3D Res (2017) 8:18
DOI 10.1007/s13319-017-0124-0

3DR EXPRESS

Performance-Driven Hybrid Full-Body Character Control for Navigation and Interaction in Virtual Environments

Christos Mousas · Christos-Nikolaos Anagnostopoulos

Received: 26 December 2016 / Revised: 18 February 2017 / Accepted: 17 March 2017 / Published online: 4 May 2017

Abstract This paper presents a hybrid character control interface that provides the ability to synthesize in real-time a variety of actions based on the user's performance capture. The proposed methodology enables three different performance interaction modules: the performance animation control that enables the direct mapping of the user's pose to the character, the motion controller that synthesizes the desired motion of the character based on an activity recognition methodology, and the hybrid control that lies within the performance animation and the motion controller. With the methodology presented, the user will have the freedom to interact within the virtual environment, as well as the ability to manipulate the character and to synthesize a variety of actions that cannot be performed directly by him/her, but which the system synthesizes. Therefore, the user is able to interact with the virtual environment in a more sophisticated fashion. This paper presents examples of different scenarios based on the three different full-body character control methodologies.

Keywords Character animation · Hybrid controller · Navigation · Object manipulation · Virtual reality interaction

1 Introduction

Recently, with the rapid development of low-cost motion capture systems, such as Microsoft's Kinect [1] and Asus Xtion [2], as well as the development of various games that use these technologies, users can interact directly with the virtual environment. These solutions enable more sophisticated interaction with the virtual environment compared to the basic controllers that videogames use [3]. This is because the users can use their body poses to interact with the environment. Therefore, an increment of the user's immersion in the environment has been provided with the means to take part in events that evolve within the environment.

Conversely, even if the user is able to interact with various tasks, the basic disadvantage of existing applications that use motion capture technologies is that the system can synthesize either predefined actions or to provide direct manipulation of character's body parts. When using predefined gestures or body activity, the system recognizes the user's activity

Electronic supplementary material The online version of this article (doi:10.1007/s13319-017-0124-0) contains supplementary material, which is available to authorized users.

C. Mousas (✉)
Department of Computer Science, Southern Illinois University, Carbondale, IL 62901, USA
e-mail: christos@cs.siu.edu

C.-N. Anagnostopoulos
Department of Cultural Technology and Communication, University of the Aegean, 81100 Mytilene, Greece
e-mail: canag@ct.aegean.gr

and synthesizes the desired motion. The ability to synthesize only a limited number of motions results in restrictions to the user's actual intention. For this reason it is assumed that an enhancement of the actions that the user is able to perform can be beneficial. It will ensure that the user is able to perform and that the system can synthesize not only the predefined motions, but also a variety of freeform actions.

Based on these requirements, this paper presents a character control interface that provides the user with the ability to directly manipulate the virtual character or to use their body actions to synthesize the desired motion sequences. Considering the variety of actions that can be performed by the user, as well as a variety of motions that cannot be synthesized directly by the system, a novel hybrid character control interface is introduced. This hybrid controller lies within the activity recognition and the direct manipulation methodologies. The activity recognition process communicates with an animation controller that is responsible for animating the virtual character based on a number of motion sequences (mostly of them are related to locomotion). Moreover, the direct manipulation allows the user to manipulate specified body parts of the virtual character based on the performance capture process. The hybrid controller allows the user to perform different actions that are contained in the database of motions, and to enhance these synthesized actions by directly manipulating specific body parts of the user to a virtual character. An example of the aforementioned control method is the ability of the system to recognize a walking motion in conjunction with the user's ability to wave its hand during the character's locomotion (see Fig. 1). We believe that such a solution could be ideal for virtual reality related applications that require navigation, interaction and/or manipulation of objects [59].

To achieve this hybrid controller, it is necessary to have an efficient methodology that will be able to determine whether the user intends to perform an action and whether the user intends to manipulate specific body parts of the character. In the proposed methodology, the system satisfies both of these requirements simultaneously. This is achieved as follows. Firstly, by analyzing small amounts of motion capture data based on its motion features, the necessary patterns are defined. Secondly, by using a searching algorithm, all possible motions-joints combinations are found and are stored in a database. Finally, based on a searching algorithm that is implemented for the purpose of the proposed methodology, the system returns the body parts of the character that are manipulated by the user and those that are manipulated by the motion controller.

In addition to the hybrid controller that is presented in this paper, the user should be able to interact with tasks, objects and the environment. Hence, in the proposed methodology, additional parameters that influence the motion synthesis process are implemented in an action controller that is attached to the character. These parameters are responsible for keeping information about the task that the character performs, allowing interactions with the environment and objects that are located within it. Thus, based on the aforementioned action controller, different examples are presented in this paper.

The remainder of the paper is organized in the following way. In Sect. 2, related work on animation techniques is presented. In Sect. 3, an overview of the proposed methodology is presented. The methodology

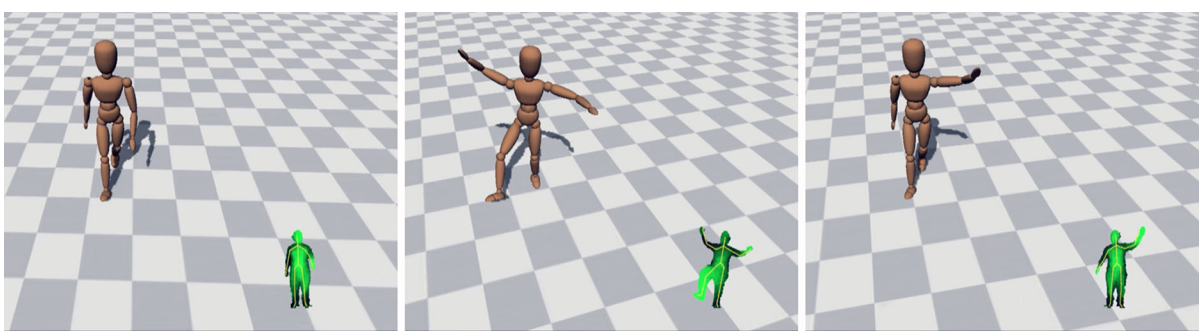

Fig. 1 The three different character animation controllers by which the proposed methodology can synthesize: a walking motion based on the motion controller (*left*), manipulation of character's body based on the direct manipulation controller (*middle*), and a walking motion in combination with a hand wave motion based on the hybrid controller (*right*)

that is used to generate the hybrid motion controller is presented in Sect. 4. Examples of scenarios that are developed with the proposed methodology are presented in Sect. 5. The implementation and the results obtained when evaluating the proposed methodology are presented in Sect. 6. Finally, in Sect. 7 the conclusions are drawn and potential future work is discussed.

