Performance-Driven Dance Motion Control of a Virtual Partner Character

Christos Mousas[*]
Graphics & Entertainment Technology Lab
Southern Illinois University, Carbondale, IL 62901, U.S.A.

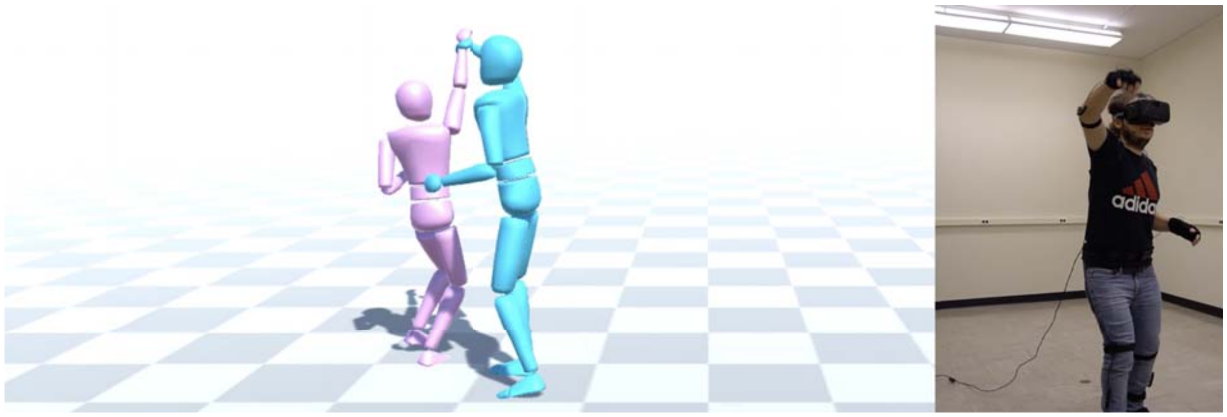

Figure 1: An example of a virtual partner's synthesized pose based on a user's performance.

Abstract

Taking advantage of motion capture and display technologies, a method giving a user the ability to control the dance motions of a virtual partner in an immersive setup was developed and is presented in this paper. The method utilizes a dance motion dataset containing the motion of both dancers (leader and partner). A hidden Markov model (HMM) was used to learn the structure of the dance motions. The HMM was trained on the motion of a chosen dancer (leader or partner), and during runtime, the system predicts the progress of the chosen dance motion, which corresponds to the progress of the user's motion. The regular structure of the HMM was extended by utilizing a jump state transition, allowing the user to improvise dance motions during the runtime. Since the jump state addition increases the model's complexity, an effort was made to optimize the prediction process to ensure runtime efficiency. A few corrective steps were also implemented to ensure the partner character's motions appear natural. A user study was conducted to understand the naturalness of the synthesized motion as well as the control that the user has on the partner character's synthesized motion.

Index Terms: Human-centered computing—Interaction paradigms—Virtual reality; Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality; Computing methodologies—Computer graphics—Animation—Motion capture

1 Introduction

Dance is a part of human culture and it has been around for at least the past several thousand years. According to Raftis [35] references to dance can be found in very early recorded history. In these days, with the growth of virtual reality technologies people are interested in extending it into virtual environments [2] [22]. Therefore, given the recent advances in low-cost motion sensing and display technologies, developing a method that provides users the ability to dance with a virtual partner was considered. Specifically, an effort was made to transfer a real-world experience into an immersive context, dancing with a partner in a salsa dance style.

In the presented implementation, it was considered that when people dance, they can both follow specific dance movements as well as improvise by weaving movements together and changing the order in which movements appear. Therefore, it was crucial to provide a way for users to improvise while dancing. To achieve this, an effort was made to construct a method that not only recognizes the progress of the user's motion to display the corresponding movements of the partner dancer, but also allows the user to improvise the dance and have the partner follow the user's improvisation. A simple example of a user controlling the dance motions of a partner character is shown in Figure 1.

To create a motion controller for the partner character, a four-phase method was developed. In the first phase, the leader's (or the partner's) motions are used to construct feature vectors for later use in training the HMM. The second phase includes the construction of the proposed HMM and the methods for training it. The third phase describes the optimization of the forward algorithm to allow the user to improvise during the application's runtime. It should be noted that two motion sequences are always required. The first corresponds to the leader and the second to the virtual partner character. Based on the role (leader or partner) taken by the user during runtime, the corresponding motion of the virtual character is used to train the HMM. The fourth and final phase of the method includes the required correction steps to align the predicted motion of the partner character. These correction steps are important since they ensure the final synthesized motion seems natural. After developing the motion control mechanism for the partner character, a user study was conducted to evaluate the properties of the motion refinements.

The remainder of the paper is divided into the following sections. Section 2 discusses previous work related to the presented method. The dataset, motion representation, and pre-processing are covered in Section 3. Section 4 presents the HMM that was developed and trained, and the optimization of the forward algorithm that improves control of the partner's dance motions. The real-time correction steps developed to adjust the virtual partner's motions are presented in Section 5. Section 6 covers the implementation details. The

[*]e-mail: christos@cs.siu.edu

conducted user study is presented in Section 7. Finally, Section 8 discusses the conclusions and future directions for further research.

2 Related Work

Interacting with a virtual character through computers has been examined extensively in recent years. In addition to methods allowing users to control a virtual character's movements based on a performance capture process (performance animation / motion reconstruction) [28] [23] [33], there are numerous other approaches to user-character interaction. Among these, Lee et al. [23] proposed a method to control a real-time virtual character using a motion capture system. In their method, the motion of the character is contained in a database and is pre-processed using a two-layer technique. A Markov process is used in the lower layer, and a clustering technique is used in the upper layer. Ho et al. [16] developed a framework for synthesizing motion of a virtual character in response to the actions performed by a user-controlled character in real time.

Other examples of user-character interaction include the multimodal social agent interaction framework called Marve developed by Babu et al. [4]. In this framework, multimodal input is used to make a virtual character respond through a rule-based system. Another framework developed by Jaksic et al. [18] provides feedback to users in an online store based on their facial emotions. Since a user's body language is known to be an important component of expressing emotion, frameworks such as Piavca developed by Gillies et al. [14] also incorporate gestures to help users interact with the virtual character.

