IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 February 2024, accepted 24 March 2024, date of publication 1 April 2024, date of current version 5 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3383318

RESEARCH ARTICLE

Motion Generation and Analyzing the User's Arm Muscles via Leap Motion and Its Data-Driven Representations

JONG-HYUN KIM[1], JUNG LEE[2], AND YOUNGBIN KIM[3], (Member, IEEE)
[1]Department of Design Technology, College of Software and Convergence, Inha University, Michuhol-gu, Incheon 22212, South Korea
[2]Department of Computer Engineering, Hanbat National University, Yuseong-gu, Daejeon 34158, South Korea
[3]Graduate School of Advanced Imaging Science, Multimedia & Film, Chung-Ang University, Seoul 06974, South Korea

Corresponding author: Youngbin Kim (ybkim85@cau.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Basic Science Research Program through NRF funded by the Ministry of Education (Contribution Rate: 60%) under Grant 2022R1F1A1063180; in part by the Institute for Information & Communications Technology Planning & Evaluation (IITP) through Korean Government (MSIT) (Artificial Intelligence Graduate School Program, Chung-Ang University, Contribution Rate: 10%) under Grant 2021-0-01341; in part by NRF through Korean Government (MSIT) (Contribution Rate: 10%) under Grant NRF2022R1C1C1008534; and in part by the Ministry of Culture, Sports and Tourism, and the Korea Creative Content Agency (Contribution Rate: 20%) under Project R2020040186.

ABSTRACT In this study, we introduce a novel framework for practicing and analyzing arm muscles in motions, such as juggling actions, by estimating the hand motion of the user using a Leap Motion device. The proposed method can map the movement of a ball in a virtual world to the hand motion of the user in real time and visualize the relaxation and contraction of muscles to determine the amount of exercise performed. Our procedure has five sections: 1) The Leap Motion device tracks the hand position of the user. 2) A behavioral pattern in which a user throws a ball is defined as an event. 3) The hand motion is mapped to the ball based on the hand position of the user using the proposed parabola-based particle approach. 4) The quantity of muscle activity is visualized and analyzed in relation to the degree of arm bending. 5) Finally, we propose a method that utilizes the symmetry data-driven approach to extend solvers, enabling the efficient handling of avatar juggling motions in a virtual environment, based on user actions. Moreover, this method enhances the results by allowing diverse control over the virtual ball's trajectory to match the user's pose. Consequently, the proposed system enables real-time juggling in a virtual environment, as well as practice and analysis of the arm muscle activity of the user. The outcomes of the analyses are expected to be applied in various industries, including healthcare. In the solver extensions, we do not utilize all the hand position information of the user. Instead, we base the trajectory of one hand on the data of the other hand, synthesizing it through a data-driven approach. This results in a relatively lightweight algorithm that generates the avatar's juggling motion. Additionally, by leveraging the user's pose and parabolic motion, we can arbitrarily synthesize the trajectory of the virtual ball, facilitating the easy creation of a variety of scenes.

INDEX TERMS Leap motion device, hand motion, arm muscles, healthcare, virtual environments, data-driven, motion generation.

I. INTRODUCTION

The mechanical structure of the musculoskeletal system affects how the human body moves. The human skeleton supports the body, and the muscles contract and relax in response to the skeleton's movement [1], [2], [31]. Physics-based systems relying on the user's muscles have been created in several ways, including directing a person's posture and movement [3], [32], [33], [34]. Methods for reproducing the natural posture and movement of the human body based

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

on the mechanics of the musculoskeletal system have been investigated continually in the field of computer graphics [4], [5], [35]. To adapt these ideas to virtual reality (VR) and augmented reality (AR) environments, various issues must first be addressed: 1) The movements of the user must be captured in real time, and stability of motion capture is just as critical as real-time performance in this process. 2) The ability to control the movement of an object in the virtual environment using the user's hand movement should be possible. If the object's movement in the virtual environment differs from the user's movement, the sensation of immersion is unavoidably diminished. 3) Finally, it must track regions of the body, such as the wrist, that are difficult to track using only hand movement.

Recently, haptic full body suits have been frequently worn in VR to maximize immersion and visual impacts [6], [36]. However, because these devices are primarily employed to acquire high-quality motion data for use in films and are prohibitively expensive for common users, this study involves the use of Leap Motion, a little gadget that precisely tracks hand movements [7], [37]. The three issues described previously must be overcome, as the degree of relaxation and contraction of the arm muscles can only be assessed through hand movements.

Choi et al. developed an interaction rule based on recurrent behavior patterns in arm movements, such as juggling, and offered a technique for synthesizing juggling movements from data [8] (see Figure 1). As with our work, this technique simulates juggling motions by studying the user's movements rather than using real objects. However, because only the rotation of the arm in a fixed position is considered, the motion is rather limited, and because it is based on Kinect, the entire skeleton must be precisely tracked to calculate wrist or elbow movement, creating a stability issue.

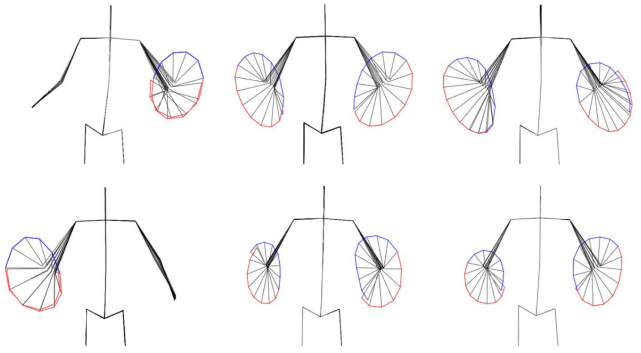

FIGURE 1. Juggling motion in previous studies [8]. The skeleton displays motion clips that were generated using motion capture data. The attach and detach parts are denoted by red and blue lines, respectively.

The Hill-type muscle model, which is based on the physics of human muscles, is utilized in a variety of fields, including computer graphics, ergonomics, and robotics, and it is based on non-linear contractile mechanics of muscles [9], [10]. However, owing to the huge amount of calculations necessary, real-time processing is difficult, making it incompatible with interactive systems, such as mixed reality. Because numerous physical parameters of the Hill-type muscle model, such as pennation angle, maximum force, and way-point, must be set individually for each muscle movement, a sophisticated parameter tuning process is required to get the desired outcome.

