Generating a Ball Sport Scene in a Virtual Environment

Jongin Choi[1], Sookyun Kim[2], Sunjeong Kim[3] and Shinjin Kang[4*]
[1] Department of Digital Media Design and Applications, Seoul Women's University
621 Hwarangro, Nowon-Gu, Seoul, Korea
[e-mail: funtech@swu.ac.kr]
[2] Department of Game Engeenering, Paichai University
155-40 Baejae-ro, Seo-Gu, Daejeon, Korea
[e-mail: nicesk@gmail.com]
[3] Department of Convergence Software, Hallym University
1 Hallymdaehak-gil, Chuncheon-si, Gangwon-do, Korea
[e-mail: sunkim@hallym.ac.kr]
[4] School of Games, Hongik University
2639 Sejong-ro, Jochiwon, Sejong, Korea
[e-mail: directx@hongik.ac.kr]
*Corresponding author: Shinjin Kang

Received January 7, 2019; revised February 25, 2019; accepted April 27, 2019; published November 30, 2019

Abstract

In sports video games, especially ball games, motion capture techniques are used to reproduce the ball-driven performances. The amount of motion data needed to create different situations in which athletes exchange balls is bound to increase exponentially with resolution. This paper proposes how avatars in virtual worlds can not only imitate professional athletes in ball games, but also create and edit their actions effectively. First, various ball-handling movements are recorded using motion sensors. We do not really have to control an actual ball; imitating the motions is enough. Next, motion is created by specifying what to pass the ball through, and then making motion to handle the ball in front of the motion sensor. The ball's occupant then passes the ball to the user-specified target through a motion that imitates the user's, and the process is repeated. The method proposed can be used as a convenient user interface for motion based games for players who handle balls.

Keywords: Virtual character, Human computer interaction, Natural user interface, Physics – based optimization, Motion synthesis

This work was supported by a research grant from Seoul Women's University(2019)

http://doi.org/10.3837/tiis.2019.11.013 **ISSN : 1976-7277**

1. Introduction

There is an increase in demand for hands-on user interactions through virtual reality technology, especially in games like VR ping-pong, where the generation of realistic real-time character animation in response to the fast movements of the ball is still an important issue. The technique of character animations has been developed to utilize various motion control techniques and create a more realistic and natural movement, which has traditionally relied on a key frame technique. Particularly, the natural behavior of the characters appearing and interacting with on-screen objects are critical factors that determine the quality of the media contents. In the case of virtual reality contents, there has been an increased interest in the necessity to generate adaptive motions for the interactions between characters and virtual environments. Motion capture equipment is commonly used for character motion generation in virtual reality. The motion capture device can analyze the user's motion in real time and instantly reflect it in the virtual world. It is possible to reflect the movement of the user as an action of the character in the virtual world as it is, or to recognize the hand motion of the user and manipulate the virtual object. Therefore, the motion capture system provides an easy solution for controlling the operation of the virtual character by allowing the user to instantly generate a real-time animation. This technique involves converting the angle of the body joint into the angle and position data on the time axis. This is very natural as it extracts data directly from human motions; however, the execution of this technique still requires an expensive system, especially to create motions of non-realistic objects. In other words, it has the advantage of being able to apply the actual movement of a user to the animation as it is. However, it is necessary to have a skilled actor to express a specific movement such as ball kicking or a waltzing interaction with another character.

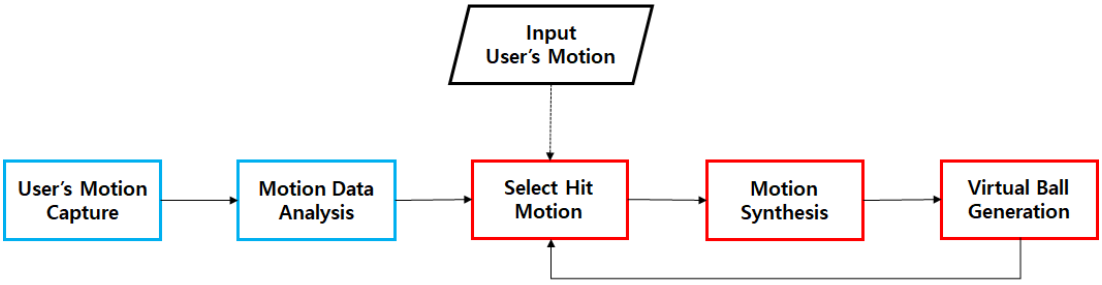


Fig. 1. System Overview.

In this paper, we propose a simple but interesting virtual avatar motion synthesis method that considers physical interaction, so that users can handle the ball naturally, even in the virtual world. In the real world, dealing with balls requires a high level of work objectives. Therefore, it takes considerable time and effort to become a skilled player, especially while manipulating a virtual ball using only a motion capture device; it would be more difficult than manipulating one in the real world. **Fig. 2** shows an overview of our system. Blue and red boxes mean preprocessing and run-time processing. It involves four main steps. The first step generates interaction information by analyzing the user's motion. The second step searches motion sequences from the first step. The motion of a user's character is synthesized by connecting motion sequences in the third step. At this time, the motion sequences that closely resemble the user's motion, as collected by the motion sensor, are selected in real-time. The

final step generates (in real time) the movement paths for objects that are synchronized with the synthesized motion of the user's character while obeying the laws of physics. To confirm the effect, we use our system to create a scene where the user can act as a well-trained football player so they can exchange balls with other virtual characters.

Creating a sports scene (such as football) where athletes play together requires skilled actors and expensive motion capture equipment. To shoot sports scenes according to the desired scenarios, it is necessary to capture the motions of the actors for a relatively long period of time through trial and error. In this study, it is possible to create such scenes by using one user's motions, to generate a desired scene relatively quickly. In addition, one user can capture the required motions in a small space, so cost and location problems are avoided as well. As the method of this study can be used as a real time interface for controlling a character, it can be utilized as a means for intuitively controlling a character's motion in a motion-based game.