The action of dancing has been examined in various ways in computer animation. Many early methods focused on dance motion synthesis based on music data [19] [12]. This type of method integrates motion data and corresponding music features. Then, given a new music track, the system predicts a motion for a specific music segment. Other researchers have worked on dance motion analysis, including the evaluation and classification of emotions [3], or on learning dance in virtual environments [30] [15]. A couple of solutions for learning dance through a virtual character are worth mentioning. One virtual reality framework developed by Chua et al. [9] allows users to practice the Chinese Tai Chi martial art. In their approach, the user observes the teacher's motions and mimics it until the user performs a similar motion as the virtual character. In another martial art training framework proposed by Komura et al. [21], users wear a motion capture suit and a head-mounted display (HMD) to practice defensive and offensive movements with a virtual character. Based on various criteria such as the total number of defense movements performed correctly and reaction time, the user is given a performance evaluation.

Like the previously mentioned research work, only the development process of such a system is examined in this paper, although the current system could easily be extended and used as a dance training system, such as the one developed by Chan eta al. [8]. Generally, this paper presents the development of a virtual reality application that provides people the opportunity to interact in a virtual environment by experiencing real-life actions in an immersive virtual reality context, which is dancing with a user-controlled virtual partner. Among previously researched virtual reality approaches, those proposed by Yang et al. [42] and Deng et al. [11] should be discussed. In both approaches, users were asked to wear motion capture suits. Specifically, Yang et al. [42] proposed a collaborative dance application between remote dancers in a tele-immersive environment. Deng et al. [11] developed a dance game based on motion capture technology. They also addressed the issue of real-time prediction of the user's performance to determine what interactive motion should be displayed by a virtual dance partner. The real-time estimation was based on indexing body parts in conjunction with flexible matching to estimate the completion of a motion as well as to reject unwanted motions.

Due to the context of this research, it is worth mentioning related work on controlling the dance motions of partner robots. Using an HMM is one approach to coordinating the movement of a partner dance robot based on the footsteps of the user [40] [39]. Based on this strategy, the robot takes the role of the partner following the human dancer, much like the virtual character is treated in this paper. By adjusting the step size of the robot [38], it was possible to make the dance motion appear more natural as well as allow the user to modify dance motions. In addition to the robot being in the partner role, there is also an approach proposed by Sakai et al. [36] in which the robot plays the lead dancer. In this approach, HMM was used to estimate the next step of the partner and collision avoidance was also considered when planning the movements of the robot. The presented method of virtual partner dance motion control is different from the previous work on dance partner robots [40] [39] [38] [36] in several ways. The mentioned methods assume that the user will always follow a predefined dance step pattern. Moreover, the robots do not provide flexible control mechanisms (e.g., dance maneuvers with hand contact), which is something that can be easily addressed in a virtual reality context.

In summary, the developed application allows users to dance with a virtual partner character in real-time. The implemented jump state transition derived from [31] allows users to improvise the dance performance while the character responds to user's motion. In this paper, the constrained training process between the states of the model allows the system to search and consequently to jump only to valid states of the model, therefore wrong motions are eliminated. Moreover, the flexibility of the virtual character allows the user to be more precise with the dance movements. Finally, in the presented implementation, users can choose which role they want to play (leader or partner) and the system's variations are adjusted automatically.

3 Preliminaries

This section presents the dataset and methods for managing motion data.

3.1 Dance Motion Dataset

This research benefits from CMU MoCap, which is a publicly available dance motion capture dataset [7] that was utilized in and tested with the presented approach. It should be noted that even though there are many dance motion datasets, this dataset was determined to be ideal for developing and evaluating a method for controlling a virtual dance partner's motions since it contains several salsa dance sequences. Figure 2 shows a salsa dance motion sequence retrieved from CMU MoCap. In this dataset, the quality of the motion capture is at a high standard and the motion data is properly synchronized and spatially aligned since both performers were captured simultaneously, which is helpful since it eliminates the need to do synchronization or any other pre-processing. If a different dataset is used at some point, synchronization and spatial alignment techniques [17] [24] might be required to deal with these issues. There is no corresponding extended documentation for CMU MoCap to describe the number of different dance couples, and the characteristics of the dancers such as physical dimensions and level of expertise for each participant in the database even though CMU MoCap has been used extensively previously by the research community. Finally, it should be noted that randomly chosen salsa dance movements from CMU MoCap were used in the user study (see Section 7).

3.1.1 Motion Representation

In both datasets, every dance performance is divided into two motion capture files, one for the leader and one for the partner. Each motion is represented by the root position p and joint rotations r_g where $g = 1, ..., G$ and G denotes the total number of joints. Therefore, at time t the pose of the character is represented as $[p(t), r_1(t), ..., r_G(t)]$. Here, for simplicity, X represents the motion of the leader dancer

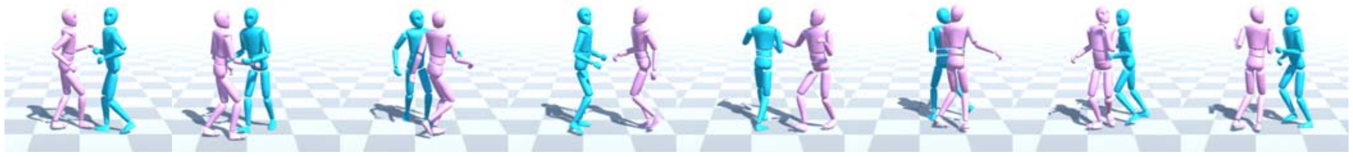
Figure 2: A salsa dance motion sequence including the leader (blue) and the partner (pink) character.

and Y is the motion of the dance partner, and these correspond with x_t and y_t, the poses of each dancer at the $t-th$ frame. Moreover, since the motions are synchronized, it can be stated that the $t-th$ pose of X corresponds to the $t-th$ pose of Y.

3.2 Construction of the Feature Vector

The motion data should be represented in a way the system can recognize and follow the user's motions. Therefore, the construction of a feature vector to hold information on the lead dancer's motions (or the partner's) is necessary. Note that the user can choose his or her role before running the application. Based on the chosen role, the system can easily adapt the feature vectors and can be trained to recognize the user's dance performance.

