J Multimodal User Interfaces (2013) 7:269–279
DOI 10.1007/s12193-012-0113-9

ORIGINAL PAPER

A kinect-based interface to animate virtual characters

Andrea Sanna · Fabrizio Lamberti · Gianluca Paravati · Felipe Domingues Rocha

Received: 23 December 2011 / Accepted: 29 September 2012 / Published online: 20 October 2012

Abstract Most virtual characters' animations are based on armatures to manipulate the characters' body parts (rigging). Armatures behave as the characters' skeletons, and their segments are referred to as bones. Each bone of the skeleton is associated with a well defined set of vertices defining the character' mesh (skinning), thus allowing animators to control movements and deformations of the character itself. This paper presents a natural and intuitive interface, which uses the Microsoft Kinect, to interactively control an armature by tracking body poses. Animators can animate virtual characters in real-time by their own body poses, thus obtaining realistic and smooth animations. Moreover, the proposed interface allows animators to save time with respect to the traditional animation technique based on keyframing. Different examples are used to compare the Kinect-based interface with the keyframing approach, thus obtaining both an objective and a subjective assessment.

Keywords Natural interfaces · Virtual character animation · Kinect-based interfaces

A. Sanna (✉) · F. Lamberti · G. Paravati · F. D. Rocha
Dip. di Automatica e Informatica, Politecnico di Torino, Torino, Italy
e-mail: andrea.sanna@polito.it

F. Lamberti
e-mail: fabrizio.lamberti@polito.it

G. Paravati
e-mail: gianluca.paravati@polito.it

F. D. Rocha
e-mail: felipe.dominguesrocha@studenti.polito.it

1 Introduction

A lot of techniques are used in Computer Animation (CA) to animate and deform virtual characters [21]. Most animation systems are based on a "simplified" representation of the characters' skeleton. The position of each segment (usually named bone) of the skeletal model is defined by animation variables (Avars). The skeleton is used to animate characters but it does not appear in the rendered frames; on the other hand, the skeletal model is used to compute exact positions and orientations of meshes selected to represent characters, which are rendered into frames. In this way, animators can make a character move frame-by-frame by changing values of Avars related to the skeleton.

Two basic approaches are used to generate the Avars values: keyframing [6,7] and motion capture [17]. The keyframing is a process where the animator sets values of Avars only for well defined frames (named keyframes), then all intermediate frames (named in-between frames) are automatically computed by interpolation techniques. In particular, a skeleton position can be set by forward (all angles between pairs of next bones are directly specified by the animator) or inverse (the animator sets only the position/angle of the last bone of the kinematics chain—the so called end effector—and all the other positions/angles are automatically computed by an IK solver) kinematics.

On the other hand, several different technologies can be used to implement motion capture systems: electromechanical, electromagnetic, optics, and acoustics. However, electromagnetic sensor and passive optical markers are by far the most commonly used technologies for capturing fully-body motions. When CA is driven by motion capture, animators act out the scene as if they were the characters to be animated. Animators' motion is recorded to a computer using video cameras and markers, and that performance is then applied to the animated characters.

Each method has its advantages and drawbacks. Basically, keyframing can produce motions that would be difficult or impossible to act out, but complex actions can be both very difficult and time consuming to reproduce. Motion capture can reproduce in a very accurate, fast and smooth way human (and animal) movements, but motion capture systems are, in general, very expensive and motion data could be not usable for different kind of characters. This last drawback can be partially cushioned [12].

This paper presents the use of depth sensors such as Microsoft Kinect [13] to provide a Natural User Interface (NUI) for animating in real-time any kind of 3D virtual character. Both professional animators and amateurs can take advantage from the proposed solution, thus obtaining preliminary animations in a faster way. The proposed solution allows users to customize their own mapping between bones of the human skeleton recognized by the Microsoft Kinect (for more details see Sect. 3) and the bones of the kinematics chain to be animated. Joint positions of the skeleton recognized by Microsoft Kinect are streamed to an application developed by the Blender Game Engine [1,3]; in this way, the animator can move in front of the Microsoft Kinect and see, in real-time, his/her movements applied to the kinematics chain (and hence to the character) to be controlled. Movements can be gathered as interpolation curves (IPOs) where each frame is stored as a key-frame. As the Microsoft Kinect is not always able to correctly identify body poses (problems can occur when parts of the body are occluded) IPOs can be later processed to fix errors in some poses and to refine the animation. Moreover, only some few key-poses captured by Microsoft Kinect might be maintained, by using the IK solver available in Blender to compute all the intermediate ones.