Hyun et al. [1] created scenes in which a number of characters played a basketball game using a sketch-based interface. It is possible to control at a high level, but fine control at a low level is not easy and only plays according to a predefined scenario. In our study, the level of character motion can be controlled by a user through the motion sensor, and it is possible to instantly change the scenario of a game while watching the generated game scenes. Lee et al. [2] created a data-based character that uses objects to play through the recurrent neural network. However, they must manually specify the information about the dribble point of the ball and the motion to be used by the character. Our method automatically detects the frame in which the character and the ball interact, finds the motion most similar to the user's motion, and automatically applies it to the character, so we can intuitively control the character's motion. Liu et al. [3] created a physically based character dribbling a basketball using deep reinforcement learning and object trajectory optimization. However, when the character touches the ball, it appears somewhat awkward because the exact control of the ball is difficult. Our method uses the interaction information between the pre-calculated ball and the character, so it is possible to create a natural-looking animation. Furthermore, if the quality of motion data is high enough, a realistic animation can be created.

2. Related Work

Designing the motions of virtual human character that actively responds to the physical environment is an established challenge in computer animation. From a research point of view, there are various examples of research areas on character motion generation. For instance, if motion capture technology is operated at a low cost, things such as the creation of new motion from acquired motion data, natural connection and synthesis between motion, physical description, and simulation become increasingly difficult. Among these, it can be said that the field that controls the character to physically describe and automatically generate a specific action, or to keep the movement of the character from the external input. Recently, motion synthesis studies are taking place considering the environment or an interaction between the virtual characters and objects.

Our research involves bringing together example-based character animation and physics-based rigid-body animation. We generate the movement path of an object from motion capture data and synthesize the motion of an avatar to handle several objects. The motion of the avatar is generated using the user's input motion from the motion sensor. Jain et al. [4] proposed a method for the manual editing of the interaction between the movement of objects and character motion. The movement path of the object and character poses can be edited. The movement path can be modified by checking the collisions between the object and the character. As the user edits

the interaction manually, this method can offer detailed control, but it requires a lot of time and effort. In our method, we do not correct the motion of the character at all. Instead, we generate movement paths for the objects that are consistent with the motion of the character.

Choi et al. [5,6] generated various actions of a character controlling the movement of a ball, but could not generate and control the motion of the character handling the ball in real time. In addition, the motion of the juggling character was created in real time, but it was applied only to a single character performing the said motion. In this study, we provide an intuitive method for a user to create and control a scene in which a number of characters can simultaneously play a game in real time.

We extracted the information required to generate the movement path of an object by analyzing motion data. Analyzing motion capture data has been used in research to help adapt the motion of a character to the music composed. Kim et al. [7] extracted common patterns from the motion of a person and generated a character to dance along to the rhythm of the music. Shiratori et al. [8] categorized the motions of a person together with the strength of the beat to synthesize a dancing motion that matched the input music. We analyze motion capture data to generate the movement path of an object. Popovic' et al. [9,10] proposed a method for generating the movement path of a rigid body consistent with the laws of physics. They generated natural moving objects by calculating the initial position and velocity of the rigid body through optimizing the movement path under several constraints. We generate the movement path of an object to be synchronized with the ball controlling motion both spatially and temporally by using their method.

A variety of methods for controlling the motion of a character through a user's input has been researched. Kovar et al. [11] and Lee et al. [12] generated a connected motion graph and synthesized an animated character that moved in a direction and toward a goal position that was specified by the user. Heck et al. [13] enabled a more detailed user control by the effective mixing of similar motions. For example, the character can arrive at or punch a goal position accurately. Gleicher [14] edited the movement path of a character and had the character move along it. Gleicher et al. [15] enabled the character to be controlled easily by generating the connection information for common poses. For example, their method could be used for the motion of returning to an idle state after executing the kicking and punching commands in a fighting game. Thorne et al. [16] and Martin et al. [17] had the movement of a character controlled in detail via sketch-based user input. The user could draw various movements and moving paths, such as normal walking, sneaking, marching, and jumping, with an electronic pen. In [18,19], a character was created that interacted with the user input more rapidly and effectively by using reinforcement learning. Levine et al. [20] made a character move and reach a goal effectively in a user-specified dynamic environment. We specified the character's motion by the user's motion via a motion sensor.

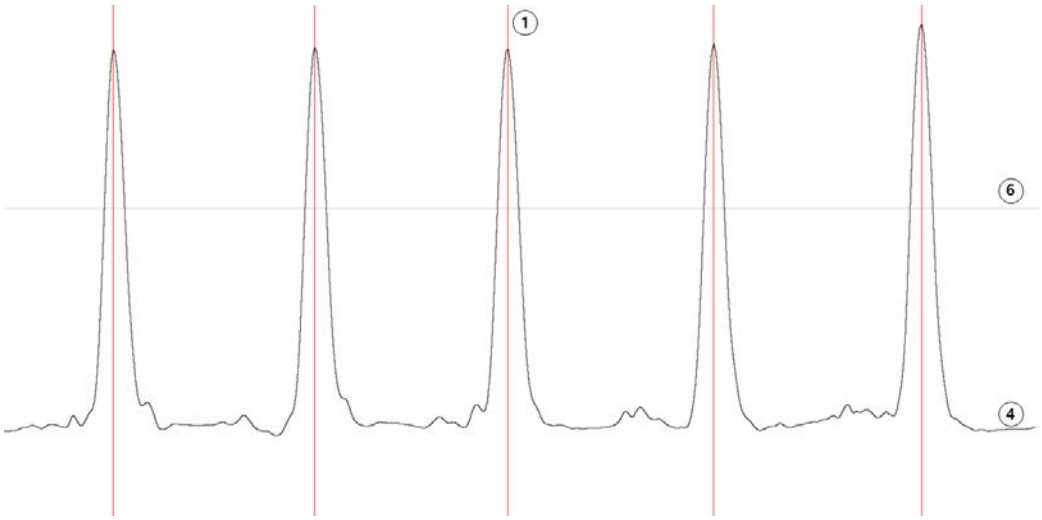
In the game industry, we can see the fact that started to pay attention not only to rendering quality but also to character animation quality. It is a typical example of the movements utilized in the Uncharted and Assassin's Creed series created by Naughty Dog and Ubisoft respectively. These companies have moved away from the simple interpolation used for motion connection and created more realistic motion of the character using procedural or parametric techniques for motion generation. The Hitman series, created by IO Interactive, introduced the latest techniques in character animation in an attempt [21] to implement realistic crowd simulations. Although it was not applied to the main character because of the relatively slow reaction times, it produced very satisfactory results. Recently, Ubisoft used the concept of [11] and [22] for making motion matching [23], which creates continuous connection of characters smoothly and continuously. In this paper, we have also created a virtual player animation using the method of Simon Clavet [23].