There are a variety of potentially useful motion features for efficiently recognizing dance motion sequences, such as Laban Movement Analysis (LMA) features [3]. However, in the current implementation, a simplified feature vector was constructed. Specifically, the feature vector ϕ at time t takes the following inputs:

- the posture of the character (excluding the root position and rotation) at the $t-th$ frame of the motion x_t (or y_t),
- the velocity, v, of the character's root at the $t-th$ frame v_t, and
- the speed s_t of the end effector (hands and feet) at time t, which is represented as $s_t = [s_{rf}, s_{lf}, s_{rh}, s_{lf}]$.

The constructed feature vector at time t used for training the HMM is represented as $\phi_t = [x_t \text{ (or } y_t), v_t, s_t]$. A few details should be noted here about the root position and rotation. The root position and rotation are excluded from x_t (or y_t) because the space in which the developed virtual reality application runs is different from the one where the motion data was captured. Therefore, calibrating the environment as well as initializing the motion data according to the user's position might be required. Another issue that inclusion of the root position and rotation data might cause is related to the ability of the user to jump to different points within motion sequences. Therefore, spatial alignment might be lost during the user's improvisation; this issue is resolved during the post-processing phase (see Section 5.1).

4 Performance-Following Using HMM

The HMM [27] [29] [10] treats a user's performance-following behavior as an inverse problem of estimating the position of the score (pose of a virtual partner character) from a user's full-body motion (motion of the leading dancer). The estimate is achieved by training the HMM to the corresponding motion of the leader (or partner) character. This functionality can be described as a process that evolves from its current state to the next one and predicts an output pose that corresponds to the performed one. By associating the poses of the user on score with a hidden state, the performance can be easily interpreted as a simple state transition sequence. However, as a user might want to improvise by weaving together different parts of the dance motion sequence, an extension of the standard HMM was developed to deal with this issue. According to Cano et al. [6], the state transition and prediction of the output pose can be described as a stochastic process. The construction of the HMM, including the jump state functionality derived from [31], its training process, and the way the forward algorithm is optimized are described in the rest of this section.

4.1 Constructing the States of the HMM

During the runtime of an application, the user should not only be able to follow the exact pattern of the template dance motion but also improvise within the dance. Therefore, to provide a reasonable solution in which the HMM treats the input performance of the user in a relaxed way, three different transition states were developed (see Figure 3 for a graphical representation). The first two transitions the HMM can handle are self-transition (see Equation 1), in which the model repeats the same state and the next transition in which the model evolves to the next state (see Equation 2).

$$A_{i\to i} = a_i + (1-a_i)\times A_i \tag{1}$$

$$A_{i\to i+1} = (1-a_i)\times A_i \tag{2}$$

In the above equation, $A_{i\to j}$ denotes the state transition probability matrix between the current state i and the next state j (the next state might be the $i-th$ in case the self transition is predicted, or the $j-th$ state $(i+1)$ in case the next transition is predicted). The durational self-transition probability a_i is determined by matching the expected staying time with the duration z_i of the $i-th$ state, which yields:

$$\begin{aligned} z_i &= \sum_{k=1}^{\infty} k\times a_i^{k-1}\times(1-a_i) \\ &= \frac{1}{(1-a_i)} \end{aligned} \tag{3}$$

The topology of the self and next transitions is expressed with a left-to-right transition structure to neighbor states of the HMM. Because giving a user the ability to improvise is an interesting issue worth implementing in a character control mechanism, an additional transition state, the jump state $A_{j\to N\to i}$, was considered. This state allows the user to jump to either an earlier or a later state and consequently jump to the corresponding dance movement.

Experimentation conducted during the development and testing process revealed that users typically pause briefly before skipping to a remote part of the dance motions. Therefore, to achieve the jump transition to a remote state, an independent pause state L_j and a resume R_i transition was developed to perform the jump. It should be noted that both L_j and R_i were introduced to constrain the model. Hence, the jump state can be easily understood if represented as a two-step transition process through the pause state as shown in Figure 3. The jump state $A_{j\to N\to i}$ can be defined as the tendency to pause and resume the performance and are expressed as the transition probabilities to the pause state and from the pause state to a remote state through a jump. Based on L_j and R_i, the transition matrices can be written as:

$$A_{j\to i} = B_{j\to i} \tag{4}$$

$$A_{j\to N} = L_j \tag{5}$$

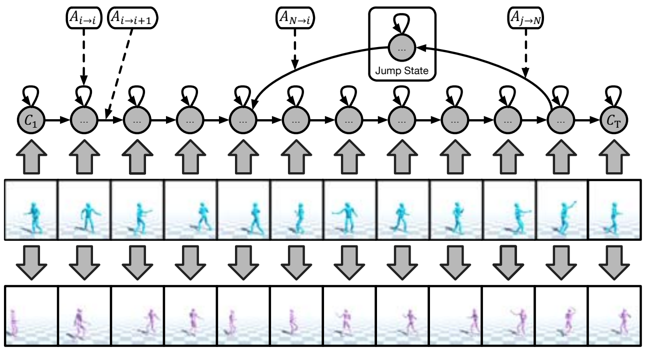

Figure 3: The representation of the developed HMM used in this paper. As can be observed, there are three transition states developed in the presented method.

$$A_{N\to i} = (1 - A_{N\to N}) \times R_i \tag{6}$$

where $i, j \in [1, M]$, the $N - th$ state denotes the pause state and is defined as $N = M + 1$, and $B_{j\to i}$ denotes a band matrix representing self and next transitions.

4.2 Training the HMM

The training process is based on setting an HMM to recognize the captured motion of the user by using the motion of the corresponding dance performance as a template motion. Since this approach does not consider capturing variations of a user's action, a single motion sequence is used to control the evolution of the partner's motion. Therefore, it can be said that limited training data is used since only a single example of the captured data is set for the training process. The structure of the HMM's state is set directly from a single template, which is a time sequence represented by $\phi_1, ..., \phi_T$ where ϕ_t represents the feature vector of the chosen dancer, as described in the previous section.