The paper introduces the Microsoft Kinect device in Sect. 2, briefly describing how it works and how it can be profitably used beyond its original intended purpose in order to design and implement new effective and efficient NUIs. Section 3 describes in detail the architecture of the proposed system, whereas tests and interface assessment are presented in Sect. 4. A critical discussion on the positive and negative aspects of the proposed solution and a technical analysis are discussed in Sect. 5. Future work directions are sketched in Sect. 6.

2 Background

The Microsoft Kinect was originally intended as an input device for the home entertainment as it enables the human-machine interaction without the need of any kind of controller. Microsoft Kinect is able to track user's body poses, thus allowing to implement NUIs that use gestures and spoken commands [13]. Microsoft Kinect's sensors generate real-time depth, color and audio data of the environment; the bar encapsulating sensors is motorized to follow the user if necessary. Microsoft Kinect also incorporates an accelerometer.

From the software point of view, the stack can be composed by the Microsoft Kinect drivers on which can be layered applications based on Microsoft SDK APIs [14]. In this case, the skeleton tracking is based on the algorithm presented in [25]. On the other hand, also applications based on the PrimeSense middleware NITE [18,20] allows to track user's skeleton; moreover, a further layer such as FAAST (Flexible Action and Articulated Skeleton Toolkit [10]) can recognize body poses and consequently triggering keyboard events. A custom VRPN (Virtual- Reality Peripheral Network [26]) server allows both software configurations to stream the user's skeleton data (Avars) over a network; in this way, "remote" applications are enabled to read the skeletal joints as trackers using any VRPN client. A third solution for Microsoft Kinect is provided by the OpenKinect open source project [19].

Many researchers are exploring possible applications of Microsoft Kinect that go beyond the original intended purpose of these sensors as home entertainment and video game controllers [9,11]. These novel applications range from 3D and enhanced and augmented video teleconferencing [8,15] to flying robot control [24]. Microsoft Kinect allows researchers to re-design and re-think the human-machine interaction paradigm; this opens new opportunities for developing effective and exciting NUIs. For instance, new natural interfaces to interact with: virtual avatars [22], 3D globes [4], people to be rehabilitated [23] and many others are already available.

A Kinect-based NUI enabling the user to animate skeletonized virtual characters is presented in this paper. Similar attempts to use Microsoft Kinect for animating human virtual characters in CA are referred to J. Brekelmans [5] (that streams Vars into Autodesk's MotionBuilder in realtime, or exports animations as BVH-3DSMax format files) and to the Bloop Project [2] (that uses Vars capture by the Microsoft Kinect to animate virtual characters in Blender).

The main contribution of this work is the devising of an interface for animators to associate in real-time natural human movements to both human and non-human virtual characters by using affordable and flexible hardware. The proposed interface moves in the direction of dealing also with non-anthropomorphic shapes as the ultimate goal. In this respect, it has to be considered a first step in this direction by extending the functionalities presented in [5]; in particular, it allows users to control any kind of kinematics chain, hence any kind of virtual character, in a customizable and user friendly way. Similarly to [2], the user can obtain a trade-off between the keyframing approach and traditional motion capture systems. According the theoretical framework Tycoon [16], the proposed interface provides users two interaction modalities, keyboard-mouse and Microsoft Kinect, that cooperate by *equivalence* in order to obtain the animation of a virtual character in a better and faster way. More specifically, the multimodal interaction proposed in this paper allows the animators to use either Microsoft Kinect or the keyframing approach, but not both at the same time.

3 The proposed solution

The architecture of the system is based on the client-server model (see Fig. 1). The server, which is the Kinect application, provides the service required by the client, which is a Blender application running some Python scripts. These two parties communicate through a TCP stream socket. Since both blocks are completely independent, the system works both on a single computer, where the client connects to the localhost, and on two separate machines. The system currently allows only one client at a time and only one socket connection between the client and the server. This is done to prevent slowdowns that could be caused by a socket overload. With this limitation, the system is assured to work at the maximum frame rate possible during the whole execution of the program. This frame rate is limited to 30 fps according to the characteristics of the Microsoft Kinect, which captures both the depth stream and the VGA video at this rate.