While several simulation and control approaches [11], [35] have been developed by merging skeletal and muscular models, it is challenging to extend them to an interactive system that controls virtual objects in response to the user's movements. Lee et al. [11] suggested a musculoskeletal simulation technique that utilizes a polyhedral-based volumetric structure and a line-segment controller to express muscle action via the skeleton. Si et al. [12] offer a volumetric simulation for visualization purposes, and similar to the finite element method (FEM), the skeleton and muscles are expressed using a line-segment controller. Fan et al. [13] stably expressed movement of the skeleton using the movement of volumetric primitives. Nonetheless, control system difficulties comparable to those reported by Si et al. [12] were discovered. Although volumetric muscles [38] are extensively utilized to control face animation, lack of joint structure in skeletal elements such as the mandible creates issues with the quality and stability of the output.

In this study, we use real-time analysis of hand movements to generate juggling motions based on physics rather than rule-based synthesis. We present a framework to visualize and assess muscular relaxation and contraction using approximated wrist movements that are easily identifiable by users.

II. RELATED WORKS

In this section, the kinematic motion synthesis techniques and the physically-based motion control methods related to data-based motion prediction and physically-based motion synthesis are reviewed. The user's motion synthesis techniques used in robotics or character animation are also discussed.

A. KINEMATIC LOCOMOTION CONTROL

A graph-based method is an approach commonly used in the early stages of data-driven motion synthesis [14], [39], [40]. In this method, transition motion between example motion clips is allowed only at preselected transition points according to the motion specifications of the user. This method is easily used in a static environment. However, it is inappropriate in the synthesis of highly responsive online motions such as those in a constantly changing environment because of the characteristics of the algorithm. Although a learning-based approach for real-time motion synthesis has been proposed recently [15], [16], [17], [18], [41], the extension of the algorithm to process external objects, such

as moving a ball in an online manner, has not been clearly described, and the algorithm extension seemed complicated. In this paper, a physically-based method, using a technique to measure the user's hand motion [42] and physics to match the ball's motion, is proposed. The proposed method, unlike the learning-based approach, allows us to synthesize smooth motions, such as juggling without heavy preprocessing, and realize motion transitions, such as catching the ball or throwing the ball harder.

B. PHYSICS-BASED MOTION CONTROL

The physics-based motion control has been intensively studied for several years in the computer graphics field; in addition to waking motion [19], [20], [21], [22], [43], [44], [45], [46], non-walking motions, such as martial arts kicks, crawling, jumping, and dancing, were also addressed [23], [24], [25], [26], [27], [30]. The dynamic equations for whole-body motion control have problems of discontinuities due to changes in contact points with the environment. Liu et al. solved this problem using a control graph representing rapid changes of contact in an example motion clip [25]. Han et al., using the soft contact method, solved the problem due to the discontinuous motion problem [24]. They focused on reference motion tracking and did not take the interaction between characters and moving external objects into consideration.

C. USER'S MOTION-BASED INTERACTION CONTROL

In robotics, some authors attempted to have a humanoid robot play with a ball [28], [47], [48], [49], [50]. The football motions were limited to kicks while standing or walking owing to their complex physical mechanisms.

Various techniques for synthesizing soccer motion have also been proposed in the field of character animation. Jain and Liu proposed a method of editing interactive motion, the interaction between human and football ball [51]. This method automatically generates the trajectory of the ball based on simple rigid body mechanics by considering the contact with body parts. The timing and site of contact between the ball and body parts were manually determined by the user. This method focused on the human motion and ball trajectory, did not consider whole-body dynamics, and expressed only the motion (e.g., one with strong stretching) observed in a fast-moving ball. Choi et al. proposed a data-driven method to generate 2D juggling motions, such as hand juggling, soccer juggling, and in-place basketball dribbling after analyzing kinematic information such as end-effector velocity [8], [29]. Ding et al. proposed a low-dimensional linear feedback strategy that effectively retrieves the control policy to generate a simple motion of kicking a soccer ball while fixing the hip [52]. Peng et al. proposed a framework that learns a control policy for walking motion by providing a soccer-specific reward function to induce a character to dribble the ball to the desired destination [21].

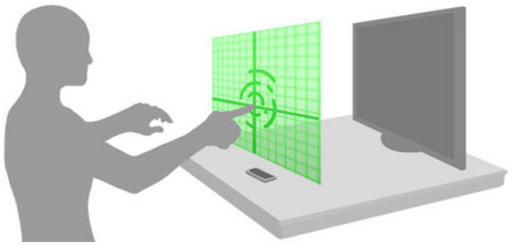

FIGURE 2. Leap Motion controller can be used to create a virtual *touch* surface in the air.

III. PROPOSED FRAMEWORK

A. GETTING HAND POSITION FROM LEAP MOTION DEVICE

In this study, the position of the hand is tracked using a Leap Motion device. We used a depth camera based on the working principle similar to that of the Kinect, produced by Microsoft, but has a sensitivity 200 times higher; thus it can detect movements of up to $\frac{1}{100}$ millimeters. Leap Motion has been used in various fields owing to its intuitive interface in that it recognizes motion in the area close to the monitor, while Kinect recognizes motion from a position a little farther from the front for a full-body scan of the monitor.

In this study, the hand position of the user is determined using the Leap Motion (see Figure 2) mentioned above. To calculate the juggling motion from this information, the ball is moved in the direction of the vector in which the hand moves when the magnitude of the change in hand motion is greater than the predetermined level (see Listing 1).

```
List<Finger> fingers = hand.Fingers;

Frame frame = controller.Frame();
for (int i = 0; i < frame.Hands.Count; i++) {
        Hand hand = frame.Hands[i];
        Vector handXBasis = hand.PalmNormal.Cross(
                hand.Direction).Normalized;
        Vector handYBasis = -hand.PalmNormal;
        Vector handZBasis = -hand.Direction;
        Vector handOrigin = hand.PalmPosition;
}
```

LISTING 1. Getting hand coordinates in Leap Motion device.

For the codes above, handXBasis, handYBasis, and handZBasis represent the coordinates when moving along the X, Y, Z axes, respectively, and handOrigin represents the origin position of the hand. The X-axis indicates a positive number when moving to the right and a negative number moving to the left, and this is true in other axes.