The $\phi_1, ..., \phi_T$ sequence is then used to create a left-to-right Markov chain represented by $C_1, ..., C_T$ states. As mentioned, three transitions are considered in this implementation: the self-transition $A_{i\to i}$, the next transition $A_{i\to j}$, and the jump transition $A_{j\to N\to i}$. Because the data is regularly sampled, the transition probabilities are set manually as $A_{i\to i} = A_{i\to j} = A_{j\to N\to i} = 0.33$ and $A_{j\to N} = A_{N\to i} = 0.165$. It should be noted that the combined phases of the jump transition ($A_{j\to N}$ and $A_{N\to i}$) were determined to share equal probabilities with the self and next transitions, such that $A_{j\to N} + A_{N\to i} = 0.33$.

For the training process, considering an observation O (O is a measured vector of length T), the probability $b_j(O)$ in a state j is set to a Gaussian distribution with vector ϕ_j as the center. Apparently, a single template is not enough to fully estimate the required distributions. Therefore, a simplified form for estimating the required distributions is used based on the following equation:

$$b_j(O) \propto \exp\left[-\sum_{t=1}^{T} \frac{(O_t - \phi_{j_t})^2}{2 \times \sigma^2} \right] \tag{7}$$

where σ denotes the standard deviation between the measured and template data.

4.3 Performance-Following and Optimization

During the presented method's runtime, the system estimates the score position of the partner's pose given the user's input motion. The score position is predicted by computing the most probable state $\hat{C}_t$ given the user's input motion $\phi_1, ..., \phi_t$ and the state random variable C_t at time t such as:

$$\hat{C}_t = \arg\max_{C_t} p(C_t | \phi_1, ..., \phi_t) \tag{8}$$

Based on Bayes' theorem, Equation 8 becomes:

$$\hat{C}_t = \arg\max_{C_t} p(\phi_1, ..., \phi_t, C_t) \tag{9}$$

Equation 9 can now be solved easily by applying the forward algorithm, in which its update rule is represented as:

$$a_t(i) = b_i(\phi_t) \times \sum_{j=1}^{J} a_{t-1}(j) \times A_{j\to i} \tag{10}$$

where $a_t(i) := p(\phi_1, ..., \phi_t, C_t = i)$ describes the forward variable. The initial variable $a_1(i) = b_i(\phi_1) \times \pi_i$ is calculated with the initial distribution π_i.

Even though the previously mentioned standard forward algorithm procedure is quite efficient for handling the self and next transitions, it is quite complex in cases requiring a calculation of the jump transition. The complexity of solving Equation 10 is $O(M^2)$ since there are M summations over M states. Since this implementation deals with a continuous and long streams of motion data (a dance performance might be more than a minute), a reduction of the complexity of the model seems necessary. Therefore, to improve algorithm's runtime efficiency, the constraints allowing the jump to a remote state were applied to reduce the algorithm's complexity to a linear order. This was achieved by simply substituting Equations 4-6 into Equation 10, hence the update rule $a_t(i)$ of the forward algorithm is represented by:

$$b_i(\phi_t) \times \left[\sum_{j=i-2}^{i} a_{t-1}(j) \times B_{j\to i} + \left(\sum_{j=1}^{J} a_{t-1}(j) \times L_j \right) \times R_i \right] \tag{11}$$

It should be noted that the second summation of Equation 11 is independent of i, which means it is sufficient to calculate this only once at each estimation. Therefore, the above equation's computational complexity becomes $O(M)$, making it quite suitable for handling efficiently long motion sequences in real-time.

5 Real-Time Partner's Motion Correction

After predicting the state in which the input pose of the user belongs, the system can easily display the corresponding pose of the partner character. However, there are a few reasons why the partner character might not have proper spatial alignment with the lead character. The first is related to the physical and virtual space used during the initial motion capture process. The second is that there might be some variation in the user's performance of the dance motions. For example, the partner's motion at time t has a specific position in the virtual space. When the user decides to jump to another state (another part of the motion sequence), the predicted motion of the user has a new and possible different spatial position. Hence, the predicted motion of the partner character after the jump state should be spatially aligned with the previously predicted motion (the last frame of predicted motion before reaching the jump state). Another issue that needed adjustment is related to contact between the hands of the leader (the user-controlled character) and the partner character. A simple inverse kinematics (IK) approach was adopted to properly correct the necessary hand contact in specific time steps of the dance motion. Finally, other issues related to the synthesis process of the partner character as well as the transition between the different parts of the motion, based on the jump transition state, also cause discontinuities in the synthesized motions sequence. Therefore, the final adjustment ensures the synthesized motion is smooth.

5.1 Spatial Alignment

To solve the first issue mentioned above, the following parameters were considered. In the initial dance motion pair, each dancer has a world position. In this case, for a specific time step t, the leader's global position is represented as p_t^{leader} and the global position of the partner is represented as $p_t^{partner}$. In this case, instead of representing the partner character with a global position, it was represented by its local position in relation to the leader dancer. Therefore, the partner's position is represented as a function of the lead dancer as $p_t^{partner} = f(p_t^{leader})$. Similarly, the partner character's rotation is defined as the function $q_t^{partner} = f(q_t^{leader})$ that aligns the root rotation of the partner $q_t^{partner}$ according to the lead dancer's root rotation q_t^{leader}. Based on this simple alignment process, given the position p_t^{user} and the rotation q_t^{user} of the user (p_t^{user} and q_t^{user} are based on the character controlled by the user) at runtime, the position and the rotation of the partner character (local position and rotation according to user) are estimated as functions of the user's position and rotation, such that $p_t^{partner} = f(p_t^{user})$ for the spatial position and $q_t^{partner} = f(q_t^{user})$ for the root rotation. Such a simple alignment process ensures that the partner character will have a reasonable spatial position and rotation in relation to the user.