2 Related Work

There are various ways to animate a virtual character, but the three most common are data-driven, kinematics and physics techniques and there has been extensive research in these areas [4, 5]. Interactive character control can also be separated according to the input device that is used for the character animation process [6]. In general, the character controller can be a standard input device, such as a keyboard and joystick such as [7]. Alternatively, it can be more specialized such as text input [8–10, 60], prosodic features of speech [11], sketch-based interfaces [12, 13] or the body) [14–16] or the body part [17] of a user (performance capture), when its motion is captured using motion capture technologies. The methodology presented in this paper lies in the field of data-driven motion synthesis, on activity recognition and on performance animation techniques, since the system permits one to either synthesize the motion of the character based on an activity recognition methodology or to manipulate the character's body according to the user's performance. Therefore, work related to those techniques is presented in the following paragraphs.

In data-driven techniques, the existing motion data that is contained in a database is used to animate the virtual character. Hence, various methodologies to interpolate [18, 19], blend [20, 21], splice [22–24], warp [25], retarget [26], and so on, enable one to synthesize a wide range of actions that the character can perform. In the methodology presented, a simple locomotion controller was built. It is similar to the one that was developed in [27], which is responsible for animating the character's locomotion. However, instead of using keyboard inputs to manipulate the character, a methodology is introduced that estimates the character's motion by capturing the user's performance.

Conversely, performance animation techniques, which are also known as computer puppetry [28, 29], manipulate body parts by using kinematics solutions [16, 30], or recognize the performer's action (activity recognition) and display the motion from a database [16, 31–34], or synthesize a new motion sequence by using the existing motion data that is contained in a database (motion reconstruction) [15, 35–38]. In these three different approaches, the input signals that are retrieved from a motion capture devices are used as the main parameters for the motion synthesis process. Hence, for animating the virtual character, methodologies that use accelerometers [31, 35, 39] or optical motion capture devices [40, 41] provide the desired control parameters for the system. Among other subjects, the research community has focused on the ability to synthesize as naturally as possible motion sequences when using a reduced number of inputs. Hence, solutions that use six [42] or even two [42] input parameters are able to reconstruct the motion of the character in real-time. These methodologies, which generally are based on a statistical analysis of existing human motion data [15, 43], map the reduced input parameters of a database of postures to find and synthesize those that are most appropriate.

Less attention has been given to a methodology that will be able to combine the activity recognition and the direct manipulation of the character's body parts by using input parameters provided by a motion capture device. Ishigaki et al. [44] and Seol et al. [45] proposed solutions that are closest to what this paper presents. In [44] by developing an activity recognition methodology, the system is able to recognize and then synthesize the motion of a character. Additionally, since the character is able to interact with various tasks a physics-based controller that uses an inverse pendulum model provides physical valid postures of the character when its lower body does not make contact with the ground. In the aforementioned method, the character is able to perform a motion based either on the motion controller or on a physics simulation controller. In [45] an activity recognition methodology enforces the synthesis of various actions contained in a database. Moreover, the system is able to blend different actions, such as enhancing the actual number of the motions that can be synthesized. In such case, the synthesized motions are always dependent on the motion sequences contained in a database. This means

that a user is limited to performing existing actions or combinations of actions instead of synthesizing actions by directly using its body. Hence, the hybrid way of controlling and synthesizing the motions of a character, which provides an enhancement of the actual actions space by combining simultaneously the activity recognition and the direct manipulation of user's body parts to the character, is the main advantage of the presented method.

3 Overview

The proposed methodology can recognize a variety of actions and map these actions to the character's controller. It offers a technique to recognize the user's activity during the performance capture process. Moreover, the presented system recognizes the user's intention to perform hybrid actions. During the application runtime, the system computes the likeliest way to control the character. This is based on a searching algorithm that returns the character control module. The system returns either a single motion that animates the character (motion controller), or the combination of joints that are manipulated by the user in conjunction with the corresponding motion that the user mimics (hybrid controller), or the whole number of joints giving the chance to the user to manipulate the whole body of the character based on its activity (direct manipulation controller). The aforementioned methodology that describes the ways that the user controls the character appears in Fig. 2.

4 Hybrid Character Controller

In this section, four basic steps were introduced to achieve the hybrid character controller. Firstly, the existing motion data that animates the character is analyzed to provide the necessary patterns, such as enabling an efficient activity recognition method. Secondly, the character's joints are assigned with semantic labels. Thirdly, all possible combinations of the joints-motions are pre-computed, according to the joints' semantic labels. Finally, by using the proposed searching methodology, the system estimates the corresponding controller mechanism that animates the character.

4.1 Computing Motion Features

To generate an activity recognition methodology, the motion data that animates the virtual character is analyzed based on its features. With this analysis process it is possible to execute the necessary patterns of human motion. Then, it is possible to develop an efficient methodology that enables different users to perform a predefined action and the system to recognize it without using a time-consuming calibration process.

For each of the motion sequences, as well as for each of the character's joints that is related to a specific action, the position $\bar{p}_{i,j}$ and velocity $\bar{v}_{i,j}$ features are computed according to:

$$\bar{p}_{i,j} = \frac{1}{N}\sum_{t=1}^{N} \|p_{i,j}(t)\| \quad \text{where} \quad \mathrm{i} = 1,\ldots,\mathrm{I};\ \ \mathrm{j} = 1,\ldots,\mathrm{J} \tag{1}$$

$$\bar{v}_{i,j} = \frac{1}{N}\sum_{t=1}^{N} \|v_{i,j}(t)\| \quad \text{where} \quad \mathrm{i} = 1,\ldots,\mathrm{I};\ \ \mathrm{j} = 1,\ldots,\mathrm{J} \tag{2}$$

where i and j denote respectively the motion index of the corresponding joint, and N denotes the number of frames of each of the motions. I and J denote repressively the total number of motions and the total number of joints of a character that were used for the pre-computation process (see Sect. 4.3). Each of these features is computed according to the root position, and velocity. Based on these features, each of the motion sequences s_i contained in the database is represented as $\bar{s}_i = \{(\bar{p}_i, v_{i,n}), \ldots, (\bar{p}_{i,n}, v_{i,n})\}$ where n denotes the n-th joint of the character.

4.2 Joint Characterization

In the second step, the joints of the character that are used for the activity recognition process are assigned with semantic labels. Two different types of labels are defined. They are the active and the inactive, and the constrained and the unconstrained joints. An active joint denotes a joint that is used for the pre-computation process (see Sect. 4.3) as well as being used during the activity recognition process. A typical example is the joints of the character's wrists. An inactive joint is not used for the pre-computation process, although, depending on the second