Recently, many researches have been made to generate a character movement using artificial intelligence. Holden et al. [24] created a data-based character that can be controlled in real time using phase-functioned neural networks. Peng et al. [25] used the deep reinforcement learning to create a bipedal physics-based character that naturally follows on the terrain. Peng et al. [26] used deep reinforcement learning to create physics-based characters that follow human skills. Yu et al. [27] created a physics-based character that learns how to move by itself through reinforcement learning and energy optimization. Zhang et al. [28] used the Mode-Adaptive Neural Networks to generate natural locomotion motions of quadrupedal animals. There have also been researches that use artificial intelligence to generate animations of interaction between characters and objects. Liu et al. [29] used Deep Q-Learning to climb on an object and create a physics-based character that balances and moves on the object. Chemin et al. [30] generated physically based characters that juggle using Reinforcement Learning.

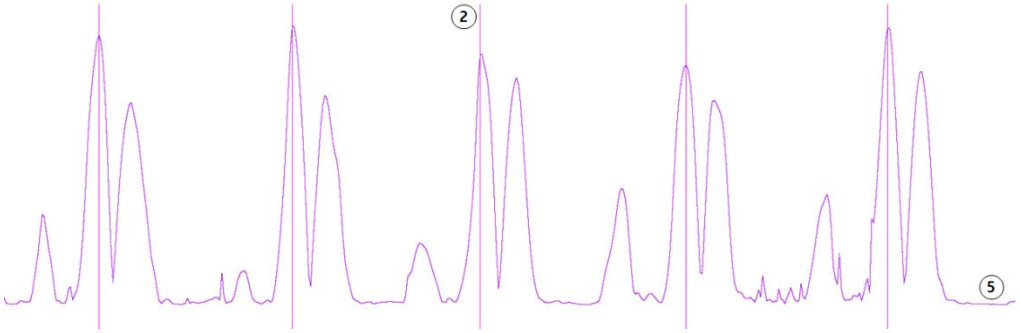
3. Sports Scene Generation

We use motion capture data to create a motion where the character controls the ball. To enable one to easily create a character's motion, we use a motion that mimics the motion of the video without actually controlling the ball. We capture the motion generated by three movements for foot volleyball: toss the ball to the nearest opponent, pass the ball to an opponent far away in a medium-distance passing motion, and strike the ball with an attacking motion to make it difficult for the opponent to obtain the ball. Three right-foot and three left-foot motions are generated. To cover all the directions, we rotate the motion at intervals of 45° to create a motion that hits the ball. In other words, the player can hit at −135°, −90°, −45°, 0°, 45°, 90°, 135°, and 180° with respect to the forward direction so that the character can receive and pass the ball in any direction.

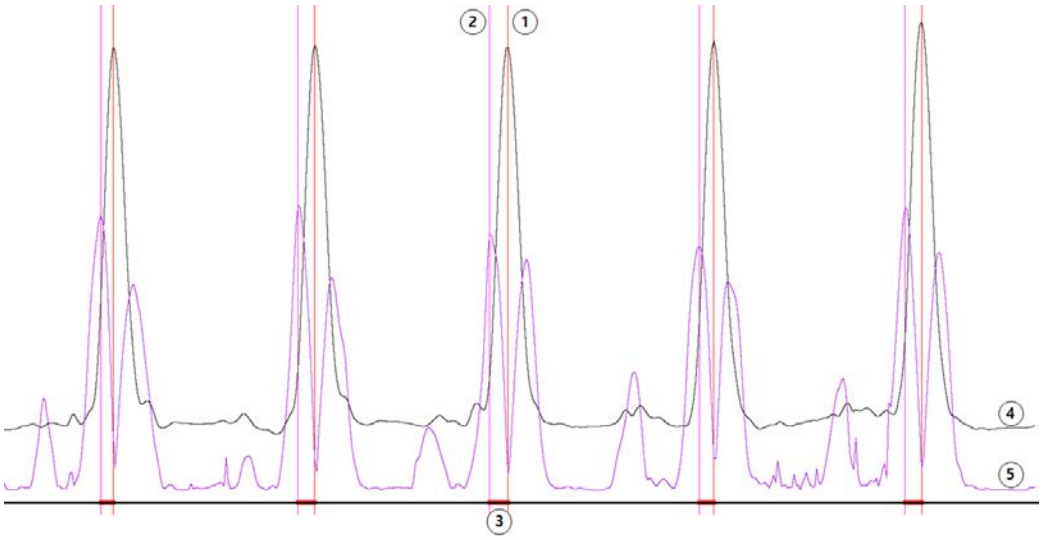
In this study, as the motion of a character is determined using the user's motion, we analyze the captured motion and search for the section (S_B) in which a player can hit the virtual ball. **Fig. 2** shows how to find S_B. First, we determine the maximum height frame (F_H), which is the frame where the hit bone is higher than in the neighboring frames. To determine F_H, we check the height of the hit bone in every frame of motion. The user specifies the threshold height, and we search for all the frames in which the hit bone is higher than this threshold. **Fig. 2(a)** shows all the F_H values found in the location graph of the hit bone. The red vertical line denotes F_H. Next, we determine all the maximum speed frames (F_S) that are faster than the neighboring frames. To determine F_S, we check the speed of the hit bone in every frame of the motion data. Note that F_S may appear to be very close to one another, particularly when the hit bone returns to its original position after hitting the ball. To effectively prevent this issue, only one frame with the faster speed is selected from the two neighboring F_S.



(a) Maximum height frames in height graph of a hit bone



(b) Maximum speed frames in speed graph of a hit bone



(c) Hit sections which are searched from (a) and (b)

Fig. 2. Motion analysis with graphs. 1:F_H. 2: F_S. 3: S_B. 4: Height graph of a hit bone. 5: Speed graph of a hit bone. 6: Threshold height.