Figure 3 shows the ball movement resulting from an external force, in this case, both hands. By default, collision processing was not applied between the ball and the hand, and it was designed for the ball to be attached to the hand when near it. As shown in Figure 3, on the application of an upward force as if bouncing hands, the ball moves upward accordingly. In this process, the faster movement of the hand caused stronger force, leading to faster movement of the ball

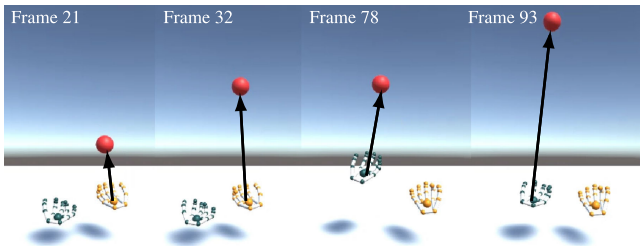

FIGURE 3. Ball movement in response to hand movement (black arrow: ball trajectories by external force).

(see Equation 1).

$$\mathbf{F} = \left(m\frac{d\mathbf{v}}{dt}\right)\mathbf{F}^h = (m\mathbf{a})\,\mathbf{F}^h \tag{1}$$

where, m is weight, $\mathbf{v}$ is velocity, $\mathbf{a}$ is acceleration.

B. PARABOLIC BALL MOVEMENT USING INVERSE DYNAMIC APPROACH

The forward dynamics approach described in the previous section is a method of calculating the movement of the ball using an external force (see Figure 4). The application of this method to juggling motion needs information on the exact magnitude of external force required to move the ball from one hand to the other. Even a slight difference between them can cause the ball to fall to a location other than the hand, leading to failure in generating juggling motion.

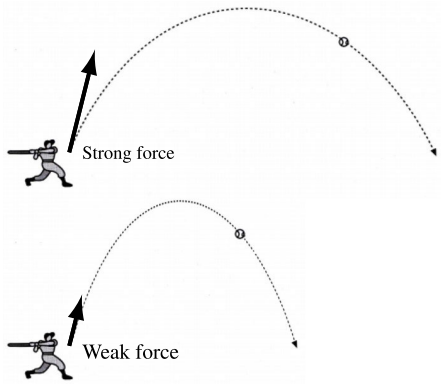

FIGURE 4. Calculated motion of ball with forward dynamics by external force.

Since the juggling motion follows a parabolic trajectory, we propose a mapping function that maps the movement of the ball inversely based on the hand position. To formulate this mathematically, we assume that the parabolic motion is performed on a 2D plane. Assuming that the ball moves on a plane and that gravity, g, is applied in the direction perpendicular to the X axis, the motion equation of motion in the form of Euler integral is as follows (see Equation 2).

$$v_z = v_{z0} \tag{2a}$$
$$v_y = v_{y0} - g\Delta t \tag{2b}$$
$$x = x_0 + v_{z0}\Delta t \tag{2c}$$

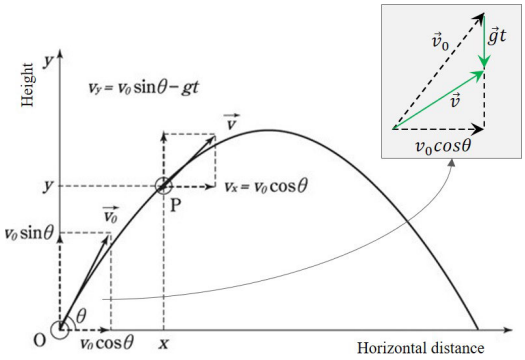

FIGURE 5. Parabolic motion graph.

$$y = y_0 + v_{y0}\Delta t - \frac{1}{2}g\Delta t^2 \tag{2d}$$

where, $\{x, y\}$ is location, $\{v_x, v_y\}$ is velocity, Δt is time-step, and $\{v_{y0}, v_{z0}\}$ is initial velocity.

Since parabolic motion shows the same trajectory as a projectile and is affected by speed and gravity at the initial stage, the above equation could be converted into simple harmonic motion, and the converted equation is as follows (see Equation 3).

$$v_{z0} = v_0 cos\theta_0 \Delta t \tag{3a}$$
$$v_{y0} = v_0 sin\theta_0 \Delta t - \frac{1}{2}g\Delta t^2 \tag{3b}$$
$$\Delta t = \frac{2v_0 sin\theta}{g} \tag{3c}$$

where, Δt is the time the ball stays in the air. To obtain the maximum horizontal movement distance, Δt was substituted into Equation 3a, and the velocity was obtained using d as shown below (see Equation 4).

$$d = \frac{v_0^2 sin2\theta}{g} \tag{4a}$$
$$d \cdot g = v_0^2 sin2\theta \tag{4b}$$
$$v_0^2 = \frac{d \cdot g}{sin2\theta} \tag{4c}$$

The distance traveled per unit time was obtained by assuming that the distance from the position of the hand is fixed, establishing the parabolic equation and calculating inversely the velocity. This was used in calculating the distance traveled per unit time along the trajectory. Equation 5a and Equation 5b show the velocity of the ball calculated using the distance, Equation 5c and Equation 5d are the positions calculated by integrating the velocities of the horizontal and vertical components over Δt. Figure 5 shows a chart expressing this movement as a chart.

$$v_x = v_0 cos\theta \tag{5a}$$
$$v_y = v_0 sin\theta - g\Delta t \tag{5b}$$
$$x = v_0 \Delta cos\theta \tag{5c}$$
$$y = v_0 \Delta sin\theta - \frac{1}{2}g\Delta t^2 \tag{5d}$$

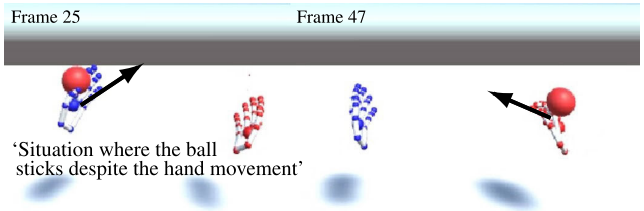

FIGURE 6. The problem of the ball stuck in hand (black arrow : direction of external force).

Though the trajectory of the ball calculated using the equations mentioned above is a parabolic one, the shape is unnatural since no acceleration was applied. The acceleration calculated based on the change in hand position was added to the parabolic-based motion as an external force (see Equation 6).

$$\mathbf{F}^a = m\mathbf{a} \tag{6a}$$

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{d}{dt}\frac{d\mathbf{p}}{dt} = \frac{d^2\mathbf{p}}{dt^2} \tag{6b}$$

The ball was set to be projected in the direction calculated using the orientation of the hand. In this case, there may be a problematic situation where the position of the flying ball is in contact with the hand. The ball is stuck to the hand; thus, flying is interrupted (see Figure 6). To address this issue, when throwing the ball, the event where the ball sticks to the hand for an extended time is disabled. The delay time is set to $3\Delta t$ in this case.