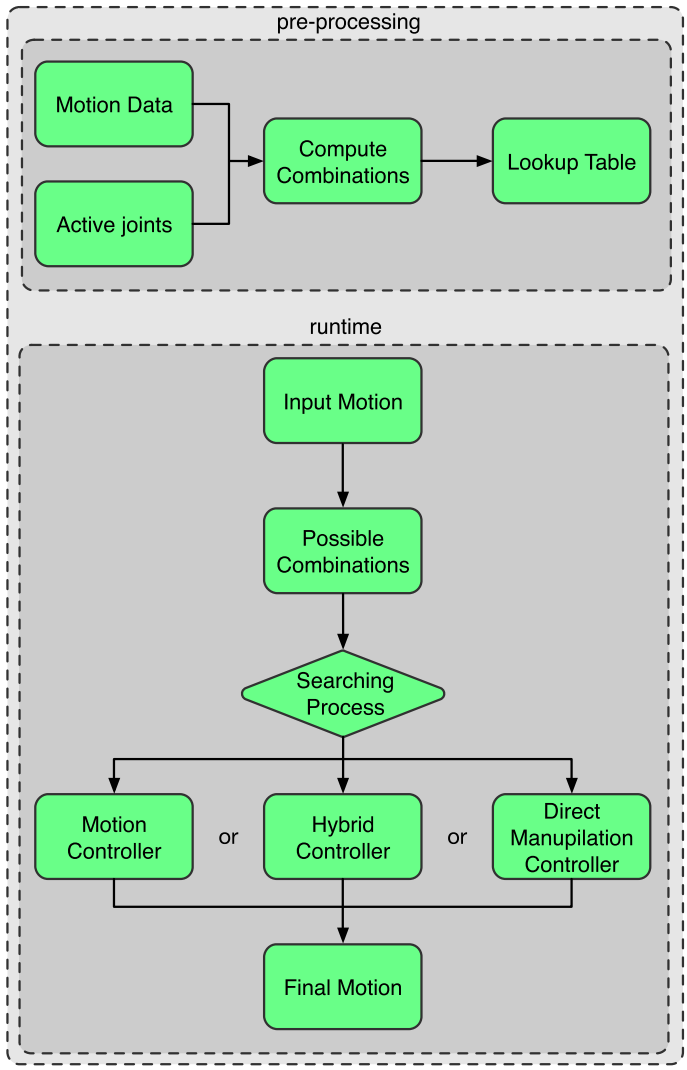

Fig. 2 The pre-processing and the runtime stages of the presented methodology. During the runtime, the system computes the controller mechanism that best matches to the user's activity

characterization (constrained or unconstrained), it may be used for the activity recognition process. Hence, typical examples are the joints in the feet, which even if they are assigned as inactive are used for the activity recognition process since they are constrained joints. A constrained joint denotes a joint that should stay static (i.e., that is in contact with the floor). Unconstrained joints are those that are permitted to move during the activity recognition process.

A simple example illustrates how each of those two different types of joint is used for the activity recognition process. Consider a walking motion where the performer uses his upper-body for the activity recognition process. In this case, the user's upper-body joints, those retrieved from the motion capture device, are designated as active, whereas the user's lower body joints are designated as inactive. Moreover, the user's feet joints (end-effectors) are designated as constrained, whereas the remainders are designated as unconstrained. Based on this representation the system can recognize a walking motion if, and only if, the user's feet are in contact with the ground. The user could perform a walking motion by using the upper-body, and also edit the motion during the runtime i.e., to stoop using their legs during the walking motion to avoid ceiling constraints (see Fig. 4). Finally, it should be mentioned that, like the walking motion, different joints of each of the actions that the character can perform are designated as active/inactive and constrained/unconstrained (see Sect. 6.1).

4.3 Pre-computing Possible Intentions

Because the user must be able to manipulate either the character's entire body or specific body parts of it, in the presented methodology it is necessary to perform a pre-computation step for each possible combination between the different motions and the active joints of the character. This pre-computation allows the direct estimation of the user's possible intention to interact with the character. Moreover, this allows the proposed methodology to estimate directly the motion that the user mimics and the body part that the user's body should manipulate.

By having a number of joints designated as active, all possible combinations of joints that can be used for the activity recognition process must first be computed. Thus, having a set of joints k designated as active, by using the powerset algorithm [46, 47] every possible combination $P(k)$ of those joints is found, including an empty set and k, which is the whole set, itself. Generally, the powerset algorithm provides the ability to automatically compute all possible combinations among a number of components. The benefit of such pre-computation is quite important in the presented methodology since it is possible to compute automatically all possible combinations when different number of motions and joints are used.

To illustrate the way that the system computes all the possible combinations between the motion data and the active joints of the character a simple explanation follows. Firstly, a number of motion sequences m_i are considered, where $i = \{1, \ldots, K\}$ and K is the total number of motions. Then, a number of active joints is considered, $j_{i,n}$, where $n = \{1, \ldots, N\}$ and N denotes the total number of active joints that belongs to the i-th motion sequence. Based on this representation the system computes all possible combinations between each motion m_i and each joints $j_{i,n}$ by assigning this process to the powerset algorithm. It should be mentioned that, for a number of joints μ and a number of motion sequences v, the number of combinations that the system generates are computed according to $v \times 2^{\mu} - (v - 1)$. The $(v - 1)$ part removes the repeated empty set, which needs to be computed only once, since the empty set is the same for all possible motions v taking part in the pre-computation process. All of those possible combinations are stored in a lookup table and the system is able to search for the most probable combination, as presented in the following subsection (Sect. 4.4).

4.4 Intention Recognition

The final step of the motion controller is to return the most probable character control module. This recognition process is presented in the remainder of this subsection.

At each time instance, during the application runtime, the motion capture device captures the user's motion. Similarly, for the motion data contained in the database, the motion features of the input motion are computed according to Eqs. 1 and 2. Therefore, the input motion segment is represented as $\check{s} = \{(\check{p}_1, \check{v}_1), \ldots, (\check{p}_n, \check{v}_n)\}$, where n denotes the joint of the performer as retrieved from the motion capture device. Thus, for each of the combinations of joints $P(k)$, resulting from the powerset algorithm, the system computes the position and velocity distance functions of the existing pre-computed combination of the character $\bar{s}_i$ contained in the database and the input motion segment $\check{s}$ as:

$$d_p = \frac{1}{M} \sum_{m=1}^{M} \|\check{p}_m - \bar{p}_{i,m}\|^2 \tag{3}$$

$$d_v = \frac{1}{M} \sum_{m=1}^{M} \|\check{v}_m - \bar{v}_{i,m}\|^2 \tag{4}$$

where m denotes the number of joints that belong to the closest combination of joints that is computed by the powerset algorithm.

Having computed d_p and d_v it is now possible to estimate the most probable intention of the user. This can be achieved by using a nearest neighbor search [48]. A drawback of using the nearest neighbor search process, is its inability to compute the empty set (the way in which the character's body is directly manipulated by the user) since the 1/M part of Eqs. 3 and 4 does not provide a result when $M = 0$. For that reason, threshold values are established for each feature, (p_{thres} for the joints position, and v_{thres} for the joints velocity), that are equal to the maximum differences computed by the exiting motion data. This maximum difference is computed by $\max(p_{k,i} - p_{k,j})$ and $\max(v_{k,i} - v_{k,j})$ for the position and velocity feature respectively, where i and j denote the frames in the motion

sequences and k the character's corresponding joint. It should be noted that different threshold values are established for each different active joint as well as for each different motion that animates the character.