5.2 Correcting Hand Contacts

In the initial motion-capture dance performance, the dancers' hands are correctly aligned. Since the user might perform variations of the initially captured motion used as the template, the hand contacts, as well as the right hand's positioning, might be lost. To successfully correct the hands of the partner character, a simple inverse kinematics (IK) approach was used. Given the corresponding motion sequences of the leader and partner characters, the system indicates whether a hand contact exists based on a simple hand contact state. Specifically, the hand states h_t indicates whether the right hand h_{right}^{leader} or the left hand h_{left}^{leader} of the leader is in contact with the corresponding right hand $h_{right}^{partner}$ or left hand $h_{left}^{partner}$ of the partner, such that $h_t = [h_{right}^{leader}, h_{left}^{leader}]$, where $h_{right}^{leader} = [0 \text{ or } 1, h_{right}^{partner} \text{ or } h_{left}^{partner}]$ and $h_{left}^{leader} = [0 \text{ or } 1, h_{right}^{partner} \text{ or } h_{left}^{partner}]$, where 0 indicates no hand contact and 1 indicates that hand contact exists. Predicting whether a hand contact exists was implemented as a simple collision detection technique using bounding boxes like the one used by Peinado et al. [34].

During the application's runtime, the system predicts the partner's motions based on the leader's motions, and since each pose belongs to a state of the HMM, the $t-th$ state of the HMM is used to indicate whether hand contact between the leader and the partner exists based on the initial motion sequences. If there is any hand contact, the virtual partner's hand IK state is enabled, thereby altering the partner's motions to reach the target position of the lead character's hand. The target effectors with which the partner's hand should be aligned are defined manually beforehand, and by default are the leader's wrist joints. Figure 4 shows snapshots of a synthesized dance pose of both characters when there is no IK solver applied as well as when there is one applied to the partner's hands.

To overcome the hand IK issue, a simple yet efficient IK solution proposed by Tolani et al. [41] was integrated into the current implementation. During runtime, when hand state h_t indicates hand contact should be enabled, a suitable arrangement of the joints is calculated given the three partner's joints to solve for (shoulder, elbow, and wrist) and the target end effector (corresponding to the leader's wrist joint). Due to the lack of any checking or optimization based on the resulting arrangement of the IK chain, the swivel parameter, which is responsible for controlling the resulting rotation of the elbow in the resulting chain, is set to a default and maintained throughout the calculations.Therefore, the system corrects the necessary hand contacts and the interaction between the user and the virtual character becomes more plausible.

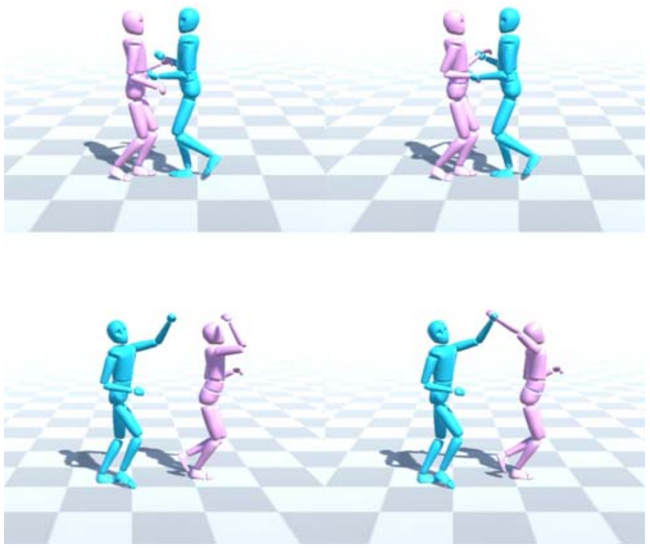
Figure 4: Leader's and partner's hands without IK adjustment (left) and with IK adjustment (right).

5.3 Motion Smoothness

Three different adjustments were implemented to ensure that the synthesized motion of the virtual partner is smooth and natural. Firstly, since the partner's motion is synthesized independently in every frame, a commonly used Laplacian smoothing algorithm for each angle value of the character's joints using the SLERP [37] approach in a quaternion space was used to ensure temporal coherence. Additionally, to avoid foot skating artifacts, a simple foot skating removal algorithm was applied [20]. Specifically, the algorithm adjusts root translations by detecting when a foot is close to the ground and its velocity is below a specified threshold. Due to the jump state transition, an adjustment to ensure smooth motion was also applied. A simple motion blending algorithm was used to avoid synthesizing motion that appears unnatural. Specifically, a velocity-based blending algorithm [25] was used to interpolate the corresponding predicted motion in the developed system. With this algorithm, the angular velocity's magnitude in the blending process is maintained equal to the target animation on each joint. Then, the joints' rotation is adjusted to be a more accurate representation of the character's target pose (predicted pose).

6 Implementation Details

The application was developed in Unity3D v5.5. The virtual reality hardware used was the Oculus Development Kit v2. The Oculus Utilities for Unity 5 package for Oculus integration into Unity3D was used. The application consisted of one scene and a simple menu that assigns the necessary variables (character, leader and partner dance motion, and user role). For the motion capture process, two different motion capture systems were tested. Firstly, the 18 IMU setup of the Perception Neuron [32] motion capture solution was used. Each IMU houses a gyroscope, accelerometer, and magnetometer. Even though the accuracy of this solution is ideal for this kind of application since the data can be transmitted wirelessly to the computer, it was found that a lower quality sensor can also produce the necessary results, and can also save time (especially useful for the user study presented in the next section). Therefore, the Kinect Sensor for Xbox One [26] was also tested and used for basic motion capture during the user study. The Kinect v2 sensor has about a 70°

horizontal field of view and can cover 4.5 meters in depth reliably. For the Kinect integration, Kinect for Windows SDK v2.0 was used along with Kinect v2 and an MS-SDK Unity package [13]. Figure 5 illustrates a user wearing the necessary equipment and Figure 6 shows a user dancing with a virtual partner using the presented method. Additional results are presented in the video accompanying this paper.