Since both hands and two balls are used in juggling, the distances between the hands and the balls were calculated so that a hand catches the ball whose distance is shorter than the predetermined one. In addition, it was set to follow the parabolic trajectory mentioned above in juggling motion. While calculating the direction and magnitude of the force of the thrown ball, the distance between the two hands, acceleration of the hand, and slope were applied to make the ball fall accurately onto the position of the opposite hand (see Figure 7).

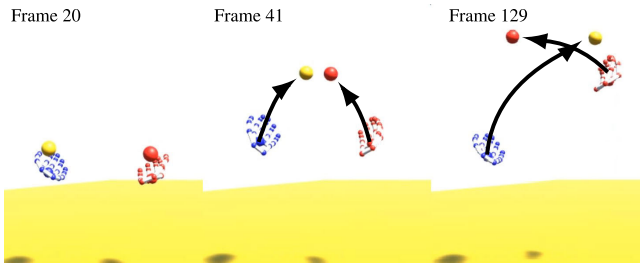

FIGURE 7. Two-handed juggling motion (black arrow: ball trajectories by external force).

If the right hand suddenly moves after the ball, thrown by the left hand, moves in a parabolic trajectory, the ball may not reach the right hand. If the right hand moves toward the ball, the ball is stuck to the right hand, allowing the user to interact continuously.

C. RELAXATION AND CONTRACTION OF MUSCLES USING ARM BENDING

This section discusses a method to model a virtual wrist shape to calculate the arm momentum using user hand motion. Since the Leap Motion device provides only information on the position of the hand, it has limitations in analyzing arm movement or muscle mass. To overcome this problem, we modeled an imaginary wrist. Additionally, we approximated the momentum of the muscle using the bending between the wrist joints during juggling.

We modeled the arm using the movement of the hand by dividing it into two sub-arm structures, the upper arm (Brachium) and the forearm (Antebrachium) (see Figure 8). After obtaining a vector from the position of the third finger to the position of palm (see black arrow in Figure 8), the vector was scaled to calculate s_2. s_1 was calculated by rotating the head of s_2 by a user-specified angle. These procedures were performed for both hands to model sub-arm structures. The final main body was made into a hexahedron shape, and the virtually modeled arms were placed at the center of the main body. The method described above models the skeleton using the positions captured by Leap Motion. The site connected to the main body (see the blue sphere in Figure 8) was fixed, and the movement of s_1 and s_2 varied depending on the position of the hand. The rotation angle in the process of changing from s_2 to s_1 was set to 120°. This angle was allowed to be adjusted by the user, and 120° is a value obtained from normal juggling motion.

Figure 8 shows a posture, which was set as the default posture in this study since it is an initial and ready posture in actual juggling.

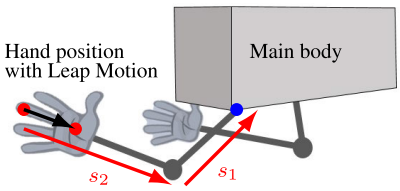

FIGURE 8. Main body and arm skeleton (blue sphere: fixed position).

This study considered the fact that the actual relaxation and contraction of the muscle is affected by bending (see Figure 9), and the bending information is calculated as follows (see Equation 7). In this study, this information is used after being normalized as 0∼1.

$$s^b = cos^{-1}\left(\frac{s_1 \cdot -s_2}{\|s_1\| \, \|-s_2\|}\right) \tag{7}$$

To calculate the muscle momentum using the wrist joint, the muscles frequently used due to bending of the wrist, not the entire muscle of the arm were segmented into patches before analysis (see Figure 10).

Muscle contraction and relaxation during juggling exercise are largely divided into two movements. Since the juggling motion of throwing the ball has only two situations, bending

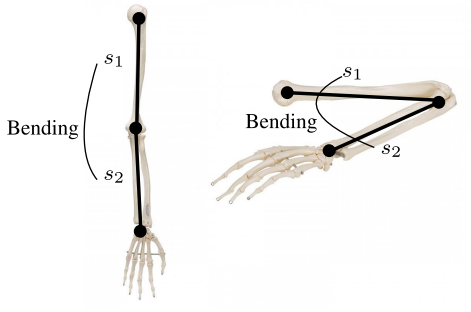

FIGURE 9. Bending force caused by folding of skeleton.

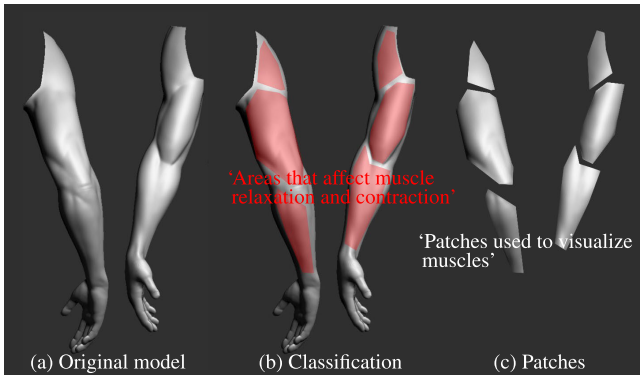

FIGURE 10. Segmentation of the patch from the arm model.

and stretching, this study considers only these two situations. The situation for calculating the bend is as follows:

1) arm bending: when s^b is 0∼0.5 (see Figure 11a)
   - contracting muscle: a color close to blue
   - relaxing muscle: a color close to red
2) arm stretching: when s^b is 0.5∼0.9 (see Figure 11b)
   - contracting muscle: a color close to blue
   - relaxing muscle: a color close to red

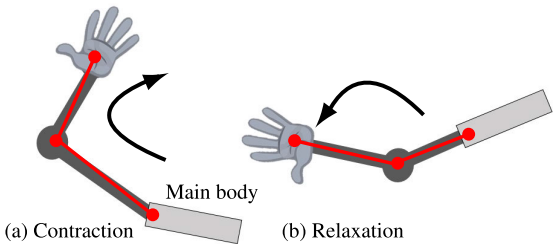

FIGURE 11. Muscle contraction and relaxation calculated by bending of the joint.