During the runtime of the presented methodology, the searching for finding the most valid action of a character is achieved by iterating through all possible combinations contained in our dataset. According to Eqs. 3 and 4 the position and velocity distances between the input motion and every possible combination are computed as $distance(\check{p}, p_i)$ and $distance(\check{v}, v_i)$. The resulted distances are stored in the corresponding arrays (P for the position and V for the velocity). Having computed the aforementioned distances, the proposed algorithm (see Algorithm 1) evaluates each component that belongs to and in order to find the combination that provides the minimum position and velocity distance simultaneously. Finally, since the input motion of a user should be within the threshold values, the following condition is established (see Eq. 5) to determine whether the user intends to directly manipulate (DM) the character or to animate the character by either using the hybrid (HC) or the motion controller (MC):

$$C = \begin{cases} HC \text{ or } MC & \text{if } (d_p \leq p_{thres} \;\&\&\; d_v \leq v_{thres}) \\ DM & \text{otherwise} \end{cases} \tag{5}$$

The direct manipulation controller, denoted by the empty set, is selected when all of the active joints are out of their thresholds. The motion controller is generated when the active joints are within their threshold. The hybrid controller is generated when a partial number of active joints outside their thresholds. As can be seen, both the hybrid and the motion controller are estimated similarly. The distinction between these two controllers is based on the result that is provided by the proposed algorithmic implementation of the searching process as presented in Algorithm 1. Given a $index \in [0, combinations.size()]$, where $combinations.size()$ is the total number of combinations as pre-computed according to the powerset algorithm, when the $index \leq combinations.size()$, a hybrid or a motion controller is returned. This depends on the index search in the lookup table (*combinations*) to find the resultant joint. An *index* is a value of the table that stores all possible combinations. In the lookup table that implemented each index row of the *combinations* is represented as a group of joints that are animated by the motion controller, g_{MC}, and a group of joints that are animated by the user, g_U, such as $combinations[index] = \{g_{MC}, g_U\}$. For different index, which means for different columns of the lookup table, different g_{MC} and g_U are used for animating the character. When the g_U is empty, the system animates the character based only on the motion controller, since all the character's joints are animated by the existing motion data. Similarly, the hybrid controller is generated when both the g_{MC} and g_U are not empty of joints. Conversely, when the $index = 0$, direct manipulation of the character's joints provides the best match of the input motion, since the powerset algorithm returns the empty set as the first possible combination during the pre-computation process. Therefore, the character is animated solely by the user. Hence, with the methodology presented, the system simultaneously recognizes the action that animates the character based on the existing motion data and the user's intention to perform a hybrid action by searching directly in a lookup table to find the combination that is assigned to the index number.

Input: The motion features of the user's posture ($\check{p}$, $\check{v}$), the example motion features (p_i, v_i) in accordance with the associated threshold values ($pThres_i$, $vThres_i$) as computed for each motion contained into the database.
Output: The `index` that represents the character manipulation module.

```
for (i=0 to C.size()) do
    P ← distance(p̌, p_i);
    V ← distance(v̌, v_i);
    if (i>1) then
        for (j=0 to P.size()) do
            if (P_j < P_0 && V_j < V_0) then
                P_0 ← P_j;
                V_0 ← V_j;
                if (P_0 ≤ pThres_j && V_0 ≤ vThres_j)
                then
                    index = j;
                else
                    index = 0;
                end
            end
        end
    end
end
return index;
```

An alternative to the aforementioned methodology is based upon the ability to select body parts that have differences between input and pre-recorded larger than the thresholds. In this case, such a methodology even if it is able to provide a solution, may result in a synthesized motion that is not the desired one. This is because the system is unable to recognize only a single action at each time step. More specifically, considering a user that performs with their right hand a walking motion and with their left hand a running motion, the system is unable to estimate which is the best match that animates the character. With the presented precomputation process, as well as with the proposed algorithmic searching process the system is able to estimate only a single motion at every iteration of the system.

5 Controller Interaction Paradigms

To demonstrate the efficiency of the proposed methodology, different interaction paradigms were implemented. Each of the paradigms that are presented in the following subsections was developed on the basis of the different actions that the system can synthesize due to the character controller's three different modules. Moreover, an action controller was developed in order to enhance the actual actions that the user is able to perform through the character. Specifically, the action controller has the following parameters:

- The character controller C is based on the activity recognition process, which estimates the motion of the character as presented in Sect. 4, enabling all possible controller states.
- The environment constraints E that characterize the environment and prevent the system from synthesizing incorrect motion (e.g., a collision with objects and walls, falling down to the ground, etc.).
- The behavior B of the objects located in the three-dimensional environment that allows the user to interact with them. B receives values $B \in \{true \texttt{ or } false\}$ indicating whether the object should follow the character's hand. Moreover, a threshold value is inserted between the character's hands and the object to enable/disable the interaction.

Thus, based on the aforementioned states, the action controller takes the following form:

$$A = \{C, E, B\} \tag{6}$$

Any state that does not participate in the examined scenario, are filled as "empty". Based on this representation of the action controller that is attached to the virtual character, the following subsection presents different examples. Finally, it should be noted that the examples presented below also appear in the accompanying video.

5.1 Locomotion Synthesis

By using their body, the user is able to manipulate the character (see Fig. 3). Moreover, additional environmental constraints are placed on the environment. The user avoids ceiling constraints (Fig. 4) that are located in the three-dimensional environment, by deforming the displayed motion. This deformation is achieved by using the inverse kinematics solver proposed by Kalmann [51], constraining the feet end-effector to remain on its position when the character's hips move down. There are also data-driven motion deformation techniques such as [53] that can be used in the proposed methodology, though a simple kinematics deformation was considered in the current implementation. In these examples the user drives the character by using two control parameters the C and E.

5.2 Objects Manipulation and Interaction

In this case two different scenarios were implemented that illustrate the flexibility of the system in estimating a motion sequence when using different active joints. In the first scenario (see Fig. 5) the user manipulates the character by using its upper body. In the second scenario (see Fig. 6) the user can manipulate the character by using either its upper or lower body. In this approach, when the user is required to animate the character based on its lower body, only one foot must be constrained at a time. This is achieved by assigning, to the character controller state, C, an indicator to tell which foot (left or right) is constrained, such as $F \in \{R \texttt{ or } L\}$. Finally, in those scenarios, the user can interact with objects that are located in the environment. Hence, the action controller takes C, E and B as parameters.

