the learned interaction patterns [Inamura et al. 2004]. Finally, the motion corresponding to the self-behavior for controlling the humanoid robot is corrected into physically valid motion in real time [Sugihara and Nakamura 2005].

While it is expected that the robot can intelligently interact with the human by using the methods proposed in Nakamura and Yamane Laboratory [2005], it is difficult to apply their method for handling close interactions such as those in partner dancing. When synthesizing fighting interactions, body contacts caused by punching and kicking can easily be maintained by enforcing contact constraints in their method. However, close interactions such as tangling the limbs without body contact in dancing and dodging in fighting cannot be handled due to the lack of representation of the spatial relationships in the motion editing process. As a result, the context of the interaction will be lost. On the other hand, our method can handle different kinds of close interactions by encoding spatial relationships (both contacting and noncontacting body parts) extracted from the reference motion data.

3. METHODOLOGY

3.1 Overview

The overview of the proposed framework is shown in Figure 2. First, a motion library is constructed using paired motions with two subjects who are closely interacting with each other (Section 3.2) in the preprocessing stage. During runtime, the live captured pose of the user is used as query to retrieve a pair of poses from the motion library (Section 3.3) for synthesizing the movement of the virtual partner. Next, the selected pose pair is edited by the interaction mesh according to the live captured pose while maintaining the context of the interaction (Section 3.4). Finally, the edited poses are rendered on the screen as feedback to the user.

3.2 Motion Library

The motion library in our proposed framework contains motions captured from two closely interacting subjects. Each motion is represented by a sequence of poses and each pose q is represented by a vector of parameters, $q = \{q_1, \ldots, q_n\}$, where q_i is a 3D vector which contains the world coordinates of the i^{th} joint and $1 \leq i \leq n$, and n is the total number of joints of the virtual character. In our motion library,

the virtual character has 25 joints and the dimensionality of q is 75. Each frame carries two poses of two subjects A and B, and each subject takes one of the two roles; the *active role* $Role_A$ and *the passive role* $Role_S$. While the subjects may switch roles throughout the motion sequence (e.g., subjects take turns to attack/lead each other), we assume that only one of the subjects, takes $Role_A$ and the other one will be followed/respond passively in each frame.

3.2.1 *Poses Normalization.* In order to facilitate the motion selection process, the root translations and rotation about the vertical axis should be ignored [Lee et al. 2002; Arikan et al. 2003; Kovar and Gleicher 2004]. However, we cannot simply normalize the poses in each pair separately as the relative distance and orientation between the two subjects have to be maintained in order to preserve the context of the interaction. For this reason, we first convert the locations of all joints from the world coordinates to the local coordinates of the character in $Role_A$ in each pose pair. More specifically, given a pair of poses, q_A and q_S, of the characters in $Role_A$ and $Role_S$, respectively, we represent the locations of all the joints by the relative positions from the root of character in $Role_A$ to the joints. The final step in the coordinate system conversion is to remove the rotation about the vertical axis applied to the characters, which is usually represented by the orientation of the root joint. Since we know the rotation about the vertical axis applied to the root of the character in $Role_A$, we can compute the corresponding 3×3 transformation matrix RM_A^{root} and cancel out the rotation by multiplying $(RM_A^{root})^{-1}$ to the relative positions. After canceling out the root translation and rotation about the vertical axis, the normalized poses can be compared directly.

In summary, the normalized poses p_A and p_S are computed by the following equations.

$$p_A = \begin{bmatrix} (RM_A^{root})^{-1} \\ \vdots \\ (RM_A^{root})^{-1} \end{bmatrix} * \left(\begin{bmatrix} q_A^1 \\ \vdots \\ q_A^n \end{bmatrix} - \begin{bmatrix} q_A^{root} \\ \vdots \\ q_A^{root} \end{bmatrix} \right) \tag{1}$$

$$p_S = \begin{bmatrix} (RM_A^{root})^{-1} \\ \vdots \\ (RM_A^{root})^{-1} \end{bmatrix} * \left(\begin{bmatrix} q_S^1 \\ \vdots \\ q_S^n \end{bmatrix} - \begin{bmatrix} q_A^{root} \\ \vdots \\ q_A^{root} \end{bmatrix} \right) \tag{2}$$

Here q_A^i and q_S^i are the position of the i^{th} joint in poses q_A and q_S in the prerecorded motion, respectively, q_A^{root} is the position of the root joint in pose q_A, and p_A and p_S are the normalized poses of the characters in $Role_A$ and $Role_S$ in local coordinates, respectively. Note that p_A and p_S have the same dimensionality (i.e., 75 in our implementation) as the poses q_A and q_S in the database.

Since we do not have the information on which subject (A or B) took $Role_A$ in the prerecorded pose pair, our system converts each pair of prerecorded pose into two pairs of normalized poses: one pair is normalized by assuming subject A takes $Role_A$ and the other pair assumes subject B takes $Role_A$. As a result, the size of the motion library is doubled in our implementation. If the roles of the subjects are annotated in the captured motion, we can normalize the pose pairs accordingly without increasing the size of the database.

3.2.2 *kd-Tree Indexing.* In order to efficiently retrieve similar poses from the motion library, a kd-tree index structure has been used in our proposed method. The kd-tree search structure scales well to large unstructured databases such as those containing over one million poses [Krüger et al. 2010]. As a result, it can greatly improve the performance of our motion selection process (Section 3.3) and enables our method to be used for real-time interactive applications. To further improve the performance, we defined a 15-dimensional feature set for comparison as in Krüger et al. [2010]. Specifically,

the 15 features are the 3D locations of 5 joints (end-effectors): *head, right hand, left hand, right foot, and left foot* in p_A and we build the kd-tree by using this feature set. In our experiments, only a few seconds are required for creating the kd-tree on a laptop computer and it was done in the preprocessing stage.

3.3 Motion Selection

Now we explain the motion selection process using the live captured motion as query. Specifically, the selection process is divided into two stages.

(1) Select the *k-nearest neighbors* $\mathcal{K}$ from the motion library.
(2) Select the best matched pair of poses based on the:
   (a) Euclidean distance between the poses of the character in $Role_A$ in $\mathcal{K}$ and the live captured pose of the user, and
   (b) temporal coherence between the candidate pose pair and the synthesized motion in the previous frames.