Considering that, in juggling movement, the muscle contracts when the arm is raised, and it relaxes when the arm is lowered, we can observe the muscle contraction and relaxation during the juggling movement in real-time. Muscle contraction occurs when the arm is stretched to throw the ball upward in the juggling movement, and in this process, the value of s^b is 0∼0.5 (see Figure 11a). Muscle relaxation occurs when the arm is lowered while turning it after throwing the ball in the juggling movement (see

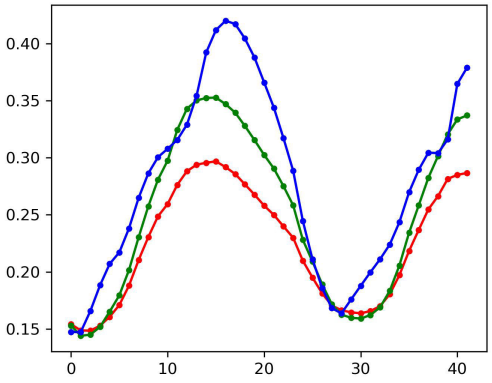

FIGURE 12. Chart depicting changes in vertical velocity, with red, green, and blue representing the Y-axis variations for the wrist, palm, and middle finger, respectively.).

Figure 11b). We visualized the determined contraction and relaxation of muscles by converting them into color.

IV. SOLVER EXTENSIONS

In this section, we discuss extending the algorithm to not only generate avatar motions but also various forms of ball trajectories in a data-driven manner, leveraging the Leap Motion device and user poses. We utilize the previously mentioned Leap Motion device to capture the user's motion data and acquire juggling actions. During this process, the movements of the user's right hand are captured and saved in a CSV file. The similarity between the stored data and the input data is computed to select the most similar motion clip. Since only the data of the right hand is used, the data-driven method is employed to virtually generate symmetrical data for the left hand. Subsequently, based on the data of both the right and left hands, a ball moving in the virtual environment is generated, and a juggling animation synchronized with the avatar's movement is synthesized.

A. MOTION DATA GENERATION

This section elaborates on the methodology for capturing the user's hand motion and generating motion data. Users interact with the Leap Motion device to derive data by moving their right hand. During this interaction, the positional coordinates of the user's right wrist, palm, and middle finger are recorded in a CSV file. Subsequently, the stored positional data is utilized to calculate the vertical velocity, serving as a benchmark for when the avatar throws or catches the ball. The change in vertical velocity is graphically represented, with red indicating the wrist, green the palm, and blue the middle finger (see Figure 12). Based on the vertical velocity, frames where the velocity is negative are identified as the catching frames of the ball, while those with positive velocity are recognized as the throwing frames.

B. CALCULATING SIMILARITY

Before calculating the similarity between the input motion and the pre-stored motion data, the frames where the maximum speed occurs during catching and throwing are

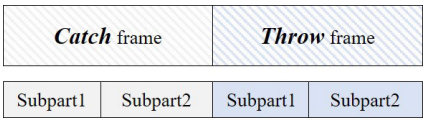

FIGURE 13. Diagram illustrating the structure of the frame at maximum speed.

determined. Subsequently, two sub-intervals are classified: 1) from the first frame to the maximum speed frame, and 2) from the maximum speed frame to the last frame (see Figure 13). This division results in a total of four sections. Within these sections, similarity is computed utilizing the motion vector and the travel time.

The calculation of similarity is based on the previous methodology [8]. The first term computes the similarity concerning the motion and travel time, while the second term indicates the similarity related to rotation (see Equation 8).

$$D = w_1 D_M\left(M_i, M_j\right) + w_2 D_T\left(M_i, M_j\right) \tag{8}$$

The notations used in the above equation are detailed in the following section, categorized by the type of similarity.

1) MOTION VECTOR AND TRAVEL TIME SIMILARITY

In this section, we describe the methodology for calculating the similarity between the vectors and travel times of the input motion data. The similarity is determined by utilizing the differences between the motion data inputs across each interval. Subsequently, a weighted product is computed across the four intervals to calculate the first term, D_M (refer to Equation 9). This process is repeated for each of the right hand's wrist, palm, and middle finger data.

$$D_M\left(M_i, M_j\right) = w_v \sum_{i=1}^{4} \left\| v_i - v_i' \right\| + w_d \sum_{i=1}^{4} \left| d_i - d_i' \right| \tag{9}$$

Here, w_v represents the weight for the motion vector, and w_d represents the weight for the travel time. Additionally, v_i and v_i' respectively denote the i-th motion vector of the input motion and the motion clip. Similarly, d_i and d_i' represent the i-th travel time of the input motion and the motion clip, respectively. The values for v_i and d_i are calculated as follows: $v_i \leftarrow p_{i+1} - p_i$, $d_i \leftarrow d_{i+1} - d_i$.

2) ROTATION SIMILARITY

In this section, we also delve into the methodology for calculating the rotational similarity of poses, irrespective of the position and direction of the input motion data. This involves comparing the rotational similarity between the input motion data to identify the smallest value (see Equation 10).

$$D_T\left(M_i, M_j\right) = min_{\theta, x_0, z_0} \sum_k w_k \left\| p_{i,k} - T_{\theta, x_0, z_0} p_{j,k} \right\| \tag{10}$$

$$\theta = arctan \frac{\sum_i w_i \left(x_i z' - x' z_i\right) - \left(\bar{x}_i \bar{z}' - \bar{x}' \bar{z}_i\right)}{\sum_i w_i \left(x_i x' - z' z_i\right) - \left(\bar{x}_i \bar{x}' - \bar{z}' \bar{z}_i\right)} \tag{11}$$

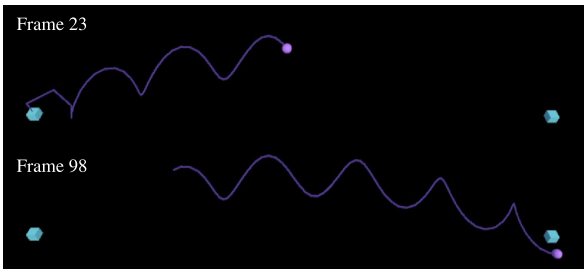

FIGURE 14. Visualization showcasing the integration of rotational motion into the ball's parabolic trajectory.

Here, θ denotes the angular equation, and x_0 and z_0 are calculated as follows: $x_0 \leftarrow \left(\bar{x} - \bar{x}' cos\theta - \bar{z}' sin\theta\right)$, $z_0 \leftarrow \left(\bar{z} - \bar{x}' cos\theta - \bar{z}' sin\theta\right)$. $T_{\theta, x_0, z_0} p_{j,k}$ represents a linear transformation that rotates by θ around the Y-axis and then translates by (x_0, z_0). Furthermore, $p_{i,k}$ and $p_{j,k}$ correspond to the points generated from the i-th frame of the input motion data and the j-th frame, respectively. The weight w_k totals to 1, having its largest value in M_i and M_j, and diminishing as it moves away from neighboring frames. Here, $\bar{x} = \sum_i w_i x_k$, and other terms with bars follow a similar notation. Our approach aims to identify the smallest rotation difference calculated by the above equation. This process is reiterated for each of the wrist, palm, and middle finger data.