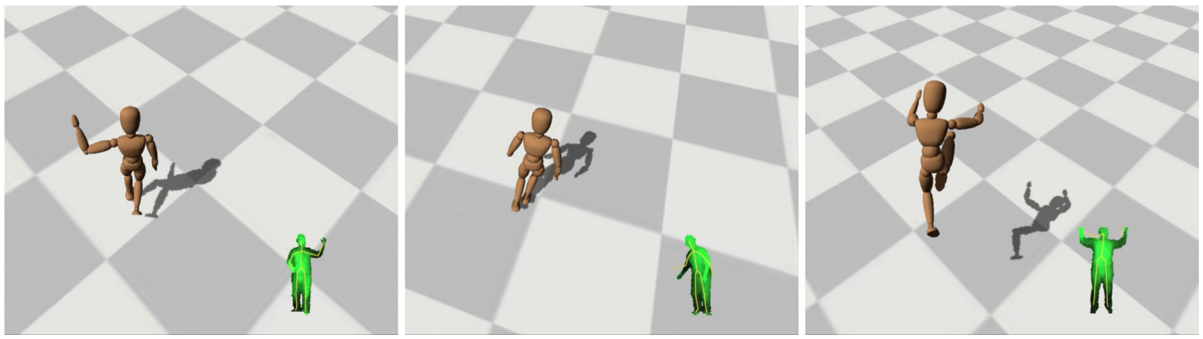

Fig. 3 The user drives the character within a virtual environment by using the motion controller or the hybrid controller. Walking when waving the hand (*left*), running when turning right (*middle*), and jumping (*right*)

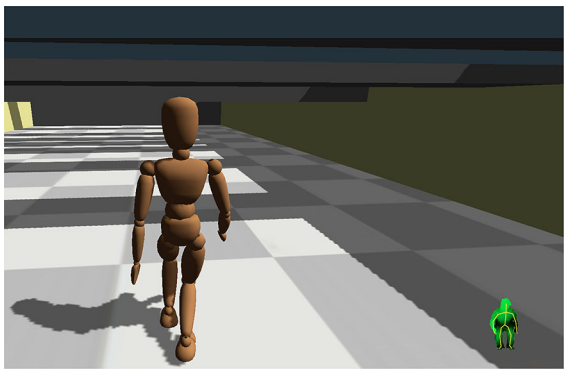

Fig. 4 The character avoids ceiling constraints when stooping with its legs

6 Implementation and Evaluation

This section presents the process of implementing and evaluating the presented methodology and the results obtained.

6.1 Implementation

To implement the proposed methodology, Microsoft's Kinect [1] motion capture device and its associated SDK that builds upon Shotton et al. [49] were used to capture the user. The motion features of the performer that are used for the activity recognition process are computed based on the skeletal information provided by the Microsoft's Kinect SDK. The presented methodology was implemented on an Intel i7 with 8 GB of memory. For the activity recognition process windows of different sized were tested during our development process varying from 16 to 512 ms. We found that a window with size 256 ms is able to respond fast as well as to provide high recognition rate. The human skeletal data that was used to animate the character were downloaded from the CMU motion capture library [50] and are represented in a acclaim skeleton file (ASF) format. The human skeleton model contains 24 joints. The use of all of these joints for the motion estimation process results in a high number of computations for the system (see Sect. 6.2.1). Thus, the basic controller that implemented a limited number of joints designated as active for each of the different motions is presented in Fig. 7. Moreover, during the application's runtime, especially when the system recognizes the hybrid controller, the analytical inverse kinematics solver proposed by Kalmann [51] was used to handle the character's joints. Finally, the input skeletal dimension, as captured by Microsoft's Kinect, is mapped to the character's skeletal dimension by using the methodology proposed in [43]. This methodology provides the ability for users with anthropometry differences (i.e., when users are either adults or children) to control the character without the need of processing the generated patterns that are used for the activity recognition process.

For the motion controller that animates the character, a small number of motions were used. They are: idle, walking (forward, right and left), running (forward, right and left), and jumping (low and high) motions. This small number of motions is sufficient since they are the basic movements of the character that need to be manipulated within a virtual environment. The remaining actions that a user may request can be synthesized by him/her by using the direct manipulation or the hybrid controller. The number of different motions and the number of frames for each motion that were used for animating the character as well as for the activity recognition

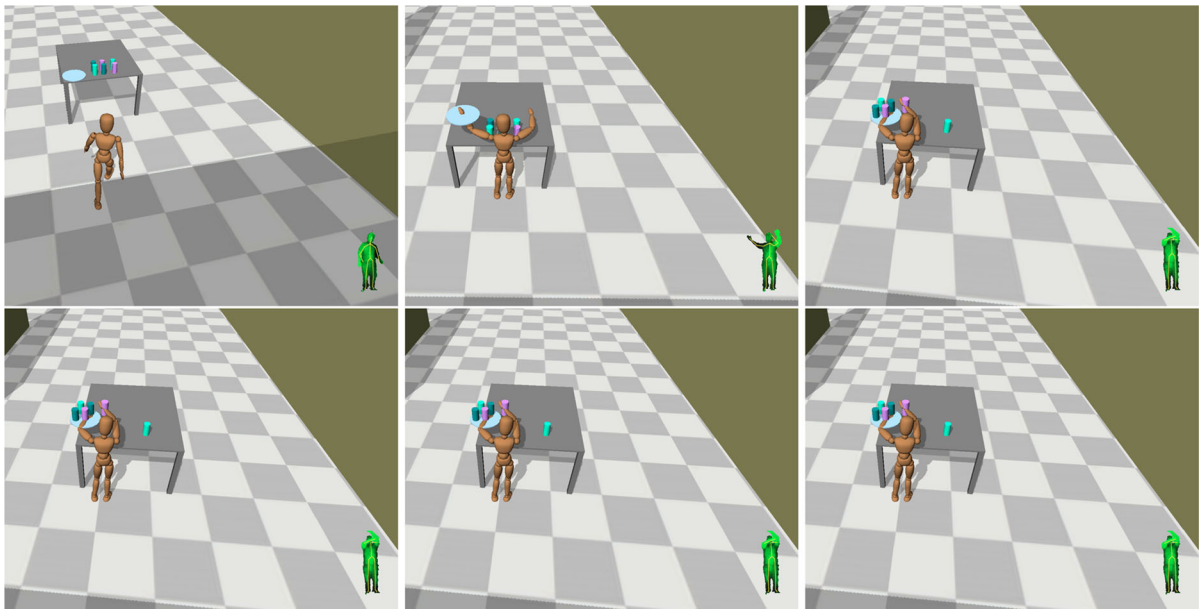

Fig. 5 By means of the character, the user can manipulate objects that are located in a three-dimensional environment. In these examples, the character is animated by the motion, the hybrid and the direct manipulation controllers

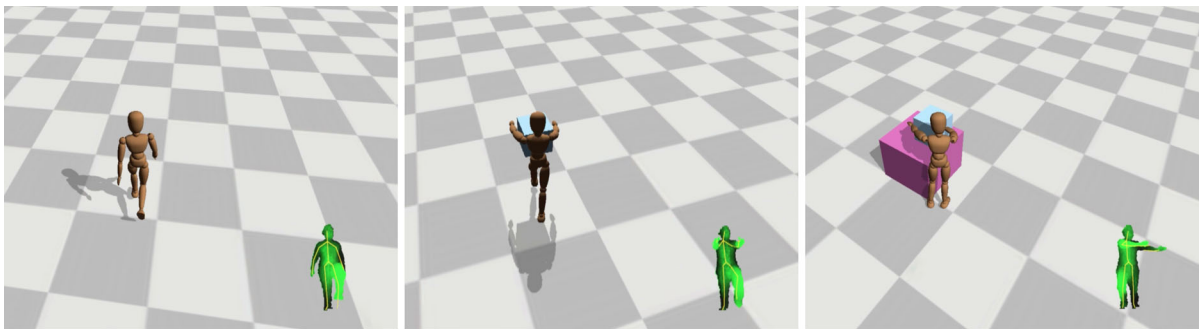

Fig. 6 By extending the controller, the user is able to manipulate the character by using its lower body to complete the task