The Kinect application is basically in charge of capturing the information of actors' movements and translating that information into a suitable format for Blender. The Microsoft Kinect is controlled by a C++ application that is responsible for all the communication between the hardware and the computer. It uses the Microsoft Kinect for Windows SDK beta; the skeleton APIs provided by the SDK allow to track the location of one or two animators standing in front of the Microsoft Kinect, providing detailed position information for each one of the twenty points or body joints that compose the skeleton. The frame data is sent to the application as a structure containing a timestamp, the frame number, the plane vector of the floor seen by the Microsoft Kinect (useful when the Microsoft Kinect is not set parallel to the ground), the gravity vector, and a set of six data blocks containing the information for each animator, where only two of them are completely populated. The skeleton data, for each of the fully tracked animators, has information on the center of mass position and the position of each of the twenty joints, along with the state of each joint, that informs if the joint is being tracked or inferred (which happens when the Microsoft Kinect cannot see this point and thus tries to accurately "guess" it basing on information of previous frames and neighboring joints). An ID is also assigned to each animator, it remains constant as long as the animator stays in the Microsoft Kinect's field of view. The body joints tracked by the Skeleton APIs are split into three main sections:

- the central area, containing the head, the neck, the spine and the hip center;
- the arms, containing for each arm the shoulder, the elbow, the wrist and the hand;
- the legs, containing for each leg the hip, the knee, the ankle and the foot.

Quaternions and positions of skeleton joints are provided by FAAST in a parent space, therefore generated data is bound to a bone structure.

The Kinect application gathers the information related to joints of a user, then defines and calculates the bones in order to allow the Blender application to directly use this information to animate the corresponding skeleton (called armature in Blender). Figure 2 shows the bones defined.

When the Kinect application is executed, it immediately creates a socket connection and waits for the Blender application's response. While the connection is established the user is calibrated. During the calibration, the program takes the first 2 s of tracking time to measure the initial position of the

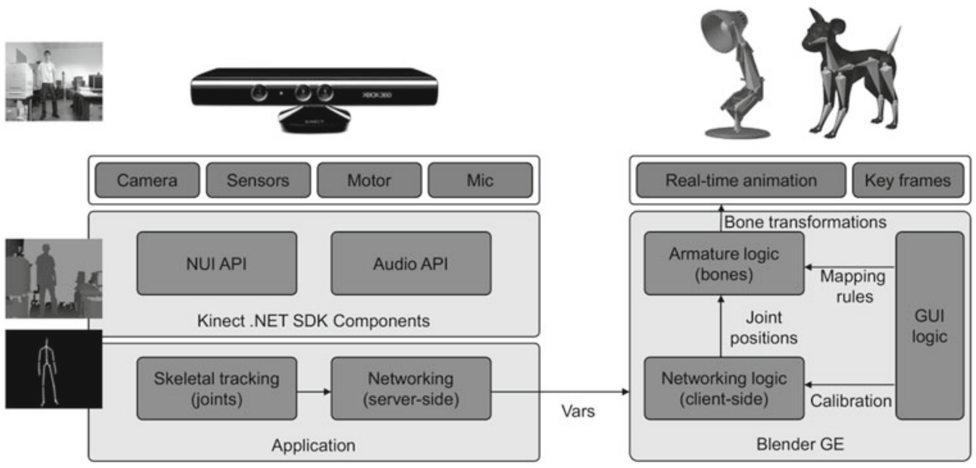


Fig. 1 Software architecture of the proposed solution. *Left pictures* show the animator, the depth map provided by the sensor and the recognized skeleton.