To generate the motion of a player handling virtual objects, the selected motion clips are concatenated based on the method described previously. The connection of motion clips involves blending the second combining section of the first motion clip with the first combining section of the second motion clip (see Equation 12).

$$\alpha(t) = 2t^3 - 3t^2 + 1 \quad (0 < t < 1) \tag{12}$$

Here, t represents the normalized time in the blending section.

C. SYNTHESIZING VIRTUAL OBJECTS WITH DATA-DRIVEN

Since we only stores the data of the right hand, real-time synthesis of the left hand's movement is necessary to apply a two-handed juggling animation. As the left hand rotates opposite to the right hand during juggling actions, the motion data is read in reverse using Unity3D's StreamReader and synthesized with a time difference to render a natural juggling motion. Moreover, as the left hand represents synthesized data, its speed is dependent on the speed of the right hand's data. Before and after the user throws the ball, the avatar holds the ball, making the ball move along with the avatar. This process easily determines which hand the ball is in through state variables.

In addition to the avatar, the trajectory of the virtual ball is implemented using the method described in Section III. Furthermore, this process allows mapping not just simple juggling trajectories but also various motions of the virtual ball. We have efficiently controlled the monotonous trajectory of the ball by adding rotational motion to the trajectory of the ball as proposed earlier (see Figure 14). The rotational motion

demonstrated in the results is calculated using the following equation (see Equation 13).

$$pos.y = v.y + cos(\theta)r \tag{13a}$$
$$pos.z = v.z + sin(\theta)r \tag{13b}$$
$$\theta += \chi \Delta t \tag{13c}$$

Here, θ represents the rotation angle of the ball, v denotes the position of the ball calculated through parabolic motion, and χ is the rotation speed. This approach is applied when the ball is not positioned at the hand position, thereby preventing unstable simulation states during the process of the user catching and throwing the ball.

V. IMPLEMENTATION

This study was implemented in the following environment: Intel i7-7700k 4.20 GHz CPU 32 GB RAM, NVIDIA GeForce GTX 1080 Ti graphics card. Unity3D and Leap Motion device were used for IDE and hand tracking device, respectively. Although the performance of the proposed method is sufficient to be utilized as a real-time application, if the resolution required for the 3D arm model to visualize muscle relaxation and contraction is too high, the overall performance may deteriorate. We applied mesh simplification; this process is not related to the algorithm optimization proposed herein; therefore, the selection of 3D model for muscle relaxation and contraction visualization is entirely dependent on the user. The only human participants in this study were the authors, and the participants gave informed verbal consent for their participation.

VI. RESULTS

We analyzed the arm movement of users obtained from the position of the hand using Leap Motion. The simulation of virtual ball juggling was based on arm movement. Some movements were tested to verify the validity of the proposed method.

A. SIMULATION RESULTS

Figure 15 shows juggling motion resulting from the proposed method, indicating that the ball is moving stably in a parabolic trajectory according to the user's wrist movement. The shape of the virtual wrist was expressed excellently similar to that in the real footage (see inset image in Figure15). Since the Leap Motion device provides only the position of the hand, it is not possible to clearly determine the bending or movement of the wrist. By contrast, the proposed method calculated the degree of bending with high accuracy; based on it, the relaxation and contraction of the muscle were determined.

Figure 16 presents a visualization of muscle changes according to various postures of the left hand. Frame 47 shows the moment when the arm muscles are bent relatively inward by putting the hand close to the chest, expressing the movement of the arm similar to that in the real world. (A) shows the triceps muscle, an outer muscle, and expresses excellently the typical relaxation form in the

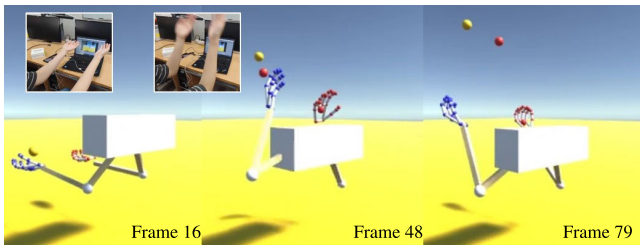

FIGURE 15. Two-handed juggling in a virtual environment with our method (inset image : user motion).

bent arm, and (B), representing the biceps, shows a relatively strong contraction. This result is the same as the change in muscle occurring in arm bending. In Frame 89, arm bending is weaker than in Frame 47 by moving the hand forward. Though (A) and (B) expressed muscle contraction and relaxation as in previous Frames, respectively, the degree of bending was weaker as shown clearly by the change in color. Frame 121 shows a relaxed muscle resulting from stretching the arm upward of the left hand, and Frame 47 and 89 show the opposite situation: (A) shows a change in the contracted muscle, and (B) shows a relatively relaxed muscle. These results are expressed in the same way as when bending or stretching an arm, and it is a result showing that the proposed method captures muscle changes using only the position of the hand.

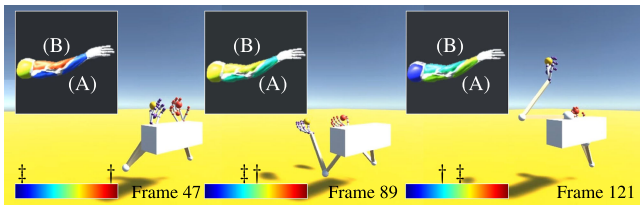

FIGURE 16. Visualization of muscle contraction and relaxation of left arm with our method (red : contraction, blue : relaxation, ‡ : color of (A), † : color of (B)).

Figure 17 shows the change of biceps, an inner muscle of the upper arm, with time. The contraction and relaxation of muscles according to bending are expressed as a chart where the values increased and decreased for relaxation and contraction, respectively. In Frame 31, the contraction of the biceps due to inward bending of the arm is clearly shown in the chart (see (A) in Figure 17). Frame 56 shows the stretching motion of arm, a relaxed muscle compared to (A) (see (B) in Figure 17). The gradual change of Frame from (A) to (B) represents a shift from contraction to relaxation, as clearly shown in the chart (see (c) in Figure 17).

Frame 93 shows an arm stretching forward where the muscle is more relaxed compared to (C) because the arm is suddenly stretched from the bent state (see (D) in Figure 17). Frame 121 shows the contraction resulting from a bent arm again, and Frame 156 shows further contraction, which was clearly shown in the chart (see (F) in Figure 17).