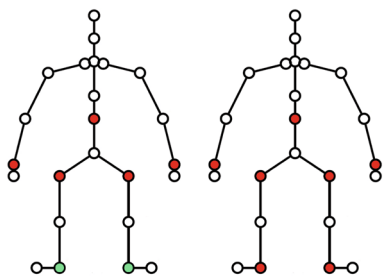

Fig. 7 The active (*red* color) and constrained (*blue* color) skeletal joints that were used for the motion recognition process for **a** idle, walking, and running, and **b** jumping. The rest of the character's joints (*white*) are classified as inactive and unconstrained. (Color figure online)

process are represented in Table 1. Moreover, based on the powerset algorithm in the presented implementation for the motion estimation process by using a number of active joints and a number of motions , the system computes a maximum of 472 possible combinations (the repeated empty sets are removed). This maximum number of possible combinations results from the number of active joints that are used for each of the actions that are presented in Table 1. In cases where recognition of a greater number of motion sequences and a greater number of joints for the activity recognition process may be required, respectively the number of possible combinations will be greater.

6.2 Evaluations

This section presents the results obtained when evaluating the proposed methodology. Specifically, three different evaluation processes were conducted. Firstly, the frame rate of the methodology was evaluated when a greater number of joints were used for the activity recognition process. Secondly, the accuracy of the system in estimating the desired action was evaluated. Finally, the presented methodology was evaluated by real users, indicating the efficiency and simplicity of the presented character control interface.

6.2.1 Performance Evaluation

Different numbers of joints were used for the character controller recognition process in the first evaluation process. For each different number of joints, all possible combinations of motions-joints were pre-computed, as were those resulting from the power-set algorithm. Then, for each of the combinations, the methodology's frame rate was computed. The results obtained from this evaluation process are summarized in Fig. 8. The proposed methodology works for interactive frame rates (22 fps) even if 8192 (2^{13}) combinations are computed. However, as the number of possible combinations increases, the methodology's frame rate decreases. It was found that, when 16384 (2^{14}) possible combinations are used, the system's frame rate reaches 9 frames per second with latency equaling to 0.615 s. We believe that making the application work in real-time at such a low frame rate may negatively affect the users. As a result, in the current implementation, the 472 possible combinations provide quite a high frame rate of approximately 104 frames per second with a latency equaling to 0.025 s.

6.2.2 Evaluating Activity Recognition

In the second evaluation, the estimation rate of correctly recognizing each of the actions that the character is able to perform was computed. This evaluation was conducted by using the leave-one-out cross validation method. The existing motion data was split into small segments. Then the motion features and the corresponding threshold values that characterize the remaining motion sequences for each of the actions were computed again. Thus, by using the small motion segment as reference input motions, the motion features that characterize the segment were computed and designated as input parameters for the proposed searching process described in Algorithm 1. This procedure was repeated for each motion sequence. Thus, a class confusion matrix that illustrates the allocation of the estimation rate, when using the active joints that were used in the presented methodology, is presented in Fig. 9. As can be seen, the proposed methodology seeks to estimate a higher rate of each of the actions contained in the database, leading to estimates that are 95.5% correct. It should be noted that the presented results obtained by using a 256 ms window. Comparing with different window sizes (13, 32, 64, 128, 256, and 512 ms,) the 256 ms window provides the optimal results.

Even if the cross validation showed that such a methodology is able to estimate quite efficiently each action, there are issues related to the motion capture system that were used. Specifically, in cases that any

Table 1 The number of all different motions that belong to a specific action and the number of frames for each of the motions that animates the character, such as those used in the presented methodology

Motion subject	Number of motions	Number of frames	Active joints	Possible combinations
Idle	1	45	5	32
Walking	3	81	5	96
Running	3	66	5	96
Jumping	2	52	7	256
Total	9	244	19	472 (480)

Also, the number of active joints that are used separately for each motion in the activity recognition process and the complete number of joint-motion combinations in the current implementation

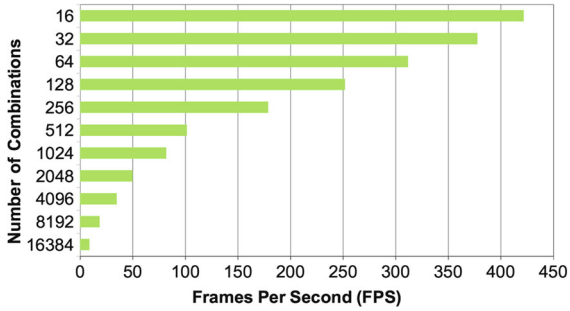

Fig. 8 The results obtained when evaluating the methodology's frame rate

body part of the user is hidden by another body part, the system may estimate a wrong action. Hence, it is assumed that a better motion capture device may be able to eliminate the wrong estimations. Moreover, considering that the activity recognition process works with a limited number of analyzed data, the results provided by the presented methodology are quite reasonable. In cases when a better estimation rate is required, either a personalization process, or an analysis of a larger amount of motion data may be quite beneficial. However, the last suggestion may negatively affect the estimation process, since intersections between different actions may appear. In this event, even if the examined motion features show that they are able to provide the desirable results, in cases

motion type

segment taken out of the database	walk forward	walk right	walk left	run forward	run right	run left	jump low	jump high	idle
walk forward	96.1	0.6	0.5	1.4	0.4	0.4	0.3	0.3	0.3
walk right	1.9	94.5	0.9	0.3	1.8	0.2	0.2	0.2	0.2
walk left	2.1	0.9	94.4	0.5	0.3	1.3	0.2	0.3	0.3
run forward	2.4	0.6	0.4	94.3	0.8	0.9	0.3	0.3	0.3
run right	1.5	2.9	0.7	0.8	92.8	0.4	0.6	0.3	0.3
run left	2.1	0.8	3.2	0.3	0.3	92.9	0.2	0.2	0.2
jump	0.3	0.1	0.1	0.2	0.1	0.1	97.5	1.4	0.2
idle	0.2	0.2	0.1	0.1	0.1	0.1	1.1	98.0	0.1
idle	0.5	0.4	0.4	0.2	0.1	0.1	0.1	0	98.2

Fig. 9 The class confusion matrix illustrates the allocation of the motion estimation rate for each of the motion sequences that were used

that more actions are required to be recognized by the system, the use of either additional or different features, such as the joint angle orientation, acceleration and so on, may also be quite beneficial. Finally, in cases that a different method for the activity recognition is required, methodologies such as the direct feature mapping [45], support vector machine [52], and many more can be included in the initial implementation of the presented methodology.