Fig. 2(b) shows all the F_S found from the velocity graph of the hit bone. The magenta vertical line indicates F_S. The section from the neighboring F_S to the F_H is the S_B in which a player can hit the ball. **Fig. 2(c)** shows all the S_B determined using the position and velocity graph of the hit bone. At the bottom of the graph, the thick red horizontal line represents S_B.

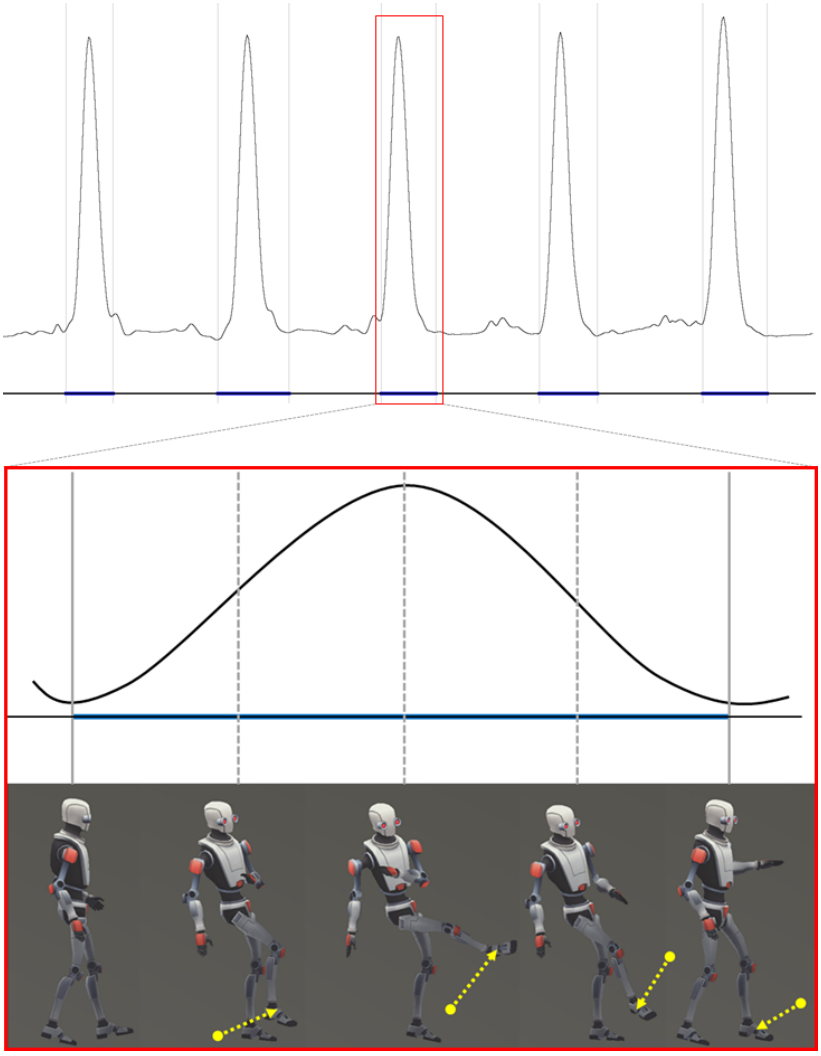


Fig. 3. Character poses in the hit motion sequence.

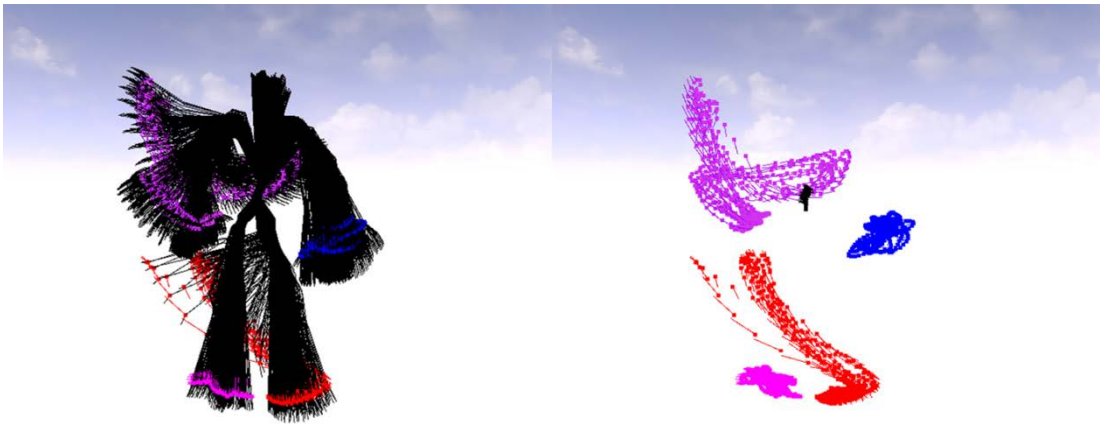
We classify the motion of a character into two broad categories. One is the motion of hitting the ball, and the other is the idle motion. We use the position graph of the hit bone to determine the hit motion section (S_H), which is the section of motion in which a player hits the ball. **Fig. 3** shows how to determine S_H. Top figure shows the result of extracting the section of the hit motion from the height graph of the hit bone. Bottom figure shows five representative poses that constitute the hit motion. Yellow arrow lines denote the movement vectors of the hit bone. The top graph is the position graph of the hit bone, and the thick blue line indicates S_H. In the graph above, the red box indicates one S_H, and the bottom figure shows its enlarged version. In the position graph of the hit bone, if we find the frame with the lowest position of the hit bone backward and forward with respect to the maximum height frame (F_H), it becomes S_H. The

figure at the bottom shows the main poses of the character appearing in the S_H. The first and the last poses show the character standing before and after hitting the ball. The second and the fourth are the poses of the character in the maximum speed frame (F_S). The third is the pose of the character in the maximum height frame (F_H). The dotted yellow line on the character's foot indicates the movement vector of the hit bone. This is used to determine the motion that will be the most similar to the user's motion input from a motion sensor in the future. All the sections except the hit motion section (S_H) are used as the link motion sections (S_L) for motion connection. That is, S_L is a section between two neighboring S_H. We determine the optimal match frames where the pose of the character is the most similar in two different S_H and connect the motion.