3.3.1 *Query Pose.* The live captured pose q_{in} of the user is taken as the query for retrieving a similar pose pair for motion synthesis. By this, we simulate the situation in which the user takes $Role_A$ and the virtual avatar takes $Role_S$. However, we cannot directly compare the poses in the motion library with q_{in}. It is because the sizes/lengths of the body segments of the user are not necessarily the same as the subjects who performed the motions in the database. To solve this problem, we first retarget q_{in} to q'_{in} which has the same body structure as the subject who takes $Role_A$ in the database.

Here, we retarget q_{in} to q'_{in} using an approach similar to the motion retargetting proposed by Gleicher [1998]. Specifically, we retarget the live captured motion to the body structure used in our database by Inverse Kinematics (IK). Similar to Gleicher [1998], the foot planting constraints are enforced to preserve the stepping pattern in the original motion. We also constrain the positions of the end-effectors on the upper body (i.e., head, left hand, and right hand) when solving the IK problem to preserve the upper body movement of the user in the motion retargetting process. Finally, the retargetted pose q'_{in} is converted into local coordinates p'_{in} by multiplying the inverse of the rotation of the root joint $(RM_{in}^{root})^{-1}$ in q'_{in} to the relative positions computed from the root joint q'^{root}_{in} to all joints. This process is similar to the pose normalization explained in Section 3.2.1. Since the root translation and rotation are removed in the normalized pose, we can directly compare p'_{in} with the normalized poses in the database.

3.3.2 *Selecting the k-Nearest Neighbors from the Motion Library.* Next, we select the 10-nearest-neighbors $\mathcal{K}$ from the motion database using the 15-dimensional feature set extracted from p'_{in}. In our experiments, we searched for the k-nearest neighbors by using the ANN library [Mount and Arya 2006]. As accurately retrieving the poses is very important in our framework, we validate the selection of k and the dimensionality of the feature set by the performance evaluation in Section 4.1.

3.3.3 *Selecting the Best Matched Pair of Poses.* Now we compare p'_{in} to all of the corresponding poses of the active role in $\mathcal{K}$. Here, we used the point-cloud distance metric [Kovar et al. 2002] for comparing the similarity of the poses. We also consider the temporal coherence of the synthesized motion by taking into account the poses generated in previous frame $\{s_{A,i-1}, s_{S,i-1}\}$, where i is the index of the frame being synthesized. Since sudden change of moving direction of body segment can easily result in visually discontinuous motions, the accelerations of the joints in the desired poses of the characters in consecutive frames should not be large. Based on this assumption, the desired poses are computed by the velocities of the synthesized poses in the previous frame. Here we estimate the poses of character in $Role_A$ and $Role_S$ at the frame being edited as $\{s_{A,i^*}, s_{S,i^*}\}$ by using the velocities of

$\{s_{A,i-1}, s_{S,i-1}\}$

$$s_{A,i^*} = \dot{s}_{A,i-1}\Delta t + s_{A,i-1}, \tag{3}$$

$$s_{S,i^*} = \dot{s}_{S,i-1}\Delta t + s_{S,i-1}, \tag{4}$$

where $\dot{s}_{A,i-1}$ and $\dot{s}_{S,i-1}$ are the velocities of the joints of the characters in $Role_A$ and $Role_S$ in the previous frame, respectively, and Δt is the step size which is 1/15s in the experiments. In order to bias the pose searching process to the desired pose, we introduced a term in the cost function (Eq. (7)) to measure the Euclidean distance between the desired pose $\{s_{A,i^*}, s_{S,i^*}\}$ and the poses in $\mathcal{K}$. However, the poses in $\mathcal{K}$ cannot be compared to $\{s_{A,i^*}, s_{S,i^*}\}$ directly as the poses are represented in different coordinate systems. For this reason, the poses in $\mathcal{K}$ will be converted into world coordinates for computing the Euclidean distance. In particular, the rotation RM_{in}^{root} of the root joint in q'_{in} will be used in the conversion since the synthesized postures should have the same orientation as in q'_{in} captured from the user. We have

$$q''_{A,j} = \begin{bmatrix} RM_{in}^{root} \\ \vdots \\ RM_{in}^{root} \end{bmatrix} * \begin{bmatrix} p_{A,j}^1 \\ \vdots \\ p_{A,j}^n \end{bmatrix}, \tag{5}$$

$$q''_{S,j} = \begin{bmatrix} RM_{in}^{root} \\ \vdots \\ RM_{in}^{root} \end{bmatrix} * \begin{bmatrix} p_{S,j}^1 \\ \vdots \\ p_{S,j}^n \end{bmatrix}, \tag{6}$$

where $q''_{A,j}$ and $q''_{S,j}$ are poses in world coordinates converted from the j^{th} pair of poses (i.e., $p_{A,j}$ and $p_{S,j}$) in $\mathcal{K}$, and $1 \leq j \leq k$. To balance the trade-off between being responsive to the user (i.e., closely following the live captured motion) and producing smooth and continuous motion (i.e., with small acceleration between consecutive frames), weights are introduced in the cost function in Eq. (7). As a result, the cost function becomes

$$dist(p'_{in}, \{s_{A,i^*}, s_{S,i^*}\}, p_{A,j}, \{q''_{A,j}, q''_{S,j}\}) = \sum_{l=1}^{n} \left[\left(p'^{l}_{in} - p^{l}_{A,j}\right)^2 + \alpha\left(\left(s^{l}_{A,i^*} - q''^{l}_{A,j}\right)^2 + \left(s^{l}_{S,i^*} - q''^{l}_{S,j}\right)^2\right)\right], \tag{7}$$

where α is the weight for temporal smoothness and is set to 0.5 in our experiments, l is the index of the joint, n is the total number of joints of each character, which is standardized in the motion database.

We select the pose pair with smallest weighted sum of Euclidean distance returned from the cost function (Eq. (7)) for the motion synthesis process. In particular, the selected pose pair in world coordinates will be edited according to q'_{in} and the details will be explained in next section. In the rest of this article, the selected poses will be represented by $\{q''_{A,sel}, q''_{S,sel}\}$.