6.2.3 *User-Centric Evaluation*

To demonstrate the efficiency of such a character controller, the presented methodology was evaluated by real users. In the evaluation that was conducted, nine postgraduate students (thirteen males and six females, of ages 24–27) were asked to animate the character and to perform the object manipulation task that appears in Fig. 5. The users took part in the presented evaluation stand in 4 m away of a projection screen. Before starting the evaluation process each user was instructed on the tasks that he/she should accomplish. Upon receipt of the instructions 5 min were given to each user to become familiar with the character controller by simply allowing him/her to animate the character in an environment that was free of objects and obstacles. For the user-centric evaluation, the time (in s) that each user took to achieve the given goal, when interacting with the tasks for the first time, was determined. To understand the efficiency of such a character controller, the time for a user to achieve the given goal (i.e., to manipulate the objects to the goal position), was measured. This procedure was repeated a total of six times at 10-min intervals. The obtained results illustrate how the users learn to interact with such a controller as time passes. The results are illustrated in Fig. 10.

The aforementioned results indicate the following. The users required on average 152 s for their first try. This time declined dramatically to an average of 89 s after the first 10 min. Moreover, as can be seen, the time that the users took to complete the task after 20 min was 82 s. The results for the remaining attempts were very similar. By computing the range between the best and the worst try for the participants the following results obtained. When the users interacted with the given task at a first try the range was approximated at 32 s. After the fourth try the range decreased by about half reaching a value of 18 s.

Again, the results for the remaining attempts were very similar. Hence, it can be stated that the proposed methodology is quite easily learned by the users since they are able to finish the procedure in the optimal time after less than 20 min and to achieve quite close results after only 10 min. Moreover, we see that in time the users are able to learn the controller mechanism and decrease the range between the minimum and the maximum time. Based on the aforementioned results we assume that such a methodology can be beneficial in applications that require a hybrid way to animate a virtual character for enabling complex interactions in virtual environments.

7 Conclusions and Future Work

This paper has presented a full-body character control interface. The proposed methodology enables three different user-character interaction modules: the motion controller, direct manipulation of character's body, and a hybrid controller that lies between the aforementioned two controllers. Based on those three character controller mechanisms, as well as by developing a simple action controller that keeps information on the tasks that the character is able to perform, different examples were developed and presented.

The evaluation of the presented methodology has shown that both the frame rate and activity recognition process work quite well. However, when the number of possible combinations increases, the methodology's frame rate decreases. Therefore, depending on the number of possible combinations, it may not be possible to work at interactive frame rates. Additionally, even if the nine different motions are recognized satisfactorily in the proposed methodology, assuming a variety of motions that the character should be able to perform, it may be necessary to have a better solution to estimate both the motion and user's intention to perform a hybrid action. Moreover, during the hybrid character control module the synthesized motion may not look natural as there is no prior motion to ensure the naturalness of the synthesized motion. Hence, by developing either a kinematic prior or physical motion model, the system may be able to ensure the naturalness of the synthesized motion. These are the main challenges that we plan to solve in our future work in order to improve the presented methodology. Conversely, when evaluating the presented methodology by real users, we found that such a character controller mechanism can be learned quite easily. Thus, these results enforce the potential usage of such methodology in embodied interactive virtual worlds.

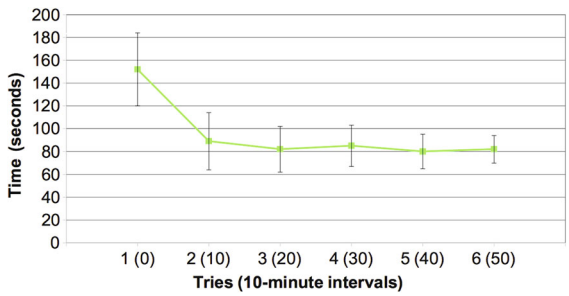

Fig. 10 Results of the evaluation of the presented methodology with real users

By developing different examples of scenarios, we have shown the possible use of such a character controller. It is assumed that there are other applications that may benefit from such a controller. Thus, in our future plans, we would first like to integrate the presented character control interface with solutions, such as ADAPT [54] and SmartBody [55] frameworks or other similar behavioral authoring frameworks, such as [56]. Hence, it will be possible for the user to take part in various events, thereby generating embodied intelligent interactions with the environments, as well as with virtual actors. Moreover, it may be interesting to implement additional character controller modules, such as a physical model related to [44, 57], making the proposed methodology quite powerful, since it will provide the user with additional choices for interaction. Finally, additional extensions that may improve the naturalness of the synthesized motion can also be implemented. Hence, solutions such as the importance-based inverse kinematics [28] or the relationship description [58] can provide improvements, especially in the presented scenario that is related to the object manipulation tasks.