4. Interactive Motion Synthesis

In this study, we use the user's motion to create animations that deal with the balls between different characters in real time. We use the link motion section (S_L) to create the seamless motion of a character. In other words, the motion of the character jumps from one S_L to another S_L to constantly connect the motion sequence. To achieve this, we determine the most natural transition section (S_T) that connects two different S_L (S_{Li}, S_{Lj}). We then determine an optimal S_T by using a method similar to motion matching [23]. If the user specifies the length of S_T, the S_T is automatically determined by comparing the poses between S_{Li} and S_{Lj}. In other words, S_T is the interval in which the pose of the character is the most similar between S_{Li} and S_{Lj}. **Fig. 3** shows the poses of the characters aligned in the X-axis direction for the pose comparison and the position and speed of the main bones. Fast calculation is possible because the position and the velocity of all the bones are calculated in advance. Equation 1 is used for the comparison between the two poses. We pre-compute the ST for all the (S_{Li}, S_{Lj}) pairs and link the motion sequences in real time. Here, D_p represents a difference value between the character pauses in each frame. p_i and v_i denote the position and the velocity of the i-th bone, respectively. ω_i^p and ω_i^v are the weights for the position and the velocity of the i-th bone, respectively.

$$D_p = \sum_{i=0}^{n} \omega_i^p \|p_i - p'_i\| + \sum_{i=0}^{n} \omega_i^v \|v_i - v'_i\| \quad (1)$$



(a) Aligned character poses. (b) Positions and velocities of main bones.

Fig. 4. Aligned pose match information

Fig. 4 shows how to connect the motion sequences continuously in S_T. Black lines denote the character poses in (a). Red and magenta represent the foot bones, violet and blue indicate the hand bones, and black denotes a root bone in (b). All the link motion sections (S_L) can be connected to each other. To determine which S_L to jump to, we use the hit motion section (S_H). When the user carries out a motion that deals with the ball in front of the motion sensor, it determines the S_H having the motion that is the most similar to the motion of the user. To do this, S_H is extracted from the user's motion in real time, and the S_H that is the most similar to the user's motion is selected. To find the S_H that is the most similar to the user's motion, we use the motion vector, which denotes the movement of the hit bone in the S_H. The motion vector is the arrow marked with a yellow dotted line at the bottom of **Fig. 3**. For a fast comparison, we pre-computed the motion vector in all S_H. The S_H having the motion vector that is the most similar to the motion vector extracted from the user's motion is searched in real time. Then, we connect the motion to S_L that is connected to the corresponding S_H for the motion connection. Thus, the motion of the character similar to the user motion is generated. We use Equation 2 to calculate the difference between the two motion vectors. Here, D_m denotes the difference between the user's motion vector and the motion vector stored in the motion database. v_{max}^m represents the maximum velocity of the hit bone in the interval of the motion clip. Further, m_i and t_i denote the motion vectors of the i-th motion vector of the hit bone and the motion vector, respectively. ω_v , ω_m, and ω_t represent the weights for the maximum velocity, motion vector, and travel time of the hit bone, respectively.

$$D_m = \omega_v v_{max}^m + \omega_m \sum_{i=0}^{n} \|m_i - m'_i\| + \omega_t \sum_{i=0}^{n} |t_i - t'_i| \quad (2)$$

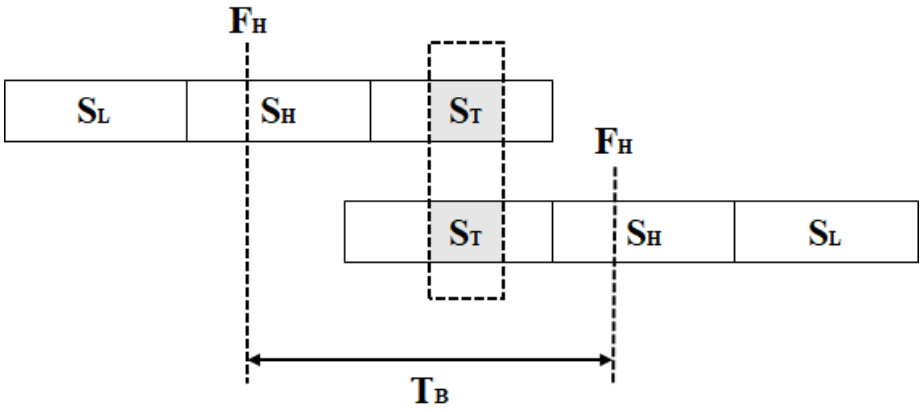


Fig. 5. Connecting motion sequences. The dotted box denotes S_T. T_B represents the ball travel time.

To create a ball trajectory that matches the character's motion, the starting position and the arrival position of the ball are required. **Table 1** shows the algorithm for accomplishing this.

Table 1. Algorithm for Generating Ball Trajectory

1. Select three characters (C_1, C_2, and C_3) to pass the ball in a sequential order.
2. Calculate the movement path of all the balls in the hit ball section (S_{B1}) of the first character (C_1).
3. Select the movement path (T_1) of the ball closest to the position of the second character (C_2).
4. Find the hit position and frame (P_{H1}, F_{H1}) of the first character (C_1) from T_1.

5. Calculate the movement path of all the balls in the hit ball section (S_{B2}) of the second character (C_2).
6. Select the movement path (T_2) of the ball closest to the position of the third character (C_3).
7. Find the hit position and frame (P_{H2}, F_{H2}) of the first character (C_2) from T_2.
8. Calculate the trajectory and the movement time of the ball from (P_{H1}, F_{H1}) and (P_{H2}, F_{H2}).

Once the start position and the arrival position of the ball are determined, the movement trajectory and the movement time of the ball are generated in real time by using the method described in [9]. We can create a moving trajectory of the ball by specifying the physical parameters such as elastic modulus, air resistance, and number of bounces. Once the movement time of the ball is determined, the motion sequence is connected on the basis of the movement time (T_B) of the ball, as shown in **Fig. 5**.