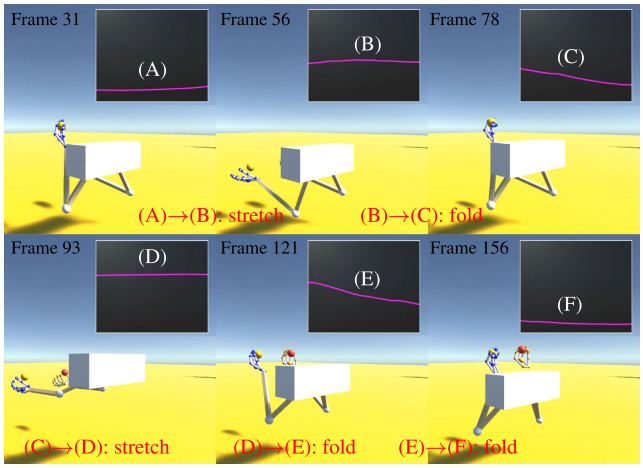

FIGURE 17. Graph showing muscle changes of left arm over time.

FIGURE 18. Avatar motion and ball trajectory precisely controlled via the data-driven method.

FIGURE 19. Experiment scene with an increased arm spacing, maintaining the configuration as depicted in Figure 18.

B. DATA-DRIVEN RESULTS

In this section, we discuss the results generated through solver extensions. For representing natural joint movements of the avatar, Inverse Kinematics (IK) was applied to the avatar within this study.

Figure 18 showcases the results of the avatar's motion and the virtual ball's trajectory controlled via the data-driven method as part of solver extensions. The method we propose not only generates real-time parabolic motion of the ball but also ensures synchronization with the avatar's motion. Notably, by utilizing the data from just one hand, it enables synchronization of movements for both hands and the virtual ball, proving beneficial for real-time content applications.

Figure 19 presents the results from a scene with a wider arm spacing. Our method stably synthesizes avatar motion even in scenes with wide spacing, ensuring consistent control over the trajectory of the virtual ball. Conversely, Figure 20 depicts the outcomes from a scene with narrower arm spacing, demonstrating stable results in real-time simulation as well.

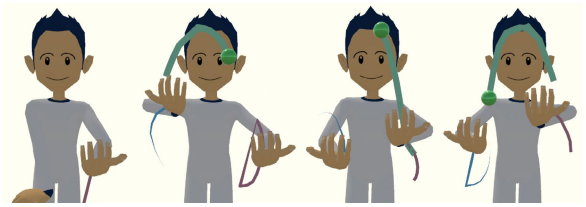

FIGURE 20. Experiment scene featuring a reduced arm spacing, consistent with the configuration presented in Figure 18.

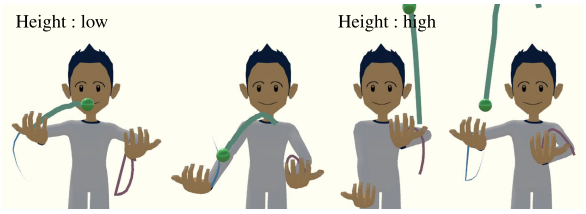

FIGURE 21. Experiment scene demonstrating varying heights of the ball, retaining the configuration as exhibited in Figure 21.

Figure 21 illustrates the results from varying the height of the virtual ball. The proposed method delivers natural, real-time animation results by synchronizing the synthesized hand position above the ball, irrespective of the ball's height in the scene, using the data-driven method. During this process, 200 pieces of data were utilized for the data-driven method.

VII. CONCLUSION AND FUTURE WORK

We modeled a virtual ball moving in a juggling trajectory in real time using the position information of the hand obtained through the Leap Motion device. We added virtual arms and joints based on the rotation vector and position of the hand and showed which muscle the user was using at given instants by visualizing the relaxation and contraction of the arm muscles through joint bending. Moreover, our method extends solvers through the symmetry data-driven approach, enabling efficient processing of the avatar's juggling motion in a virtual environment in response to user actions. Additionally, it enhances results by allowing diverse control over the virtual ball's trajectory to align with the user's pose.

Our algorithm is designed for individualized juggling motion and does not account for multi-players. Future plans include extending this algorithm to control multiple paths by multiple users and facilitating interaction between players. Additionally, since the data-driven method relies on data stored in the database, synthesizing data from a side camera rather than the front may lead to inaccuracies. To mitigate this issue, we plan to harness neural networks in the future.

REFERENCES

[1] S. Park and J. K. Hodgins, "Data-driven modeling of skin and muscle deformation," in *Proc. ACM SIGGRAPH*, 2008, pp. 1–6.