3.4 Motion Synthesis

Since the pose of the character in active role $q''_{A,sel}$ in the selected pose pair should be similar to, but not necessarily the same as, the pose captured from the user q'_{in}, it is possible that the synthesized motion is different from the performance of the user if the selected pose pair is not edited accordingly. To maintain the consistency between the movement of the user and the synthesized motion, the pose of the character in active role $q''_{A,sel}$ will be morphed to the pose of the user q'_{in}. In order to preserve the context of the motion, we apply the interaction mesh motion adaptation method proposed by Ho et al. [2010] to edit the selected poses $\{q''_{A,sel}, q''_{S,sel}\}$. An example is shown in Figure 3(a)–(d). In the rest of this section, we will explain how we adopt the proposed framework in Ho et al. [2010] to our method.

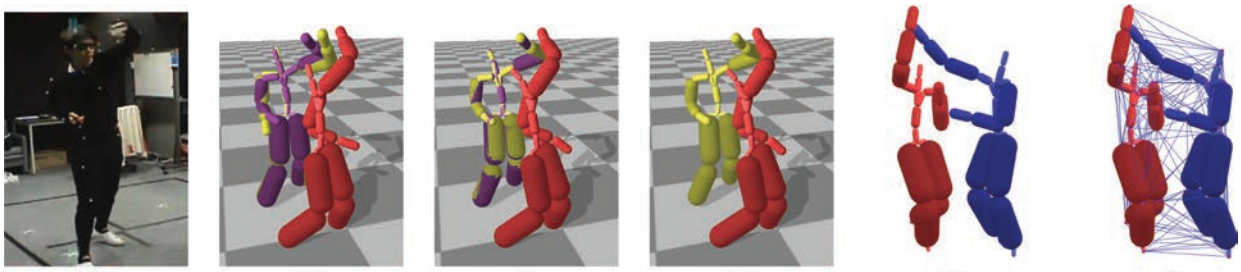

Fig. 3. (a)–(d) The selected poses are edited by the interaction mesh motion adaptation framework [Ho et al. 2010]. (a) The posture of the user; (b) the posture of the user (purple), selected leader pose (yellow), and selected follower pose (red), (c) intermediate poses edited by interaction mesh; (d) final poses of the leader and follower; (e)–(f) an example of interaction mesh extracted from a pair of salsa dance poses; (e) the original postures; (f) the interaction mesh extracted from the poses. Dance motion data used in creating these figures are obtained from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001].

3.4.1 *Interaction Mesh Preparation.* For each pair of poses in the motion library, a corresponding interaction mesh will be computed in the preprocessing stage (offline). Interaction mesh [Ho et al. 2010] is a volumetric mesh composed of vertices and edges. The vertices are the locations of joints and points sampled from the surface of the characters and objects in the virtual environment. The structure of the interaction mesh is then constructed using edges computed by Delaunay tetrahedralization of the point cloud (i.e., the vertices). By performing Delaunay tetrahedralization, the vertices which are close to each other tend to be connected by edges. As a result, the vertices sampled from the closely interacting body parts and objects are connected in the interaction mesh. By minimizing the distortion of the interaction mesh while editing the poses, the spatial relationships can be preserved.

In this research, we focus on interactions between characters only. We apply the Delaunay tetrahedralization [Si and Grtner 2005] to the point cloud that contains the locations of the joints sampled from the two characters in each pair of pose. An example of the interaction mesh extracted from a pair of salsa dance poses in our motion database is shown in Figure 3(e)–(f).

3.4.2 *Editing the Poses.* Now we present the method for editing the selected poses $\{q''_{A,sel}, q''_{S,sel}\}$. Our goal is to edit the selected poses $q''_{A,sel}$ to be as similar to the live captured pose q'_{in} as possible while preserving the context in the pose pair $\{q''_{A,sel}, q''_{S,sel}\}$. In addition, the edited poses should also be similar to the synthesized poses in the previous frame $\{s_{A,i-1}, s_{S,i-1}\}$ to maintain the temporal coherence and reduce jaggy jumps between consecutive frames. We formulate this motion edit process as an optimization problem and the details are explained as follows.

—*Notations.* Let m be the number of vertices in the interaction mesh, $v^i_j (1 \leq j \leq m)$ be the vertices at frame i, V_i be a vector of size $3m$ that includes all v^i_j such that $V_i = ({v^i_1}^{\mathrm{T}}, \ldots, {v^i_m}^{\mathrm{T}})$, and ${v^i_j}'$ and V'_i be the updated vectors after motion editing.

—*Energy Functions.* As in Ho et al. [2010], the following energy functions are used in our method, namely *deformation energy* $E_L(V'_i)$, *velocity energy* $E_V(V'_i, V'_{i-1})$, and *constraint energy* $E_C(V'_i)$. Minimizing the deformation energy can minimize the distortion of the interaction mesh. As a result, the spatial relationships of the characters can be preserved. In order to maintain the smoothness and continuity of the motion, the velocity energy is also minimized. Finally, the soft constraints are formulated as constraint energy.

—*Constraints.* Here we explain the constraints enforced in the spacetime optimization. In particular, we use the bone-length constraints $C_B(V_i')$, morphing constraints $C_M(V_i')$, and collision constraints $C_C(V_i')$ proposed in Ho et al. [2010]. We further introduce the antifoot-sliding constraints $C_F(V_i')$ in our method. The details of each constraint are explained next.

Bone-length constraints are used for maintaining the distance between adjacent joints of characters as the body segments are rigid. In order to prevent penetration between the bounding volumes of the body segments, collision constraints are enforced and the colliding body segments will be moved apart.

Morphing constraints. In this research, the selected pose $q''_{A,sel}$ will be gradually morphed to q'_{in} such that the synthesized pose of the character in $Role_A$ will be similar to the pose of the user. At every morph-step, which will be further explained shortly, an intermediate target pose $q_{interpolated}$ is linearly interpolated by the following equation

$$q_{interpolated} = \frac{i}{m}(q'_{in} - q''_{A,sel}) + q''_{A,sel}, \tag{8}$$

where m is the total number of morph-steps and i is the index of the current morph-step. Next, the morphing constraints are updated using $q_{interpolated}$ as the target pose

$$C_M(V_i') = q_{interpolated} - V'_{A,i}, \tag{9}$$

where $V'_{A,i}$ contains the positions of the vertices sampled from the currently editing pose of the character in $Role_A$.