We define the state of the object as $q \equiv (x, v)^T$, where x and v are the position and velocity of the object. In free flight, the state of the object as a function of time can be presented by an ordinary differential equation:

$$\frac{d}{dt}q(t) = F(t, q(t), u) \tag{3}$$

F is derived from Newton's laws and the control vector u contains the parameters for the simulation such as the initial position and velocity. If we integrate this equation, we obtain:

$$q(t) = q_0 + \int_{t_0}^{t} F(t, q(t), u)dt \tag{4}$$

Because this equation is only for free flight, we process the collision of the object as follows:

$$q^+ = q^- + I(q^-, u) \tag{5}$$

Here, q^- and q^+ denote the states before and after the collision, respectively. I denotes the collision impulse. We can abstract the function for the object to include the collision as follows:

$$q(t) = S(t, u) \tag{6}$$

To calculate the control vector u, we linearize this equation partially, by adopting a differential approach:

$$\delta q_i(t) = \frac{\partial S(t_i, u)}{\partial u}\delta u \tag{7}$$

We solve the equation for δu using a conjugate-gradient technique. The differential update is only a small step in the computed direction:

$$u' = u + \epsilon\delta u \tag{8}$$

When the control vector u' is updated, the rigid-body simulator calculates the new movement path for the object. This process is repeated until the movement path converges. **Fig. 6** shows the result of generated ball trajectory. The black curves show the trajectory of all

movements of the ball that can be generated in the hit section. The cyan lines show the speeds of the hit bone in all frames in the hit section. The cyan dots are the hit positions. The white curve shows a selected ball trajectory among them. The red curve is the movement trajectory of the ball, which is exactly adjusted to the hit position of the next character.

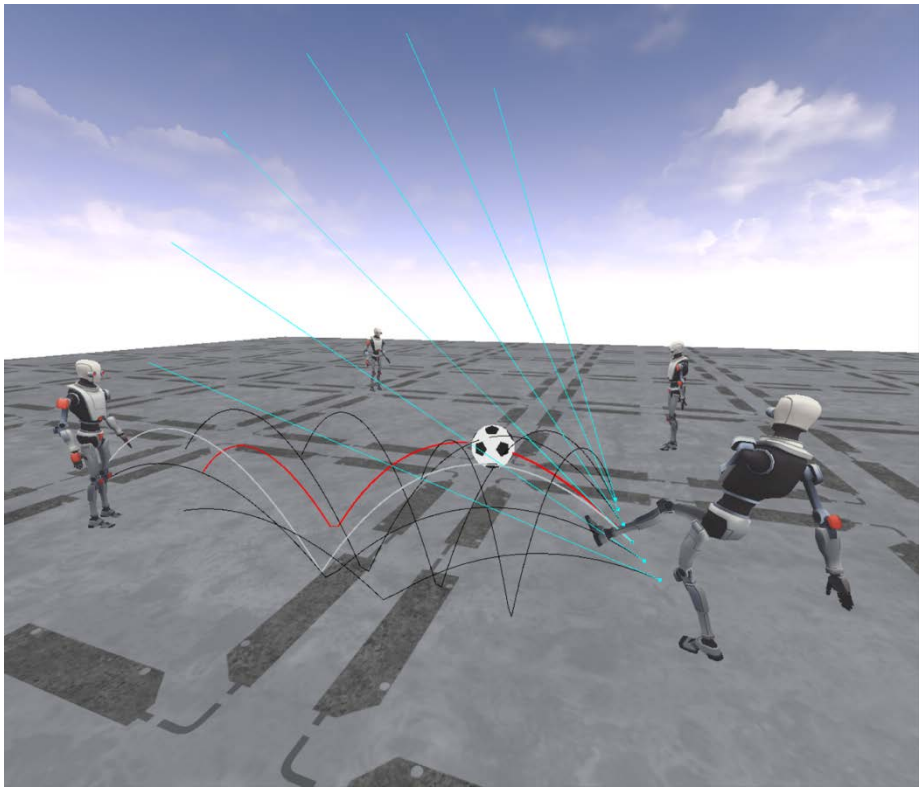


Fig. 6. Ball trajectory generation.

5. Experimental Results

Using our method, it is possible to easily generate collaborative animation of athletes handling a ball. When a user's motion is inputted through the motion sensor such as Kinect, the character motion most similar to the input motion is searched and applied in real time; we can input the motion of the desired character in real time without any additional manual work and create a desired game scene. That way, we were able to quickly and easily create scenes with several characters playing together.

Using the method proposed, we have created virtual players that control virtual balls. We used a computer with a Kinect motion sensor, an Intel Core i7-7700k CPU, 64GB of RAM, and an Nvidia GeForce GTX 1080 Ti GPU to create motions that involve ball-handling with multiple characters effectively. **Fig. 7** shows the result animations generated, and we captured both the left and right motions by subtracting the ball from 0 to 180 degree by 45 degree from the front. We captured the motion in three types of motion: toss the ball for a close distance, pass for a medium distance, and attack the opponent with a strong ball. To generate the result animation, after designating the path sequence of the ball, the number of times of the bounce, the user's motion is input through the motion sensor, and the motion of the character is specified by the user’s motion. The air resistance was 0.2, the coefficient of elasticity was 0.7,

the gravity was 9.8 m/s, and the number of bounces per pass was one. **Fig. 8** is an animation where six characters are places instead of a team of three.