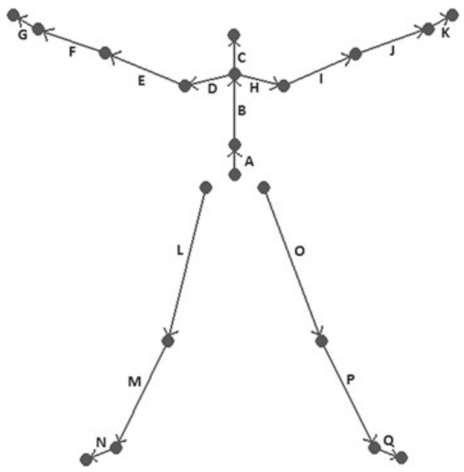


Fig. 2 Bones representing the user's skeleton as determined by the Kinect application

user and associates it with the rest position of the armature to be controlled in Blender.

The client side is a Blender application composed of Python scripts that control the execution of the program inside the Blender Game Engine (BGE). It constantly receives the user's skeleton information for each frame from the server (the Kinect application), computes the necessary transformations required and applies them to the kinematics chain to be controlled.

The Blender 3D view contains different objects needed to create the animation. The essential one for the functionality of the system is the armature to be manipulated and the model associated with that armature. The left picture of Fig. 6 shows the armature and the body mesh used for tests of human character animations. It contains all seventeen bones tracked by the Microsoft Kinect.

Figure 3 shows how the animator is captured and their movements are translated in action of a virtual character. From right to left are visible: the image of the animator captured by the Microsoft Kinect's color camera, the extracted skeleton, the depth image and the animation of the human character in the Blender's game engine.

As mentioned above, the BGE is controlled by Python scripts running in Blender. Two of them control the main services of the program. The first constantly captures the data coming from the socket connection, pushing it to a data queue; the other one, which is the primary script, is responsible for using that data to perform the necessary transformations and thus, being able to take the armature from its current position (initially the rest position) to the actual position of the user. The BGE supports armature modifications in real-time with its built-in Logic library. To act on the armature object during the execution of BGE, the armature pose channel must be modified. Changes in location can be applied only on bones unconnected to their parents (i.e., armature bone Vars are set by forward kinematics). To maintain the hierarchy logic however, this was restricted to the root bone only. All the other bones are affected with rotation exclusively.

Two non-essential scripts were also developed to improve the usability of the program during its execution. These scripts are not necessary but were included to aid the user running the application. The first script adds camera manipulation during the run-time of the program. When using the arrow keys on the keyboard, the camera is rotated around the armature whereas the user is being tracked at the same time. The other script is able to show an AVI video during the execution of Blender Game Engine. This is used to play animations already processed during the tracking of an actor, so that the animator can see the animation that needs to be performed.

3.1 Digital puppetry

Apart from the common human model seen above, the animator might be in charge to produce animations of different model types such as animals, caricatures and even rigid objects. This is not possible on a direct one-to-one matching

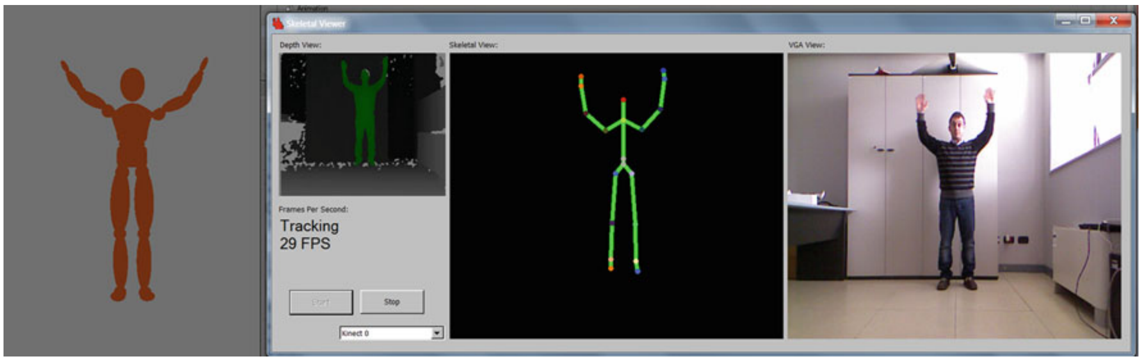
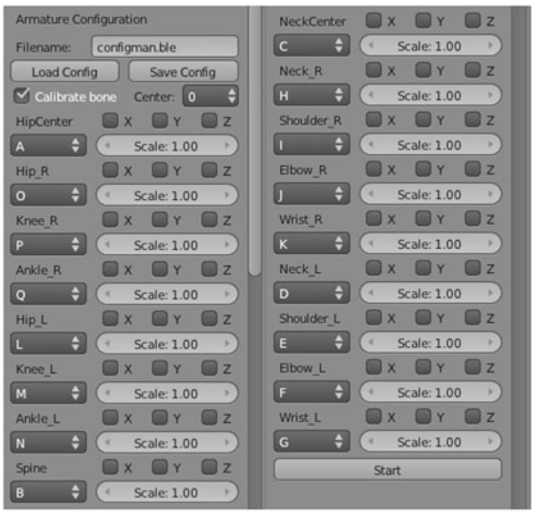