Antifoot-sliding constraints. In order to avoid artifacts such as foot-sliding appearing in the synthesized motion, we further introduce an additional constraint to fix the position(s) of the supporting foot/feet on the ground. First, the system detects whether a foot is landed on the ground by checking the height of the foot (i.e., the y-position in world coordinates). If the foot is detected as landed, the system will further check the landing status of the foot in the previous frame. In case that the foot was landed in both of the previous and current frame, the antifoot-sliding constraint will be enforced. We have

$$C_F(V_i') = V'_{feet,i-1} - V'_{feet,i}, \tag{10}$$

where $V'_{feet,i}$ and $V'_{feet,i-1}$ are the positions of the vertices sampled from the joints on the landed foot/feet in the current and previous frame, respectively.

Soft and hard constraints. In the proposed method, antifoot-sliding constraints are hard constraints. Since maintaining the correct bone lengths and avoiding penetrations between body parts are important, bone-length and collision constraints are also set as hard constraints. In order to prevent the system from overconstraining, morphing constraints are set as soft constraints to stabilize the motion.

—*Iterative Morphing.* At every morph-step, the intermediate target pose $q_{interpolated}$ and morphing constraints are updated using Eqs. (8) and (9), respectively. Finally, the poses of the characters are adapted by minimizing the sum of the deformation, velocity, and constraint energy subject to the hard constraints $H_i V_i' = h_i$. The adapted motion is computed by solving

$$\arg\min_{V_i',\lambda_i(1\le i\le n)} E_L + w_\Delta E_V + E_C + \lambda_i^T(H_i V_i' - h_i), \tag{11}$$

where V_i' is the set of new vertex positions at the current frame, λ_i are the Lagrange multipliers and w_Δ is a constant weight (we use 0.2). The optimization problem in Eq. (11) can be solved by differentiating it with respect to V_i' and λ_i, and solving a system of linear equations. The reader is referred to Ho et al. [2010] for further details.

Table I. Accuracy of Pose Selection by Searching Method M_A

Size of nearest-neighbor search	Best pose found	Either of the best 2 poses found	Time required (ms)
5	92.85%	95.73%	13.8
10	95.07%	97.73%	14.9
15	96.54%	98.75%	15.8
20	97.87%	98.98%	17.0
25	98.09%	98.98%	18.0
30	98.75%	99.20%	19.9
35	98.75%	99.20%	21.5
40	98.75%	99.20%	23.1
45	99.20%	99.42%	24.9
50	99.20%	99.42%	26.7

4. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed method with five experiments. The experiments were conducted on a desktop computer with Intel Core 2 Duo 2.4 GHz processor for motion synthesis and the Motion Analysis Eagle Digital optical motion capture system for acquiring the motion of the user. Our motion library contains the salsa dance motion data from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001] and fighting motion data synthesized by the method proposed in Shum et al. [2007]. There are 15464 frames of dance motions and 13740 frames of fighting motions captured at a frame rate of 60 Hz. For solving the linear equations stated in Section 3.4.2, UMFPACK [Davis 2004] and GotoBLAS [Goto and Van De Geijn 2008] were used.

4.1 Performance Evaluation on Motion Selection

In the first experiment, we compared the performance of our proposed method with a brute-force method. The purpose of this experiment is to evaluate the effectiveness of the reference poses selection process using the proposed nearest-neighbor search and the 15-dimensional feature sets for the poses. We randomly chose 1000 dancing and fighting poses from the motions captured from novice users, which are different from those captured from professional dancers and fighters in the motion database, as queries for pose pair selection. For each query, we compared the results returned by two searching methods: (M_A) the 10-nearest-neighbors search using the 15-dimensional feature set, and (M_B) brute-force search which sequentially searches all the poses in the motion library and compare the poses using Eq. (7). The computation time required by the search algorithms and accuracy of the retrieved poses were compared and summarized in Table I.

Regarding the accuracy of our proposed method, the best matched pose found in M_B is included in the 10-nearest neighbors returned by M_A in 95.07% of the queries. While the best matched pose returned by M_B was not found in around 5% of the queries, we found that either of the best two matched poses returned from M_B can be found in the 10-nearest neighbors returned by M_A in nearly 98% of the queries.

Increasing the size of the nearest-neighbor search and the dimensionality of the feature sets for the poses can further improve the chance of including the best matched pose in the result set. However, additional computation is required and we argue that the selection of the size of nearest-neighbor search and the feature set should depend on the availability of computational resources. To balance the interactive performance and the accuracy of the motion selection in our proposed framework, we chose 10-nearest-neighbor search and the 15-dimensional feature set as in Krüger et al. [2010] for all the experiments.

Table II. Computation Time Required for the Two Searching Methods M_A and M_B

Number of pose pairs in the motion database	M_A	M_B
7500	12.4ms	40.5ms
15000	13.6ms	77.0ms
30000	15.0ms	145.2ms

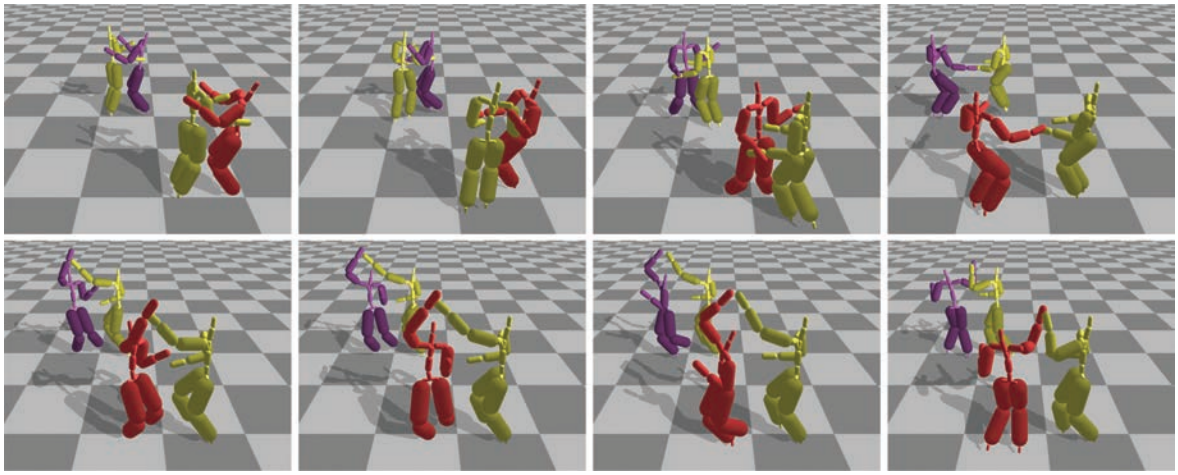

Fig. 4. Snapshots of our leave-one-out cross-validation experiment (salsa dance). The partner dance motion pair (yello: Leader, purple: Follower) is removed from the motion database. The motion of the leader is used as user input motion. A virtual partner (red) is synthesized accordingly. Dance motion data used in creating these figures are obtained from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001].