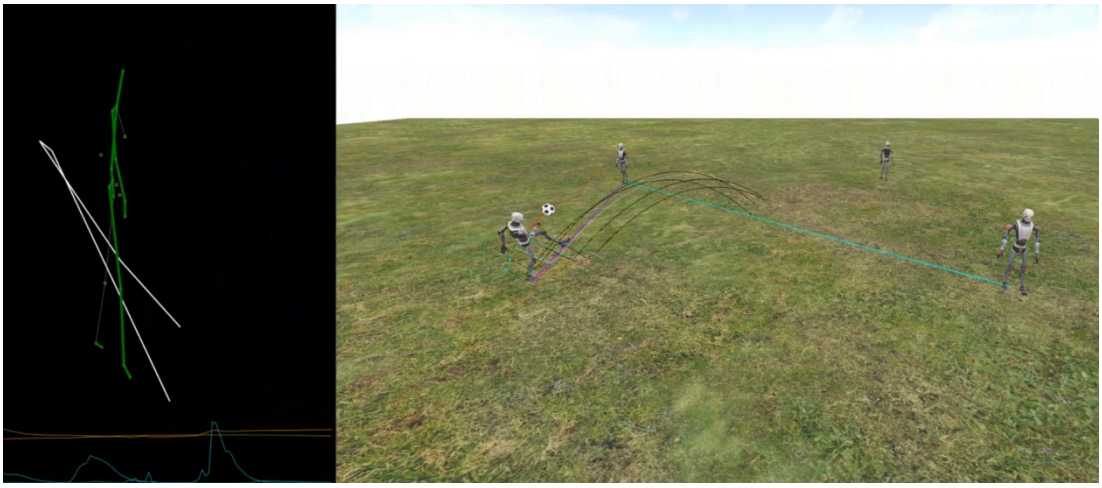


Fig. 7. User's input motion and virtual player's motion

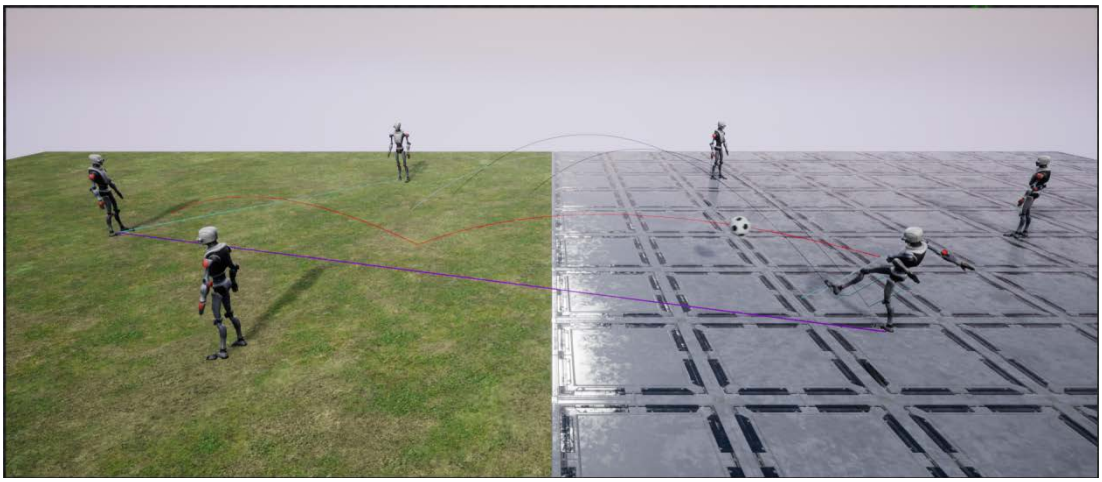


Fig. 8. Ball-controlling animation.

5. Discussion

The method of this paper does not actually control the ball. Instead, it only uses the imitation motion, so the reality may be somewhat reduced. However, with low-cost motion sensors, you can quickly and easily create animations for players who are skilled in the virtual world. Our method assumes that fictitious players make and receive balls without a mistake. Therefore, it is difficult for the player to control the sudden action of losing the ball by striking the opponent. The model does not consider how far one can actually hit the ball. If the distance among the characters is too far or narrow, the motion of the character will appear unnatural. If the character looks at the ball in the idle state or turns toward the side with the ball, the drawing is more realistic. In the motion that we use, the toss and pass have similar motions, which sometimes makes it difficult to distinguish. Thus, we can more precisely distinguish using the maximum speed of the hit bone. We described our method with foot volleyball as an example,

but we can apply our method to various ball games that control balls such as squash, tennis, volleyball, and separktakraw. However, the quality of the motion may deteriorate because it is not the motion of the actual player. In fact, if one uses the motion of the real player to control a ball, one can create a higher-quality animation.

6. Conclusion and Future Work

We introduced a method to generate scenes in which several characters play together using a low-cost motion sensor such as Kinect. The method of this study allows users to easily create scenes where various characters play through motion analysis although the quality of the motion may be somewhat deteriorated. In the future, we will add a system that creates a more natural match scene considering the actual travel time of the ball. It will also be interesting to use virtual reality and motion capture equipment to play with tools such as tennis racquets, or to create scenes where multiple users play together over a network. Now, we are using the motion data to create a virtual player's motion, but in time, we will train the character using deep learning, and then let them play with each other. It will be a difficult but a very meaningful attempt.

References

[1] K. Hyun, K. Lee and J. Lee, "Motion Grammars for Character Animation," *Computer Graphics Forum*, Vol. 35, Issue 2, pp. 103-113, May 2016. Article (CrossRef Link)

[2] K. Lee, S. Lee and J. Lee, "Interactive Character Animation by Learning Multi-Objective Control," *ACM Transactions on Graphics (SIGGRAPH Asia 2018)*, Vol. 37, Issue 6, Dec 2018. Article (CrossRef Link)

[3] L. Liu and J. Hodgins, "Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning," *ACM Trans. Graph.*, vol. 37, Issue 4, Article No. 142, July 2018. Article (CrossRef Link)

[4] S. Jain and C.K. Liu, "Interactive synthesis of human-object interaction," in *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '09*, pp.47-53, 2009. Article (CrossRef Link)

[5] J.I. Choi, S.J. Kang, C.H. Kim and J. Lee," Virtual ball player," *Visual Computer*, Vol. 31, No. 6-8, pp. 905-914, June 2015. Article (CrossRef Link)

[6] J.I. Choi, S.J. Kim, C.H. Kim and J. Lee, "Let's be a virtual juggler," *Computer Animation and Virtual Worlds*, Vol. 27, No. 3-4, pp. 443-450, May 2016. Article (CrossRef Link)

[7] T.H. Kim, S.I. Park and S.Y. Shin, "Rhythmic-motion synthesis based on motion-beat analysis," in *Proc. of ACM SIGGRAPH 2003 Papers, SIGGRAPH '03*, pp.392-401, 2003. Article (CrossRef Link)

[8] T. Shiratori, A. Nakazawa and K. Ikeuchi, "Dancing-to-music character animation," *Computer Graphics Forum*, vol. 25, no. 3, pp. 449-458, 2006. Article (CrossRef Link)