[2] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–13, Aug. 2019.
[3] L. Kumarapu and P. Mukherjee, "AnimePose: Multi-person 3D pose estimation and animation," 2020, *arXiv:2002.02792*.
[4] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–12, Dec. 2019.
[5] H. Park, R. Yu, Y. Lee, K. Lee, and J. Lee, "Understanding the stability of deep control policies for biped locomotion," 2020, *arXiv:2007.15242*.
[6] M. Sra and C. Schmandt, "MetaSpace II: Object and full-body tracking for interaction and navigation in social VR," 2015, *arXiv:1512.02922*.
[7] W. Lu, Z. Tong, and J. Chu, "Dynamic hand gesture recognition with leap motion controller," *IEEE Signal Process. Lett.*, vol. 23, no. 9, pp. 1188–1192, Sep. 2016.
[8] J. Choi, S. Kim, C. Kim, and J. Lee, "Let's be a virtual juggler," *Comput. Animation Virtual Worlds*, vol. 27, nos. 3–4, pp. 443–450, May 2016.
[9] D. G. Thelen, "Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults," *J. Biomechanical Eng.*, vol. 125, no. 1, pp. 70–77, Feb. 2003.
[10] F. E. Zajac, "Muscle and tendon: Properties, models, scaling, and application to biomechanics and motor control," *Crit. Rev. Biomed. Eng.*, vol. 17, no. 4, pp. 359–411, 1989.
[11] S.-H. Lee, E. Sifakis, and D. Terzopoulos, "Comprehensive biomechanical modeling and simulation of the upper body," *ACM Trans. Graph.*, vol. 28, no. 4, pp. 1–17, Aug. 2009.
[12] W. Si, S.-H. Lee, E. Sifakis, and D. Terzopoulos, "Realistic biomechanical simulation and control of human swimming," *ACM Trans. Graph.*, vol. 34, no. 1, pp. 1–15, Dec. 2014.
[13] Y. Fan, J. Litven, and D. K. Pai, "Active volumetric musculoskeletal systems," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–9, Jul. 2014.
[14] O. Arikan and D. A. Forsyth, "Interactive motion generation from examples," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 483–490, Jul. 2002.
[15] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–11, Jul. 2016.
[16] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Aug. 2017.
[17] Y. Lee, K. Wampler, G. Bernstein, J. Popović, and Z. Popović, "Motion fields for interactive character locomotion," *Commun. ACM*, vol. 57, no. 6, pp. 101–108, Jun. 2014.
[18] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, Jul. 2012.
[19] M. de Lasa, I. Mordatch, and A. Hertzmann, "Feature-based locomotion controllers," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 1–10, Jul. 2010.
[20] U. Muico, J. Popović, and Z. Popović, "Composite control of physically simulated characters," *ACM Trans. Graph.*, vol. 30, no. 3, pp. 1–11, May 2011.
[21] 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, no. 4, pp. 1–13, Aug. 2017.
[22] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, Aug. 2018.
[23] M. Al Borno, M. de Lasa, and A. Hertzmann, "Trajectory optimization for full-body movements with complex contacts," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 8, pp. 1405–1414, Aug. 2013.
[24] D. Han, H. Eom, J. Noh, and J. S. Shin, "Data-guided model predictive control based on smoothed contact dynamics," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 533–543, May 2016.
[25] L. Liu, M. V. D. Panne, and K. Yin, "Guided learning of control graphs for physics-based characters," *ACM Trans. Graph.*, vol. 35, no. 3, pp. 1–14, Jun. 2016.
[26] L. Liu, K. Yin, M. Panne, and B. Guo, "Terrain runner: Control, parameterization, composition, and planning for highly dynamic motions," *ACM Trans. Graph.*, vol. 31, p. 154, Nov. 2012.
[27] 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, pp. 143–157, Jul. 2018.
[28] C. Sung, T. Kagawa, and Y. Uno, "Whole-body motion planning for humanoid robots by specifying via-points," *Int. J. Adv. Robotic Syst.*, vol. 10, no. 7, p. 300, Jul. 2013.
[29] J.-I. Choi, S.-J. Kang, C.-H. Kim, and J. Lee, "Virtual ball player," *Vis. Comput.*, vol. 31, nos. 6–8, pp. 905–914, Jun. 2015.
[30] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu, "Sampling-based contact-rich motion control," in *Proc. ACM SIGGRAPH Papers*, Jul. 2010, p. 128.
[31] M. Romeo, C. Monteagudo, and D. Sánchez-Quirós, "Muscle and fascia simulation with extended position based dynamics," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 134–146, Feb. 2020.
[32] R. Yu, H. Park, and J. Lee, "Figure skating simulation from video," *Comput. Graph. Forum*, vol. 38, no. 7, pp. 225–234, Oct. 2019.
[33] P. Khungurn and D. Chou, "Pose estimation of anime/manga characters: A case for synthetic data," in *Proc. 1st Int. Workshop coMics Anal., Process. Understand.*, Dec. 2016, pp. 1–6.
[34] J. Chemin and J. Lee, "A physics-based juggling simulation using reinforcement learning," in *Proc. 11th Annu. Int. Conf. Motion, Interact., Games*, Nov. 2018, pp. 1–7.
[35] H. Park, R. Yu, and J. Lee, "Multi-segment foot for human modelling and simulation," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 637–649, Feb. 2020.
[36] B. Lange, A. Rizzo, C.-Y. Chang, E. A. Suma, and M. Bolas, "Markerless full body tracking: Depth-sensing technology within virtual environments," in *Proc. Interservice/Ind. Training, Simulation, Educ. Conf. (I/ITSEC)*, 2011, pp. 1–8.
[37] L. E. Potter, J. Araullo, and L. Carter, "The leap motion controller: A view on sign language," in *Proc. 25th Austral. Computer-Hum. Interact. Conf., Augmentation, Appl., Innov., Collaboration*, Nov. 2013, pp. 175–178.
[38] E. Sifakis, I. Neverov, and R. Fedkiw, "Automatic determination of facial muscle activations from sparse motion capture marker data," in *Proc. ACM SIGGRAPH Papers*, Jul. 2005, pp. 417–425.
[39] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *Proc. ACM SIGGRAPH Classes*, Aug. 2008, pp. 1–10.
[40] 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. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, Jul. 2002, pp. 491–500.
[41] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal character animation with continuous control," in *Proc. ACM SIGGRAPH Papers*, Jul. 2007, p. 7.
[42] S. Clavet, "Motion matching and the road to next-gen animation," in *Proc. GDC*, 2016, p. 4.
[43] M. Da Silva, Y. Abe, and J. Popović, "Simulation of human motion data using short-horizon model-predictive control," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 371–380, Apr. 2008.
[44] D. Han, J. Noh, X. Jin, J. S. Shin, and S. Y. Shin, "On-line real-time physics-based predictive motion control with balance recovery," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 245–254, May 2014.
[45] Y. Lee, S. Kim, and J. Lee, "Data-driven biped control," in *Proc. ACM SIGGRAPH Papers*, Jul. 2010, pp. 1–8.
[46] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," in *Proc. ACM SIGGRAPH Papers*, Jul. 2010, pp. 1–8.
[47] S. Barrett, K. Genter, T. Hester, M. Quinlan, and P. Stone, "Controlled kicking under uncertainty," in *Proc. 5th Workshop Humanoid Soccer Robots Humanoids*, 2010, pp. 1–6.
[48] T. Hester, M. Quinlan, and P. Stone, "Generalized model learning for reinforcement learning on a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2369–2374.
[49] L. Leottau, C. Celemin, and J. Ruiz-del-Solar, "Ball dribbling for humanoid biped robots: A reinforcement learning and fuzzy control approach," in *Robot Soccer World Cup*. Cham, Switzerland: Springer, 2014, pp. 549–561.
[50] S.-J. Yi, S. McGill, and D. D. Lee, "Improved online kick generation method for humanoid soccer robots," in *Proc. 8th Workshop Humanoid Soccer Robots*, 2013, pp. 1–6.
[51] S. Jain and C. K. Liu, "Interactive synthesis of human-object interaction," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, Aug. 2009, pp. 47–53.
[52] K. Ding, L. Liu, M. van de Panne, and K. Yin, "Learning reduced-order feedback policies for motion skills," in *Proc. 14th ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, Aug. 2015, pp. 83–92.