In addition, we also compared the average time in query required by the two searching methods and the results are summarized in Table II. The computation costs show that the kd-tree indexing scales well and enables efficient motion search in a large motion database. This is particularly important for real-time interactive applications as the motion selection has to be done at every frame.

In summary, the results show that the selection of retrieving 10-nearest neighbors using the 15-dimensional feature set can obtain results which are close to the optimal results returned from brute-force search while improving the performance of our method significantly.

4.2 Leave-One-Out Cross-Validation

In the second evaluation, we performed the leave-one-out cross-validation. The purpose of this experiment is to focus on evaluating performance of our proposed framework while excluding the factors affecting the quality of the synthesized motion caused by the performance of the user (e.g., errors introduced by the skill level of the user). Here we removed a pair of salsa dance motion from the database and use the motion of the leader as query. The snapshots of the synthesized dance motion are shown in Figure 4. In Figure 4, the characters colored in yellow and purple were performing the captured motions of the leader and follower, respectively. The motion of the character colored in red is synthesized by our method.

To quantitatively evaluate the effectiveness of our proposed method, we computed the success rate of the salsa moves in the synthesized motion. Here, the success rate is computed by

$$Success\ Rate = \frac{Number\ of\ Successful\ Moves}{Total\ Number\ of\ Moves}. \tag{12}$$

Table III. Success Rates of the Synthesized Motions

Experiment	Total moves	Successful moves	Success rate
Salsa Dance - Leave-one-out (expert)	27	26	96.30%
Salsa Dance - User A (beginner)	40	34	85.00%
Salsa Dance - User B (beginner)	31	26	83.87%
Fighting - Leave-one-out (expert)	26	24	92.31%

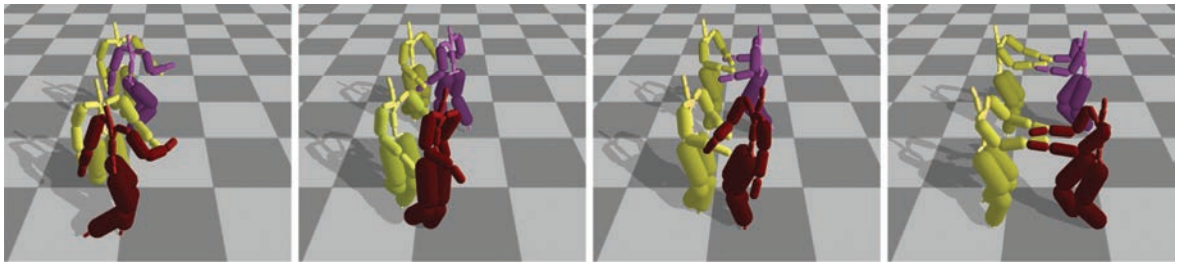

Fig. 5. Snapshots of our leave-one-out cross-validation experiment (salsa dance). Note that the synthesized dance partner (red) rotated in a different direction comparing to the captured motion. Dance motion data used in creating these figures are obtained from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001].

We counted the number of successful moves by checking the labels of the synthesized move and the corresponding move in the original motion. If the labels are the same, the synthesized move will be classified as a successful move. We labeled and categorized the salsa moves in the database into 6 categories: Normal open holds, Normal closed holds, Leading a left/right turn, Cross body lead, Walks, and In and Out. The results are summarized in Table III. The results show that 96.30% of the moves of the virtual partner were successful moves. Since some of the moves in the captured motion pair are not included in the database, the synthesized motion may differ visually from the original motions in some of the moves as shown in Figure 5 and that explains why the success rate is not 100%. Nevertheless, valid salsa moves were created and the experimental results show that smooth and continuous partner dance animation can be synthesized by using our method.

4.3 Partner Dance Motion Synthesis

In the third experiment, we interactively synthesized a virtual dance partner by using live captured dance motions of two novice users. Figures 6 and 7 show the screenshots of the dance performed by the users (left) and the synthesized partner dance motions (right). The synthesized motions were rendered and projected on to a large screen on the wall as a real-time visual feedback to the user (Figure 1). As the motion can be synthesized at 15 frames per second, the user can interactively adjust his/her performance to create the desired partner dance motion. The readers are referred to the attachment video for the resulting motions.

Again, we quantitatively evaluated the effectiveness of our method by computing the success rate of the synthesized motion. Specifically, we asked the user to mimic the basic salsa moves contained in our motion database. First, the users studied the salsa moves by watching the animation created by the salsa dance motions in the database. Next, the users tried to perform the salsa moves in random order. Here, a successful move means that the synthesized partner (follower) moves in a way that the user wants. The results are summarized in Table III. The results show that over 83% of the moves of the partner can be synthesized correctly in all the experiments. For the failed moves, we found that most of the moves performed by the users are not similar to the motions in the database. As the users

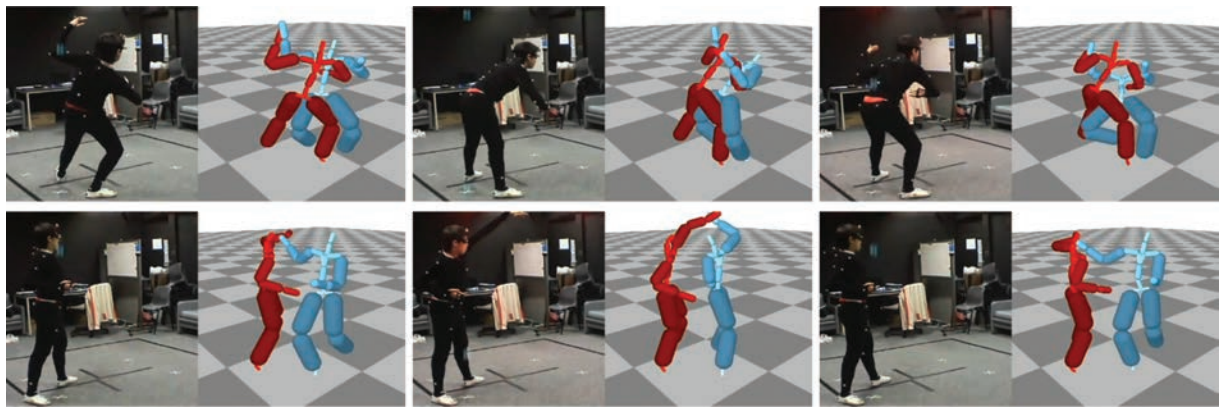

Fig. 6. Synthesized partner dance motions (by User A). (left) Live capturing the dance motion of the user. (right) The synthesized dance motions of the leader (red) and follower (blue). Dance motion data used in creating these figures are obtained from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001].