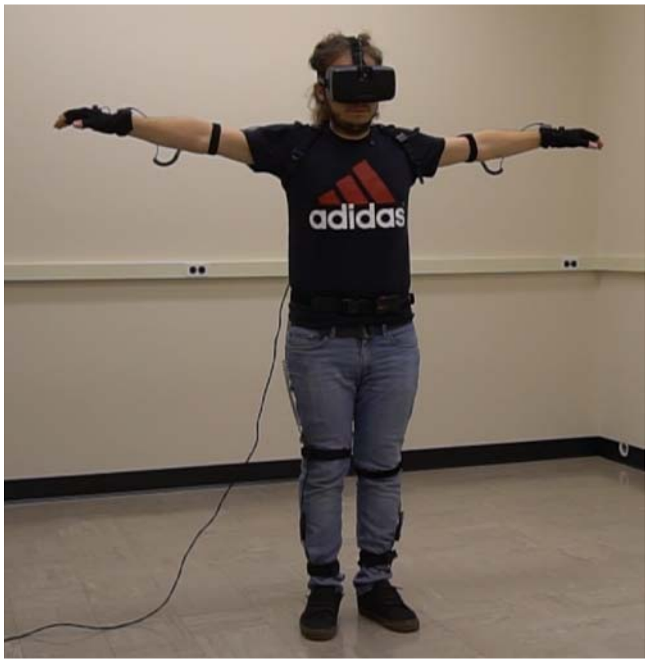

Figure 5: A user wearing the motion capture suite and the virtual reality HMD device.

7 User Study

Evaluating the properties of the motion refinements of the developed system is critical to understand the ways users perceive the interaction with a virtual partner when they dance. Therefore, a user study was conducted to examine the naturalness of the virtual partner's synthesized motion as well as the how users perceive the control they have over the partner's motion.

In this study, 36 participants took part (19 males, aged $M = 23.34$, $SD = 1.16$ and 17 females, aged $M = 23.79$, $SD = 1.35$). Among the participants, 23 had at least one prior experience with virtual reality. Participants were recruited through posters on campus, e-mail, and in-class announcements. All participants were volunteers, and there was no reward given for participation. The user study took no more than 60 minutes per volunteer, and all participants completed the study.

Participants came to the lab where the necessary equipment was located, and the experimenter informed them about the project. Participants were also briefly introduced to the motion capture system and to the virtual reality headset, and the experimenter informed them what they should pay attention to while they are experiencing it. The experimenter asked them to wear the HMD to walk through a path in a virtual world to ensure they feel comfortable wearing the HMD while walking. The experimenter also recommended that they watch the dance motion that they would be asked to dance in an interactive application where the participants were able to change the camera view as well as zoom in on different body parts of the dancing character. Then, the experimenter asked them to train themselves by performing the dance movements without any equipment to ensure that they know the exact movements of the dance. Moreover, the experimenter informed them that they can also improvise the performance by weaving together different parts of the dance. To avoid confusion as well as avoid asking the participants to memorize dozens of movements, the chosen dance performance had a 15-second duration. Each participant was assigned to be the lead dancer regardless of gender since it was decided that every participant should experience the exact same dance performance.

The conducted user study is based on a set of three questions (see Table 1) related to two different aspects of the user experience. Question 1 (Q1) ascertains the naturalness of the partner's motions when the smoothing factors (see Section 5.3) are applied or not applied. Questions 2 (Q2) and 3 (Q3) determine how the users perceived the degree of control they had on the virtual character's synthesized motions. Participants were required to give their feedback using a 7-point Likert scale. Based on the questionnaire, four different variations (two for Q1, and two for Q2 and Q3) of the experimental scenario were developed. In the first variation, the motion smoothness is omitted and in the second, the motion smoothness is applied. In the third variation, the hand contact functionality (see Section 5.2) is enabled (for the applicable parts of the dance motions), and for the fourth, the hand contact is disabled throughout the dance performance.

To cover all four variations, the experiment had the following structure. The participant was first introduced to the virtual environment and to the virtual partner. Then, the participant had to perform a couple of steps to reach a signed goal position in the virtual space. It should be noted that the user was observing the environment from a first-person viewpoint (the virtual camera was attached to the lead character's head, see Figure 6). Then, a randomly chosen variation of the motion smoothness was shown and the user was asked to dance with the virtual partner for 3 minutes. Then, the experimenter asked Q1 and the process repeated with the other motion smoothness variation. Again, the participant was asked to reach the signed position in the virtual space, dance with the virtual partner, and respond to Q1. The process then continued with the motion control part. As before, one variation (hand contact enabled or disabled) was chosen randomly. After asking the user to reach the goal position and interact with the virtual partner character by dancing with it for 3 minutes, the experimenter asked Q2 and Q3. Finally, the process continued with the last iteration, in which the remaining hand contact variation was chosen, and the participant responded to Q2 and Q3 again. It should be noted that during Q1 the hand contact functionality was enabled, and during Q2 and Q3 the motion smoothness was applied.

For Q1, the results show that the displayed motion achieved higher naturalness scores when the smoothness adjustments were applied ($M = 5.83$, $SD = 0.44$) than when the smoothness adjustments were not applied ($M = 1.55$, $SD = 0.93$). A Wilcoxon signed-rank test showed that there is a significant decrease between the displayed motion scores when no adjustments applied to the virtual character ($W = 712.5$; $z = -2.3129$; $p = 0.0203$), which means the adjustments were effective and enhanced the realism of the motion of the virtual partner. For Q2, the participants indicated that the partner character followed their performance more accurately when hand contact was enabled ($M = 6.73$, $SD = 1.09$) than when hand contact was disabled ($M = 2.11$, $SD = 0.48$). A Wilcoxon signed-rank test supported this finding by showing there is a significant decrease in the participants' perception of the virtual partner's ability to follow users movements when the hand contact is disabled ($W = 617$; $z = -2.41$; $p = 0.0229$). For Q3, the participants said that the partner character is forced to follow their dance performance more strictly when hand contact is enabled ($M = 5.62$, $SD = 0.37$) than when hand contact is disabled ($M = 3.31$, $SD = 0.85$). Again, a Wilcoxon signed-rank test confirmed there is a significant decrease in the responses of users when the hand contact is disabled ($W = 723$; $z = -2.329$; $p = 0.0189$). Based on Q2 and Q3, it appears the hand