Fig. 3 *From right to left* are visible: the image of the animator captured by the Microsoft Kinect's color camera, the extracted skeleton, the depth image and the animation of a human character in the Blender's game engine



Armature Configuration
Filename: configman.ble
Load Config | Save Config
Calibrate bone ✓ | Center: 0

HipCenter	A	Scale: 1.00	NeckCenter	C	Scale: 1.00
Hip_R	O	Scale: 1.00	Neck_R	H	Scale: 1.00
Knee_R	P	Scale: 1.00	Shoulder_R	I	Scale: 1.00
Ankle_R	Q	Scale: 1.00	Elbow_R	J	Scale: 1.00
Hip_L	L	Scale: 1.00	Wrist_R	K	Scale: 1.00
Knee_L	M	Scale: 1.00	Neck_L	D	Scale: 1.00
Ankle_L	N	Scale: 1.00	Shoulder_L	E	Scale: 1.00
Spine	B	Scale: 1.00	Elbow_L	F	Scale: 1.00
			Wrist_L	G	Scale: 1.00

Fig. 4 The *configuration panel* that allows to customize the bone mapping

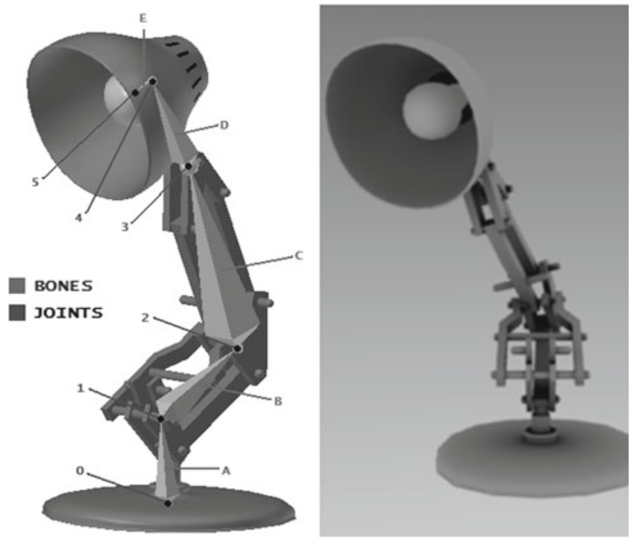


Fig. 5 *The first character* a Pixar-like lamp. The five bones of the armature are clearly visible as well as the mapping with bones tracked by the Kinect application. *right picture* shows a frame of the first lamp animation

of the human body to the model's armature shown in Fig. 6, considering that there is absolutely no correlation between them. To cover this issue some kinds of body mapping must be made, associating individually each part of the user's body to a part of the model. In order to allow the user to customize the mapping in an easy way, a configuration panel was created. This panel is located on the properties shelf in the 3D view window of Blender (Fig. 4 shows the panel). In particular, the armature configuration incorporates the options related to the armature in question and the possible mappings to be applied. The user has the choice of saving the current configured set-up to be later loaded without the need to redo it from scratch. On the other hand, the user can choose to arbitrary map each bone of the Blender armature to a bone tracked by the Kinect application. Moreover, a scale factor allows to scale a rotation to any value selected. This simply causes the angle of rotation to be multiplied by the factor, so if for instance the actor makes a movement of 50° on the XY plane and the scale factor is set to 1.5, the resulting rotation applied to that armature bone would be 75° on the same direction. Flip options, if selected, invert the rotation being performed for that particular bone on the selected axis. For example, a bone is being rotated to the left on the Z axis at the default settings with a particular motion; by setting the Z axis flip to true, that same motion would cause the bone to rotate to the right instead, with the same proportion as before. Scale factors and flip options can enable users to generate some movements otherwise impossible to reproduce.