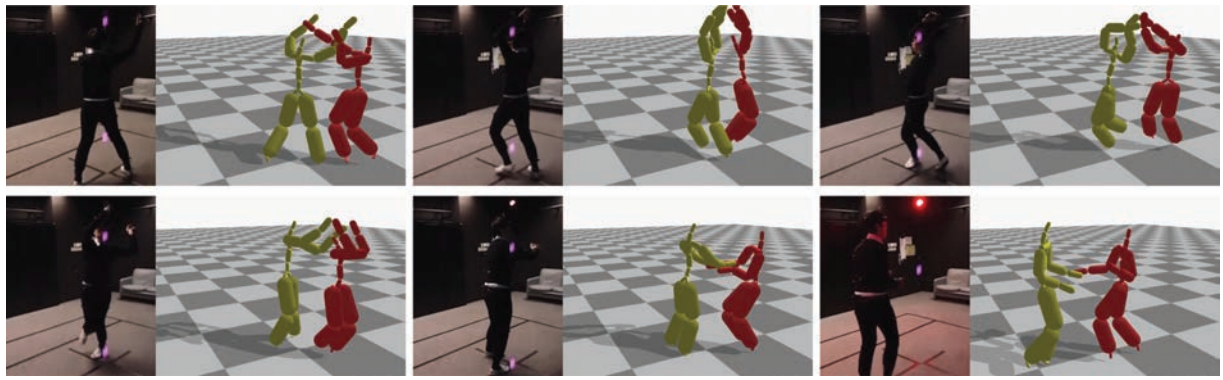

Fig. 7. Synthesized partner dance motions (by User B). (left) Live capturing the dance motion of the user. (right) The synthesized dance motions of the leader (yellow) and follower (red). Dance motion data used in creating these figures are obtained from the CMU Motion of Body (MoBo) Database [Gross and Shi 2001].

participating in this experiment are beginners, we expect that experienced salsa dancers can achieve a higher success rate which can be comparable to the results obtained in the leave-one-out experiment in Section 4.2.

4.4 Fighting Motion Synthesis

In the last experiment, we interactively synthesized fighting motions using the proposed method. This experiment is to show that our method is applicable for a wider range of multicharacter interactions. Both attacking (e.g., punching, kicking, etc.) and defending moves (e.g., dodging, blocking, etc.) are included in our motion library. We extracted a sequence of 1000 frames from the collected motions and use it as the input for motion synthesis, which is similar to the leave-one-out cross-validation explained in Section 4.2. The results are shown in Figure 8. Note that the synthesized opponent reacts to the input motion accordingly. When the character controlled by the input motion (Figure 8 yellow) attacks, the synthesized opponent (Figure 8 red) defends, and vice versa. The readers are referred to the attachment video for the resulting motions.

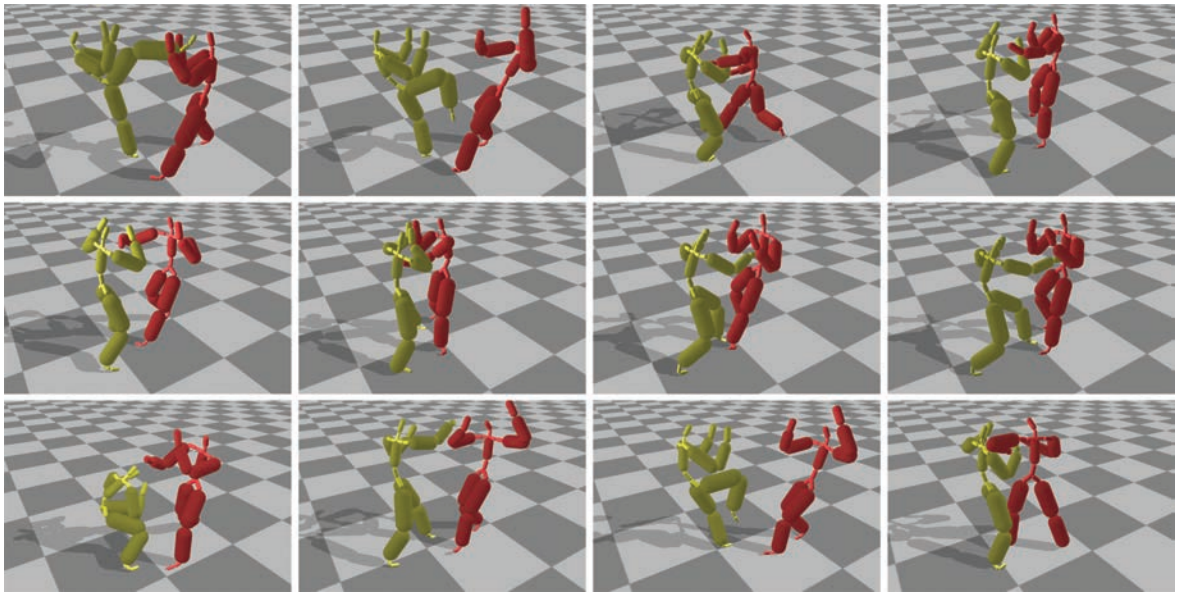

Fig. 8. Synthesized fighting motion. The synthesized opponent (red) reacts to the input motion (yellow) accordingly. Fighting motion data used in creating these figures are synthesized by the method proposed in Shum et al. [2007].

We also performed the quantitative evaluation explained in Section 4.2 on the synthesized fighting motion. We labeled and categorized the fighting moves into four categories: kicking, punching, dodging, and blocking. The results are summarized in Table III. From the results, 92.31% of the moves of the virtual opponent were successful moves. While there are a small number of failed moves, continuous and smooth fighting interactions were synthesized.