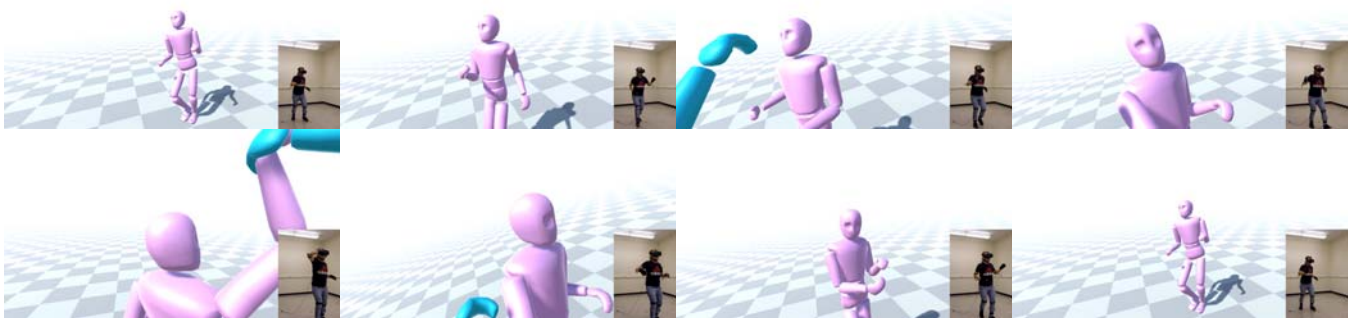

Figure 6: A user dancing with a virtual partner while wearing the necessary equipment. The visual output is based on the user's point of view.

Table 1: The questionnaire that was used to evaluate the properties of the virtual partner's dance motion control system.

No.	Question	Anchors of the Scale	Variation
1	Does the virtual partner dance in a natural-looking way?	1 indicates not all, 7 indicates totally.	Motion synthesized either with or without motion smoothness.
2	Does the virtual partner follow your dance performance properly?	1 indicates not all, 7 indicates totally.	Hand contact is either enabled or disabled.
3	Do you feel that you force the virtual partner to dance in a specific way?	1 indicates not all, 7 indicates totally.	Hand contact is either enabled or disabled.

contact adjustments implemented in this application were beneficial to the users since they reported the virtual partner followed their movements more appropriately and that they had better control over the virtual partner's motion while using hand contact.

8 Conclusions and Future Work

In this paper, a method for controlling a virtual partner's motion based on a performance capture process of a user was presented. Specifically, the paper examined a virtual dance motion control method for salsa dance performances, giving users the ability to interact with a virtual partner when dancing with him or her. The developed method uses a publicly available dataset containing salsa dance motion sequences including the motions of both the leader and the partner performer. To achieve a real-time prediction of the user's motion and consequently the control of the virtual partner's motion, a method based on HMM was developed. Specifically, the HMM was trained using motion data of one of the dancers, and the forward algorithm was used to handle the system's real-time prediction and motion-follow behavior. Since the user should be allowed to engage in freeform dancing, a jump transition was introduced, allowing the model to jump from its current state to another state within the dance performance, thus letting the user weave together different parts of the dance motion sequence. Considering that long motion sequences might be used, an optimization of the forward algorithm was introduced, decreasing the recognition process' complexity. In addition to the prediction of the partner's pose based on the user's input pose, a few additional adjustments were also implemented. Firstly, the spatial alignment of the partner character's motion was corrected. Secondly, an adjustment to the partner's hands was required to ensure the validity of the dance performance as well as to enhance its realism. Finally, the motion data was filtered to ensure the newly synthesized motion is smooth and eliminate artifacts introduced by the jump state transitions.

Along with the development of the presented method, a user study was also conducted. The first part of the user study dealt with the way users perceive the virtual partner's synthesized motion (whether it is natural-looking or not). The second part dealt with the way users perceive the control they have over the virtual partner's synthesized motion when applying (or not) the hand contact functionality. The results indicated that users perceive the synthesized motion as more natural when applying the smoothness factors; they also indicated that the hand contact is beneficial since the feeling of controlling the partner properly is enhanced.

The current implementation considers and solves a few issues related to performance-driven user-character interaction. Undoubtedly, there are other performance-driven user-character interaction scenarios that the presented method can be applied to, but adjustments specific to the interaction scenario are likely to be necessary. Examples include close interactions while fighting as well as social interaction such as hugging. In such scenarios, spatial relations [1] and deformations [43] between the 3D meshes, as well as motion constraints [5] need to be considered. Furthermore, to cope with the previously mentioned additions, issues regarding runtime efficiency should also be kept in mind. Considering the growth of virtual reality applications and scenarios in which users are called to interact closely with virtual characters, further development of powerful and flexible approaches that could be adapted for close interaction scenarios between users and virtual characters is necessary. This is an issue that should be addressed in future work.

References

[1] R. A. Al-Asqhar, T. Komura, and M. G. Choi. Relationship descriptors for interactive motion adaptation. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 45–53. ACM, 2013.

[2] H. Alizadeh, A. Witcraft, A. Tang, and E. Sharlin. Happyfeet: Embodiments for joint remote dancing. In *Graphics Interface*, pp. 117–124, 2016.

[3] A. Aristidou, E. Stavrakis, P. Charalambous, Y. Chrysanthou, and S. L. Himona. Folk dance evaluation using laban movement analysis. *Journal on Computing and Cultural Heritage*, 8(4):Article No. 20, 2015.

[4] S. Babu, S. Schmugge, R. Inugala, S. Rao, T. Barnes, and L. Hodges. Marve: a prototype virtual human interface framework for studying human-virtual human interaction. In *Intelligent Virtual Agents*, pp. 120–133. Springer, 2005.

[5] B. L. Callennec and R. Boulic. Interactive motion deformation with prioritized constraints. *Graphical Models*, 68(2):175–193, 2006.

[6] P. Cano, A. Loscos, and J. Bonada. Score-performance matching using hmms. In *International Computer Music Conference*, 1999.

[7] Carnegie Mellon University, Graphics Lab. Motion Capture Database from `http://mocap.cs.cmu.edu/`, 2017.

[8] J. C. Chan, H. Leung, J. K. Tang, and T. Komura. A virtual reality dance training system using motion capture technology. *IEEE Transactions on Learning Technologies*, 4(2):187–195, 2011.