4 Interface assessment

Several experiments were performed to test the functionality of the interface animating different test subjects/characters. Tests were aimed to evaluate the proposed solution both from a technical point of view (e.g., measuring times needed to generate animations, see Sect. 4.1) and from the user point view (subjective tests, see Sects. 4.2 and 4.3).

4.1 Objective evaluations

A set of six skilled testers was asked for performing two tests. The skilled users were Ph.D. students dealing with computer graphics and virtual reality. The first test involves the animation of a Pixar-like lamp (Fig. 5 shows the mapping with bones tracked by the Kinect application). Testers were asked for animating the lamp that basically pans left and right and later returns to the center position. For this animation, only three bones move out of the five composing the lamp. The movements are steady and smooth and the lamp bends about 30° to the two sides. The head rotates along with the body to simulate a motion where the lamp looks from side to side. The second test involves the animation of a human character; the armature is shown in Fig. 6 as well as the mapping with bones tracked by the Kinect application. The generated animation simulates a football player kicking a ball. The movement is much more complex than the animation of the lamp since most of the bones move along because of the reaction chain in the human body. During this, the arms move at different positions and the whole body moves along. For both animations, a video showing the front and the side movements of the character was provided to the testers.

The animators were asked to reproduce the movement of the videos by using the standard keyframing method and the proposed approach. Concerning the latter approach, the six users received a very brief training session about the usage of the graphical user interface to start and stop the recording of

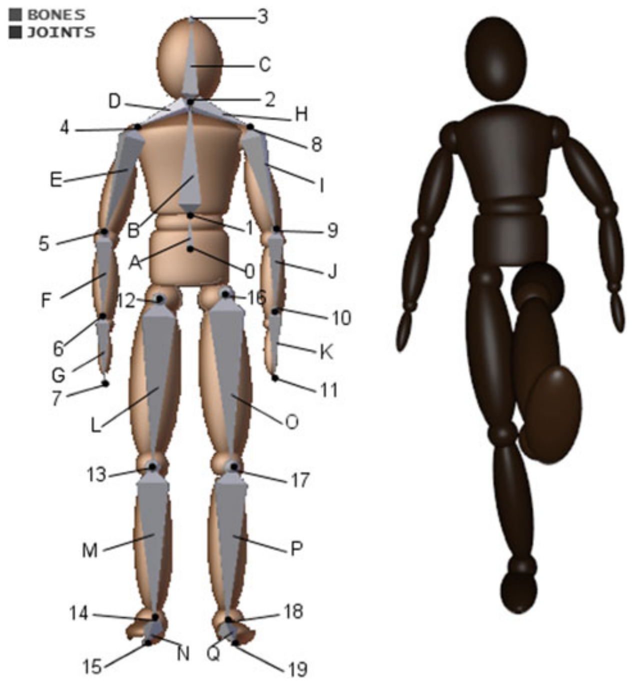


Fig. 6 The human character used for animation tests. Armature bones are clearly visible as well as their mapping to bones tracked by the Kinect application. *The right picture* shows a frame of the kicker animation

Table 1 Completion times (min) necessary to re-create the animation of the lamp and of the human character by the proposed solution and the keyframing approach

Tester	Keyframing		Microsoft Kinect	
	Lamp	Human	Kinect Lamp	Human
1	30	60	5	5
2	25	35	5	5
3	30	45	4	4
4	20	25	5	10
5	10	20	5	10
6	10	15	2	2

the human movements. For the sake of clarity, the animators were not allowed to make any modification/adjustement to the animation obtained with the Kinect-based method for these tests.

These experiments revealed that the Kinect-based approach greatly improves the productivity during the animation phase. Indeed, it allowed testers to animate a lamp and a human character by reaching time savings, on the average, respectively equal to 79 and 82 % with respect to the standard keyframing approach (see Table 1).