Through this experiment, we demonstrated that our proposed method is general and able to handle different kinds of close interactions. In particular, the user can perform both offensive and defensive moves to lead the virtual partner to launch appropriate movements, either attacks or defenses, accordingly. When comparing with the experiments in previous sections, the body contacts involved in partner dancing and fighting are different. In partner dancing, there are a lot of body contacts between the characters. But for fighting interactions, such as the dodging (avoiding) motions, there can be no body contact between the characters. It is difficult to apply traditional motion editing techniques to handle these cases as we cannot enforce contact constraints for preserving the context of the interaction. On the other hand, our proposed method applies spatial relationships for synthesizing close interactions which can handle interactions with and without body contacts.

4.5 Evaluating the Naturalness of the Synthesized Motion

We further carried out an experiment to qualitatively evaluate the naturalness of the motion synthesized by our proposed method. In particular, the motion is evaluated by the Zero Moment Point (ZMP) error metric [Ikemoto et al. 2007] which computes the distance between the ZMP and the support polygon if the ZMP is outside the support polygon. Since there is no guideline for a good threshold value to decide whether a motion is natural or not from the ZMP error, we compare the ZMP errors between the synthesized dancing and fighting motions in the leave-one-out tests (Section 4.2 and 4.4) and the corresponding original motions. The results are listed in Table IV. The very similar ZMP errors computed from the motions indicate that the motions synthesized by our proposed method should be as natural

Table IV. Average Zero Moment Point (ZMP) Error Distances (per frame) Computed from the Dancing and Fighting Motions

Motion type	Original motion	Synthesized motion
Salsa Dance	0.70	0.72
Fighting	2.24	2.21

as the original motions. As our method enforces antifoot-sliding constraints to handle the foot-sliding problem, the foot-sliding metric proposed in Ikemoto et al. [2007] is not used in this evaluation.

In summary, the experimental results show that the proposed method successfully synthesized smooth and continuous motion of the virtual partner responding to the character controlled by the live captured motion of the user in close interactions such as dancing and fighting. For partner dance, we tested our method by users with different skill levels: the motions of professional salsa dancers (i.e., from the CMU motion database [Gross and Shi 2001]) in the leave-one-out cross-validation in Section 4.2, as well as the beginners in Section 4.3. The proposed framework can be applied to interactive applications such as computer games and virtual dance learning systems for users at any skill level. Finally, we further tested our method with fighting motions to show that our method is general and able to handle different kinds of multicharacter close interactions.

5. DISCUSSION

In this section, we will discuss some of the assumptions and limitations in the motion selection process and the construction of the motion database in the proposed method. First of all, we assume that the user takes $Role_A$ in the interaction. Nevertheless, it is also possible to use the live captured motion of the user as $Role_S$ and synthesize the motion of the virtual partner in $Role_A$ accordingly. Switching between the roles is another interesting approach and it can enhance the user's experience. For example, in the fighting motions synthesis experiment in Section 4.4, switching $Role_A$ (i.e., attacking in fighting) between the characters is more reasonable and realistic. However, for partner dance applications, letting the user take $Role_A$ enables a higher level of control and the user can direct how the partner dance motion sequence will be synthesized. To conclude, the design of the role switching scheme is application dependent. For motion synthesis, treating the user input motion in $Role_A$ increases the controllability of the results. On the other hand, switching the roles between the user-controlled character and the virtual partner/opponent can enhance the user's experience, which is more suitable for applications such as interactive learning and computer games.

Next, we explain the decision on selecting a single pair of motion from the database for reference. In character animation, it is common to create new motion by interpolating multiple motion sequences in the motion database. By this, we can reduce the size of the motion database by generating new motions at runtime. However, such an approach is difficult to apply on synthesizing close interactions because artifacts such as interpenetrations of the body segments are likely present in the synthesized motions. Taking into account the spatial relations of the body parts of the characters when selecting candidate poses for interpolation [Ho and Komura 2009b] can be one of the solutions. Interpolating the postures by nonlinear methods such as SGPLVM [Grochow et al. 2004] is another interesting future direction.

Finally, we discuss the selection of close interactions for making up the database. In the experiments, our motion database contains a single type of fighting and partner dance (i.e., the salsa) motions. A broader range of motions can be synthesized by adding different kinds of close interactions to the motion library. Given the complexities of O(n log n) for kd-tree construction and O(k log n) for k-nearest-neighbors search where n is the number of pose pairs in the database, the proposed method scales well to the size of motion database. We believe that increasing the size of the motion library does not have

significant impact on the performance of our proposed method. Another potential problem arising from the increase in number of motions in the database is the ambiguity in selecting the reference pose pairs from similar moves for motion synthesis. Specifically, when there are two or more pose pairs having the pose of the character in $Role_A$ which is similar to the live captured pose of the user, any of the pose pairs can be selected and thus the user will have less control on the resultant interaction. Since our method only takes into account the live captured pose and the previously generated pose for reference motion selection, additional information has to be given to our method in such an ambiguous case. For example, the user can give feedback to our system to specify which pose or action he/she wants when similar pose pairs are found. An interesting future direction will be automating the process to remove ambiguity by recognizing and analyzing the movements of the user in the action level over time for predicting the intention of the user in the future for selecting the appropriate reference pose pair. Techniques such as "motion symbol tree" and "motion symbol graph" proposed in Takano et al. [2011] for recognizing and predicting human behaviors from observation will be explored.

6. CONCLUSION AND FUTURE WORK

In this article, we proposed a framework for synthesizing a motion of a computer-controlled character in response to the user-controlled character during close interactions in real time. We implemented an interactive system based on the proposed method which enables the user to control the motion of the virtual partner intuitively and interactively. Experimental results showed that realistic partner motions in dancing and fighting can be generated in real time.

The proposed method can be applied to a wide variety of applications such as interactive computer games, character animation, virtual dance learning systems [Magnenat-Thalmann et al. 2008; Chan et al. 2011], etc. By extending our method to take into account the physical limitations of the characters, it is possible to synthesize motions for controlling humanoid robots to closely interact with humans with body contact. In addition, we plan to produce a dancing system that can manage different types of dances as a future research direction.

ACKNOWLEDGMENT

The authors thank the anonymous referees for their helpful comments. They also thank Dr. Hubert P. H. Shum of Northumbria University for providing the fighting motion data used in this project. The salsa dance motion data used in this project was obtained from mocap.cs.cmu.edu.

REFERENCES

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. *ACM Trans. Graph. 22*, 3, 402–408.