References

1. Microsoft Kinect Motion Capture System, from http://www.microsoft.com/en-us/kinectforwindows/. Accessed 12 2016.

2. Assus Xtion Motion Capture Device, from http://www.asus.com/Multimedia/Xtion_PRO/. Accessed 12 2016.
3. England, D. (2011). *Whole body interaction*. London: Springer.
4. Van Welbergen, H., Van Basten, B. J., Egges, A., Ruttkay, Z. M., & Overmars, M. H. (2010). Real time animation of virtual humans: A trade-off between naturalness and control. *Computer Graphics Forum*, *29*(8), 2530–2554.
5. Multon, F., France, L., Cani-Gascuel, M. P., & Debunne, G. (1999). Computer animation of human walking: A survey. *The Journal of Visualization and Computer Animation*, *10*(1), 39–54.
6. Sarris, N., & Strintzis, M. G. (2005). *3D modeling and animation: Synthesis and analysis techniques for the human body*. Hershey: IGI Global.
7. McCann, J., & Pollard, N. (2007). Responsive characters from motion fragments. *ACM Transactions on Graphics*, *26*(3), 6.
8. Oshita, M. (2010). Generating animation from natural language texts and semantic analysis for motion search and scheduling. *The Visual Computer*, *26*(5), 339–352.
9. Mousas, C., & Anagnostopoulos, C.-N. (2015). CHASE: Character animation scripting environment. In *Virtual Reality Interaction and Physical Simulation*, pp. 55–62
10. Mousas, C., & Anagnostopoulos, C.-N. (2015). *Character animation scripting environment. Encyclopedia of computer graphics and games*. Berlin: Springer.
11. Levine, S., Theobalt, C., & Koltun, V. (2009). Real-time prosody-driven synthesis of body language. *ACM Transactions on Graphics*, *28*(5), 17.
12. Thorne, M., Burke, D., & van de Panne, M. (2007). Motion doodles: An interface for sketching character motion. In *ACM SIGGRAPH 2007 courses*, p. 24.
13. Davis, J., Agrawala, M., Chuang, E., Popović, Z., Salesin, D. (2003). A sketching interface for articulated figure animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 320–328.
14. Raunhardt, D., & Boulic, R. (2011). Immersive singularity-free full-body interactions with reduced marker set. *Computer Animation and Virtual Worlds*, *22*(5), 407–419.
15. Liu, H., Wei, X., Chai, J., Ha, I., Rhee, T. (2011) Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*, pp. 133–140.
16. Bleiweiss, A., Eshar, D., Kutliroff, G., Lerner, A., Oshrat, Y., & Yanai, Y. (2010). Enhanced interactive gaming by blending full-body tracking and gesture animation. In *ACM SIGGRAPH ASIA 2010 Sketches*, p. 34.
17. Ouzounis, C., Mousas, C., Anagnostopoulos, C.-N., & Newbury, P. (2015). Using personalized finger gestures for navigating virtual characters. In *Virtual Reality Interaction and Physical Simulation*, pp. 5–14.
18. Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, *21*(3), 473–482.
19. Mukai, T., & Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Transactions on Graphics*, *24*(3), 1062–1070.
20. Kovar, L., & Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 214–224.
21. Park, S. I., Shin, H. J., & Shin, S. Y. (2002). On-line locomotion generation based on motion blending. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 105–111.
22. van Basten, B., & Egges, A. (2012). Motion transplantation techniques: A survey. *IEEE Computer Graphics and Applications*, *32*(3), 16–23.
23. Mousas, C., Newbury, P., & Anagnostopoulos, C. N. (2013). Splicing of concurrent upper-body motion spaces with locomotion. *Procedia Computer Science*, *25*, 348–359.
24. Mousas, C., & Newbury, P. (2012). Real-time motion synthesis for multiple goal-directed tasks using motion layers. In *Virtual Reality Interaction and Physical Simulation*, pp. 79–85.
25. Witkin, A., & Popović, Z. (1995). Motion warping. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 105–108.
26. Gleicher, M. (1998). Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 33–42.
27. Feng, A. W., Xu, Y., & Shapiro, A. (2012). An example-based motion synthesis technique for locomotion and object manipulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 95–102.
28. Shin, H. J., Lee, J., Shin, S. Y., & Gleicher, M. (2001). Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, *20*(2), 67–94.
29. Sturman, D. J. (1998). Computer puppetry. *IEEE Computer Graphics and Applications*, *18*(1), 4–38.
30. Unzueta, L., Peinado, M., Boulic, R., & Suescun, A. (2008). Full-body performance animation with sequential inverse kinematics. *Graphical models*, *70*(5), 87–104.
31. Slyper, R., Hodgins, J. K. (2008). Action capture with accelerometers. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 193–199.
32. Numaguchi, N., Nakazawa, A., Shiratori, T., & Hodgins, J. K. (2011). A puppet interface for retrieval of motion capture data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 157–166.
33. Yin, K., & Pai, D. K. (2003). Footsee: An interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 329–338.
34. Misumi, H., Fujimura, W., Kosaka, T., Hattori, M., & Shirai, A. (2011). GAMIC: Exaggerated real time character animation control method for full-body gesture interaction systems. In *SIGGRAPH Posters*, p. 5.
35. Tautges, J., Zinke, A., Krüger, B., Baumann, J., Weber, A., Helten, T., et al. (2011). Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics*, *30*(3), 18.
36. Mousas, C., Newbury, P., & Anagnostopoulos, C.-N. (2014). Data-driven motion reconstruction using local regression models. In *Artificial Intelligence Applications and Innovations*, pp. 364–374.
37. Mousas, C., Newbury, P., & Anagnostopoulos, C.-N. (2014). Evaluating the covariance matrix constraints for

data-driven statistical human motion reconstruction. In *Spring Conference on Computer Graphics*, pp. 99–106.
38. Mousas, C., Newbury, P., & Anagnostopoulos, C.-N. (2014). Efficient hand-over motion reconstruction. In *International Conference on Computer Graphics, Visualization and Computer Vision*, pp. 111–120.
39. Shiratori, T., & Hodgins, J. K. (2008). Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics*, *27*(5), 123.
40. Wei, X., Zhang, P., & Chai, J. (2012). Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics*, *31*(6), 188.
41. Shiratori, T., Park, H. S., Sigal, L., Sheikh, Y., & Hodgins, J. K. (2011). Motion capture from body-mounted cameras. *ACM Transactions on Graphics*, *30*(4), 31.
42. Min, J., & Chai, J. (2012). Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics*, *31*(6), 153.
43. Chai, J., & Hodgins, J. K. (2005). Performance animation from low-dimensional control signals. *ACM Transactions on Graphics*, *24*(3), 686–696.
44. Ishigaki, S., White, T., Zordan, V. B., & Liu, C. K. (2009). Performance-based control interface for character animation. *ACM Transactions on Graphics*, *28*(3), 61.
45. Seol, Y., O'Sullivan, C., & Lee, J. (2013). Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 213–221.
46. Halmos, P. R. (1960). *Naive set theory*. Berlin: Springer.
47. Powerset Algorithm, from http://rosettacode.org/wiki/Power_set. Accessed 12 2016.
48. Samet, H. (2006). *Foundations of multidimensional and metric data structures*. Los Altos: Morgan Kaufmann.
49. Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., & Moore, R. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1297–1304.
50. CMU Graphics Lab Motion Capture Database, from http://mocap.cs.cmu.edu/. Accessed 12 2016.
51. Kallmann, M. (2008). Analytical inverse kinematics with body posture control. *Computer Animation and Virtual Worlds*, *19*(2), 79–91.
52. He, Z., & Jin, L. (2009). Activity recognition from acceleration data based on discrete consine transform and SVM. In *IEEE International Conference on Systems, Man and Cybernetics*, pp. 5041–5044.
53. Min, J., Chen, Y. L., & Chai, J. (2009). Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics*, *29*(1), 9.
54. Shoulson, A., Marshak, N., Kapadia, M., & Badler, N. I. (2013). ADAPT: The agent development and prototyping testbed. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 9–18.
55. Thiebaux, M., Marsella, S., Marshall, A. N., & Kallmann, M. (2008). Smartbody: Behavior realization for embodied conversational agents. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Vol. 1*, pp. 151–158.
56. Kapadia, M., Singh, S., Reinman, G., & Faloutsos, P. (2011). A behavior-authoring framework for multiactor simulations. *IEEE Computer Graphics and Applications*, *31*(6), 45–55.
57. Liang, X., Hoyet, L., Geng, W., & Multon, F. (2010). Responsive action generation by physically-based motion retrieval and adaptation. In *Motion in Games*, pp. 313–324.
58. Al-Asqhar, R. A., Komura, T., & Choi, M. G. (2013). Relationship descriptors for interactive motion adaptation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 45–53.
59. Dam, P., Braz, P., Raposo, A. (2013). A study of nnavigation and selection techniques in virtual environments using microsoft kinect®. In *International Conference on Virtual, Augmented and Mixed Reality*, pp. 139–148.
60. Mousas, C. (2017). Towards developing an easy-to-use scripting environment for animating virtual characters. arXiv preprint arXiv:1702.03246.