[9] P. T. Chua, R. Crivella, B. Daly, N. Hu, R. Schaaf, D. Ventura, T. Camill, J. Hodgins, and R. Pausch. Training for physical tasks in virtual environments: Tai chi. In *Virtual Reality, 2003. Proceedings. IEEE*, pp. 87–94. IEEE, 2003.

[10] Y. Cui and C. Mousas. Master of puppets: An animation-by-demonstration computer puppetry authoring framework. *3D Research*, 9(1):Article No. 5, 2018.

[11] L. Deng, H. Leung, N. Gu, and Y. Yang. Real-time mocap dance recognition for an interactive dancing game. *Computer Animation and Virtual Worlds*, 22(2-3):229–237, 2011.

[12] R. Fan, S. Xu, and W. Geng. Example-based automatic music-driven conventional dance motion synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):501–515, 2012.

[13] R. Filikov. Kinect v2 Asset from `https://www.assetstore.unity3d.com/en/#!/content/61376`, 2017.

[14] M. Gillies, X. Pan, and M. Slater. Piavca: a framework for heterogeneous interactions with virtual characters. *Virtual reality*, 14(4):221–228, 2010.

[15] K. Hachimura, H. Kato, and H. Tamura. A prototype dance training support system with motion capture and mixed reality technologies. In *IEEE International Workshop on Robot and Human Interactive Communication*, pp. 217–222. IEEE, September 2004.

[16] E. S. Ho, J. C. Chan, T. Komura, and H. Leung. Interactive partner control in close interactions for real-time applications. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 9(3):Article No. 21, 2013.

[17] E. S. L. Ho, T. Komura, and C.-L. Tai. Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics (TOG)*, 29(4):33, 2010.

[18] N. Jaksic, P. Branco, P. Stephenson, and L. M. Encarnação. The effectiveness of social agents in reducing user frustration. In *CHI'06 extended abstracts on Human factors in computing systems*, pp. 917–922. ACM, 2006.

[19] J. W. Kim, H. Fouad, J. L. Sibert, and J. K. Hahn. Perceptually motivated automatic dance motion generation for music. *Computer Animation and Virtual Worlds*, 20(23):375–384, 2009.

[20] M. Kim, K. Hyun, J. Kim, and J. Lee. Synchronized multi-character motion editing. *ACM Transactions on Graphics*, 28(3):79, 2009.

[21] T. Komura, B. Lam, R. Lau, and H. Leung. e-learning martial arts. *Advances in Web Based Learning–ICWL 2006*, pp. 239–248, 2006.

[22] M. Kyan, G. Sun, H. Li, L. Zhong, P. Muneesawang, N. Dong, B. Elder, and L. Guan. An approach to ballet dance training through ms kinect and visualization in a cave virtual reality environment. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(2):23, 2015.

[23] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, 2002.

[24] J. Lee and K. H. Lee. Precomputing avatar behavior from human motion data. *Graphical Models*, 68(2):158–174, 2006.

[25] S. Levine, C. Theobalt, and V. Koltun. Real-time prosody-driven synthesis of body language. *ACM Transactions on Graphics*, 28(5):172, 2009.

[26] Microsoft. Kinect SDK from `https://www.microsoft.com/en-us/kinectforwindows/`, 2017.

[27] C. Mousas. Full-body locomotion reconstruction of virtual characters using a single inertial measurement unit. *Sensors*, 17(11):Article No. 2589, 2017.

[28] C. Mousas and C.-N. Anagnostopoulos. Performance-driven hybrid full-body character control for navigation and interaction in virtual environments. *3D Research*, 8(2):18, 2017.

[29] C. Mousas and C.-N. Anagnostopoulos. Real-time performance-driven finger motion synthesis. *Computers & Graphics*, 65:1–11, 2017.

[30] A. Nakamura, S. Tabata, T. Ueda, S. Kiyofuji, and Y. Kuno. Dance training system with active vibro-devices and a mobile image display. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3075–3080. IEEE, August 2005.

[31] T. Nakamura, E. Nakamura, and S. Sagayama. Acoustic score following to musical performance with errors and arbitrary repeats and skips for automatic accompaniment. In *Sound and Music Computing*, pp. 299–304, 2013.

[32] NeuroMoCap. Perception Neuron from `https://neuronmocap.com/`, 2017.

[33] C. Ouzounis, C. Mousas, C.-N. Anagnostopoulos, and P. Newbury. Using personalized finger gestures for navigating virtual characters. In *Virtual Reality Interaction and Physical Simulation*, pp. 5–14, 2015.

[34] M. Peinado, D. Maupu, D. Raunhardt, D. Meziat, D. Thalmann, and R. Boulic. Full-body avatar control with environment awareness. *IEEE Computer Graphics and Applications*, 29(3), 2009.

[35] A. Raftis. *The world of Greek dance*. Finedawn Publishers, 1987.

[36] Y. Sakai, T. Takeda, Y. Hirata, and K. Kosuge. Collision avoidance based on estimated step of other dance couples for male-type dance partner robot. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3264–3269. IEEE, 2007.

[37] K. Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, 19(3):245–254, 1985.

[38] T. Takeda, Y. Hirata, and K. Kosuge. Dance partner robot cooperative motion generation with adjustable length of dance step stride based on physical interaction. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3258–3263. IEEE, 2007.

[39] T. Takeda, Y. Hirata, and K. Kosuge. Dance step estimation method based on hmm for dance partner robot. *IEEE Transactions on Industrial Electronics*, 54(2):699–706, 2007.

[40] T. Takeda, K. Kosuge, and Y. Hirata. Hmm-based dance step estimation for dance partner robot-ms dancer. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3245–3250. IEEE, 2005.

[41] D. Tolani, A. Goswami, and N. I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[42] Z. Yang, B. Yu, W. Wu, R. Diankov, and R. Bajscy. Collaborative dancing in tele-immersive environment. In *ACM International Conference on Multimedia*, pp. 723–726, October 2006.

[43] K. Zhou, W. Xu, Y. Tong, and M. Desbrun. Deformation transfer to multi-component objects. *Computer Graphics Forum*, 29(2):319–325, 2010.