CHAN, J. C. P., LEUNG, H., TANG, J. K. T., AND KOMURA, T. 2011. A virtual reality dance training system using motion capture technology. *IEEE Trans. Learn. Technol. 4*, 2, 187–195.

DAVIS, T. A. 2004. Algorithm 832: UMFPACK V4.3—An unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw. 30*, 2, 196–199.

DENG, L., LEUNG, H., GU, N., AND YANG, Y. 2011. Real-time mocap dance recognition for an interactive dancing game. *Comput. Animation Virtual Worlds 22*, 2–3, 229–237.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'98).* ACM Press, New York, 33–42.

GOTO, K. AND VAN DE GEIJN, R. 2008. High-performance implementation of the level-3 blas. *ACM Trans. Math. Softw. 35*, 1, 1–14.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIC, Z. 2004. Style-based inverse kinematics. *ACM Trans. Graph. 23*, 3, 522–531.

GROSS, R. AND SHI, J. 2001. The cmu motion of body (mobo). Tech. rep. CMU-RI-TR-01-18. Database Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. http://mocap.cs.cmu.edu.

HO, E. S. L. AND KOMURA, T. 2009a. Character motion synthesis by topology coordinates. *Comput. Graph. Forum 28*, 2, 299–308.

HO, E. S. L. AND KOMURA, T. 2009b. Indexing and retrieving motions of characters in close contact. *IEEE Trans. Vis. Comput. Graph. 15*, 3, 481–492.

HO, E. S. L., KOMURA, T., AND TAI, C.-L. 2010. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph. 29*, 4, 1–8.

HO, E. S. L. AND KOMURA, T. 2011. A finite state machine based on topology coordinates for wrestling games. *Comput. Animat. Virtual Worlds 22*, 5, 435–443.

HSU, E., GENTRY, S., AND POPOVIC, J. 2004. Example-based control of human motion. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'04)*. Eurographics Association, 69–77.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2007. Quick transitions with cached multi-way blends. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'07)*. ACM Press, New York, 145–151.

INAMURA, T., NAKAMURA, Y., TOSHIMA, I., AND TANIE, H. 2004. Embodied symbol emergence based on mimesis theory. *Int. J. Robotics Res. 23*, 4–5, 363–377.

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

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. Graph. 21*, 3, 473–482.

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

LEE, D., OTT, C., AND NAKAMURA, Y. 2009. Mimetic communication with impedance control for physical human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'09)*. 1535–1542.

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 Trans. Graph. 21*, 3, 491–500.

LEE, J. AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'04)*. Eurographics Association, 79–87.

LIU, C. K., HERTZMANN, A. AND POPOVIC, Z. 2006. Composition of complex optimal multi-character motions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'06)*. Eurographics Association, 215–222.

MAGNENAT-THALMANN, N., PROTOPSALTOU, D., AND KAVAKLI, E. 2008. Learning how to dance using a web 3d platform. In *Proceedings of the 6th International Conference on Advances in Web Based Learning (ICWL'07)*. H. Leung, F. Li, R. Lau, and Q. Li, Eds., Lecture Notes in Computer Science, vol. 4823, Springer, 1–12.

MOUNT, D. M. AND ARYA, S. 2006. ANN: A library for approximate nearest neighbor searching. Programming manual. College Park, Maryland. http://www.cs.umd.edu/~mount/ANN/.

NAKAMURA AND YAMANE LABORATORY. 2005. Animatronic humanoid robot project in prototype robot exhibition, Aichi EXPO. http://www.ynl.t.u-tokyo.ac.jp/research/expo2005/expo2005-e.html.

NAKAYAMA, D., KOSUGE, K., AND HIRATA, Y. 2009. Human-adaptive step estimation method for a dance partner robot. In *Proceedings of the IEEE International Conference on Automation and Logistics (ICAL'09)*. 191–196.

SAKAI, Y., TAKEDA, T., HIRATA, Y., AND KOSUGE, K. 2007. Collision avoidance based on estimated step of other dance couples for male-type dance partner robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*. 3264–3269.

SHUM, H. P. H., KOMURA, T., AND YAMAZAKI, S. 2007. Simulating competitive interactions using singly captured motions. In *Proceedings of the ACM Symposium on Virtual Reality Software Technology*. 65–72.

SHUM, H. P. H., KOMURA, T., AND YAMAZAKI, S. 2008. Simulating interactions of avatars in high dimensional state space. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D'08)*. ACM Press, New York, 131–138.

SI, H. AND GRTNER, K. 2005. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, B. W. Hanks, Ed. Springer, 147–163.

SUGIHARA, T. AND NAKAMURA, Y. 2005. A fast online gait planning with boundary condition relaxation for humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*. 305–310.

SUGIHARA, T., YAMAMOTO, K., AND NAKAMURA, Y. 2005. Architectural design of miniature anthropomorphic robots towards high mobility. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*. 2869–2874.

TAKANO, W., IMAGAWA, H., AND NAKAMURA, Y. 2011. Prediction of human behaviors in the future through symbolic inference. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'11)*. 1970–1975.

TAKEDA, T., HIRATA, Y., AND KOSUGE, K. 2007a. Dance partner robot cooperative motion generation with adjustable length of dance step stride based on physical interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*. 3258–3263.

TAKEDA, T., HIRATA, Y., AND KOSUGE, K. 2007b. Dance step estimation method based on hmm for dance partner robot. *IEEE Trans. Indust. Electron. 54*, 2, 699–706.

TAKEDA, T., KOSUGE, K., AND HIRATA, Y. 2005. HMM-based dance step estimation for dance partner robot -ms dancer. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*. 3245–3250.

TANG, J. K. T., CHAN, J. C. P., AND LEUNG, H. 2011. Interactive dancing game with real-time recognition of continuous dance moves from 3D human motion capture. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication (ICUIMC'11)*. ACM Press, New York.

TREUILLE, A., LEE, Y., AND POPOVIC, Z. 2007. Near-optimal character animation with continuous control. In *ACM SIGGRAPH Papers*. ACM Press, New York.

TSURUTA, S., KAWAUCHI, Y., CHOI, W., AND HACHIMURA, K. 2007. Real-time recognition of body motion for virtual dance collaboration system. In *Proceedings of the International Conference on Artificial Reality and Telexistence*. IEEE Computer Society, 23–30.

Received April 2012; revised January 2013; accepted April 2013