[9] J. Popović, S.M. Seitz, M. Erdmann, Z. Popović and A. Witkin, "Interactive manipulation of rigid body simulations," in *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, pp.209-217, ACM Press/Addison-Wesley Publishing Co.*, pp. 209-217, 2000. Article (CrossRef Link)

[10] J. Popović, S.M. Seitz and M. Erdmann, "Motion sketching for control of rigid-body simulations," *ACM Trans. Graph.*, vol.22, no.4, pp.1034-1054, Oct. 2003. Article (CrossRef Link)

[11] L. Kovar, M. Gleicher and F. Pighin, "Motion graphs," in *Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, New York, NY, USA, ACM*, pp.473-482, 2002. Article (CrossRef Link)

[12] J. Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins and N.S. Pollard, "Interactive control of avatars animated with human motion data," in *Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02, New York, NY, USA*, pp.491-500, 2002. Article (CrossRef Link)
[13] R. Heck and M. Gleicher, "Parametric motion graphs," in *Proc. of the 2007 Symposium on Interactive 3D Graphics and Games, I3D'07, New York, NY, USA*, pp.129-136, 2007. Article (CrossRef Link)
[14] M. Gleicher, "Motion path editing," in *Proc. of the 2001 Symposium on Interactive 3D Graphics, I3D '01, New York, NY, USA*, pp.195-202, 2001. Article (CrossRef Link)
[15] M. Gleicher, H.J. Shin, L. Kovar and A. Jepsen, "Snap-together motion: Assembling run-time animations," in *Proc. of ACM SIGGRAPH 2008 Classes, SIGGRAPH '08, New York, NY, USA*, pp.52:1-52:9, 2008. Article (CrossRef Link)
[16] M. Thorne, D. Burke and M. van de Panne, "Motion doodles: An interface for sketching character motion," in *Proc. of ACM SIGGRAPH 2006 Courses, SIGGRAPH '06, New York, NY, USA, ACM*, 2006. Article (CrossRef Link)
[17] M. Guay, R. Ronfard, M. Gleicher and M.P. Cani, "Space-time sketching of character animation," *ACM Transactions on Graphics*, vol.34, no. 4, Article No. 118, Aug. 2015. Article (CrossRef Link)
[18] J. Lee and K.H. Lee, "Precomputing avatar behavior from human motion data," in *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, Aire-la-Ville, Switzerland, Eurographics Association*, pp.79-87, 2004. Article (CrossRef Link)
[19] S. Levine, J.M. Wang, A. Haraux, Z. Popović and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Transactions on Graphics*, vol.31, no.4, pp.28:1-28:10, July 2012. Article (CrossRef Link)
[20] S. Levine, Y. Lee, V. Koltun and Z. Popović, "Space-time planning with parameterized locomotion controllers," *ACM Trans. Graph.*, vol. 30, no.3, pp. 23:1-23:11, May. 2011. Article (CrossRef Link)
[21] A. Treuille, Y. Lee and Z. Popović, "Near-optimal character animation with continuous control," in *Proc. of ACM SIGGRAPH 2007 Papers, SIGGRAPH '07, New York, NY, USA, ACM*, 2007. Article (CrossRef Link)
[22] Y. Lee, K. Wampler, G. Bernstein, J. Popović and Z. Popović, "Motion fields for interactive character locomotion," *ACM Transactions of Graphics*, vol.29, no.6, pp.138:1-138:8, Dec. 2010. Article (CrossRef Link)
[23] S. Clavet, "Motion matching," in *Proc. of Game Developer Conference 2016*, 2016. Article (CrossRef Link)
[24] D. Holden, T. Komura and J. Saito, "Phase-functioned neural networks for character control," *ACM Trans. Graph*, Vol. 36, Issue 4, Article No. 42, July 2017. Article (CrossRef Link)
[25] X.B. Peng, G. Berseth, K. Yin and M. Van De Panne, "DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph*, Vol. 36, Issue 4, Article 41, July 2017. Article (CrossRef Link)
[26] X.B. Peng, P. Abbeel, S. Levine and M. van de Panne, "DeepMimic: example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, Vol. 37, Issue 4, Article 143, July 2018. Article (CrossRef Link)
[27] W. Yu, G. Turk and C.K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph*, Vol. 37, Issue 4, Article 144, July 2018. Article (CrossRef Link)
[28] H. Zhang, S. Starke, T. Komura and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Trans. Graph*, Vol. 37, Issue 4, Article No. 145, July 2018. Article (CrossRef Link)
[29] L. Liu and J. Hodgins. "Learning to Schedule Control Fragments for Physics-Based Characters Using Deep Q-Learning," *ACM Trans. Graph.*, Vol. 36, Issue 3, Article No. 29, June 2017. Article (CrossRef Link)

[30] J. Chemin and J. Lee, "A physics-based Juggling Simulation using Reinforcement Learning," in *Proc. of ACM Siggraph Conference on Motion, Interaction and Games*, 2018.
Article (CrossRef Link)

Jongin Choi, He received PhD at Korea University in 2016 from the Department of Computer Science from Korea University. After completion, he joined Nexon Korea as a lead client programmer. He has worked at NCSoft Korea as a lead animation programmer in a new AAA online game. Now he is a professor in the major of game contents in Youngsan University.

Sookyun Kim, He received PhD. in Computer Science & Engineering Department from Korea University, Seoul, Korea, in 2006. He joined Telecommunication R&D center at Samsung Electronics Co., Ltd., from 2006 and 2008. He is now a professor at Department of Game Engineering at Paichai University, Korea. He has published many research papers in international journals and conferences. His research interests include multimedia, pattern recognition, image processing, mobile graphics, geometric modeling, interactive computer graphics, and virtual reality..

Sunjeong Kim, She is a professor in the department of convergence software in Hallym University. Her research interests include Geometric Modeling, Scientific Visualization, Virtual Reality and Augmented Reality, GP-GPU Programming.

Shinjin Kang, He received an MS degree at Korea University in 2003. After graduation, he joined Sony Computer Entertainment Korea (SCEK) as a video game programmer. From 2006, he has worked at NCsoft Korea as a lead game designer from 2009. He received a PhD degree in Computer Science and Engineering at Korea University in 2011. And he is now a professor at the school of games in Hongik University.