4.2 Subjective evaluations of animators

The same set of users involved in the experiments described in Sect. 4.1, was also asked for subjectively evaluating the proposed solution with respect to the keyframing approach. In particular, they had to rate the two solutions; a range from 1 (minimum) to 5 (maximum) was chosen. Although the number of testers involved is low, their subjective impressions are fundamental to understand current limits of the solution and to sketch future work directions.

Table 2 Subjective evaluation of the two approaches for animating virtual characters [1:poor–5:excellent]

Tester	Keyframing		Microsoft Kinect	
	Lamp	Human	Kinect Lamp	Human
1	4	3	2	3
2	3	3	3	4
3	4	2	2	4
4	4	3	3	4
5	3	4	4	5
6	4	4	3	3

The subjective assessment of the two methods (see Table 2) was positive for the human animation; in particular the proposed solution outperforms the keyframing approach as the average score is 3.83 versus 3.17. On the other hand, subjective evaluations revealed that approval rating for animating the lamp character is better for the keyframing approach (3.67 vs. 2.83). The reason given by interviewed testers is inherent to the difficulty in managing with body gestures non-human armatures without specific training. In this regard, it is worth outlining that tests were driven without giving the possibility of customizing the bone mapping for each user as a matter of time; in this way, each user had to use an already given mapping. Whereas it is straightforward to map an anthropomorphic character, it is not so immediate to map a different kinematics chain. In the case of the lamp, during the mapping phase the target armature was constrained to follow the movements of the right arm of the user, from the shoulder to the hand palm. However, this is only a possible solution. The subjective evaluation concerns only this specific choice that, in general, might be different due to a preliminar customization step. The fact that subjects preferred the keyframing approach for animating non-human shaped objects suggests that in some cases, depending on the character to be animated, a longer training phase to become familiar with the new mapping schema is necessary.

Testers were be also asked for indicating if they would change the way they work for animating a character from the keyframing approach to the Kinect-based approach. Positive answers were 66.67 %, and the main reasons were inherent to the more simple, natural and quick interface to create animations. In particular, the main advantage is represented by the speed of the envisioned motion capture system. A limited number of steps allows animators to have a first draft of the character animation, which can be later edited. Indeed, 50 % of interviews indicate that users, at this stage, prefer to use the Kinect-based approach as a supplement to integrate the keyframing method. This is due to the fact that uncertainties in body joint positions can sometime prevent smooth transitions between frames, thus making the Kinect-based approach more attractive to capture some specific poses to be later interpolated by means of the standard keyframing approach. Users prefer to use the Kinect-based approach as a supplement rather than only keyframing approach because, even in this case, the overall procedure of creating character animations would be speeded up. Finally, negative answers were rightly justified by experts because the proposed method is not able to take into account more complex animations where arms and legs occlusions take place. Moreover, some interviewed users expressed complaints regarding the calibration phase not able to correctly define the right proportion between the observed armature (real pose) and the reconstructed one, sometimes resulting in odd joint positions; in this case, the animation does not faithfully reproduce the wanted effect and a post-capture tuning phase is necessary.

4.3 Subjective evaluations of produced animations

Another kind of subjective test involved a larger set of not skilled users. In particular, two animations have been produced using both the proposed solution and the keyframing approach; moreover, when Avars were captured by the Microsoft Kinect, the color camera also recorded the movie of the animator. In this way, for each action/animation, three videos were available: a video representing the animator, a video representing the animation of a virtual character animated by the proposed solution, and a video representing the animation of a virtual character animated by the keyframing approach. Figures 7 and 8 show for both animations some frames of the reference animation captured by the Microsoft Kinect's color camera and the corresponding frames for the animations obtained by the proposed NUI and the keyframing approach.

A group of 66 students (27 males and 39 females) of the Virtual Set course, Master of Science Degree in EcoDesign at the Politecnico di Torino was asked for evaluating the two solutions with respect the "original" video. For each action (wave and kick) the three videos were projected together, in loop, for 5 min. Students were simply informed that the two virtual animations have been generated by different techniques without any other explanations. They had to choose the animation that, for their opinion, was more similar to the "real" action performed by the animator. Moreover, they had to provide a motivation. Table 3 shows results, collected by an anonymous questionnaire, where answers have been grouped in categories. For the first animation (wave), more than 80 % of students chose the proposed solution claiming that the animation seemed

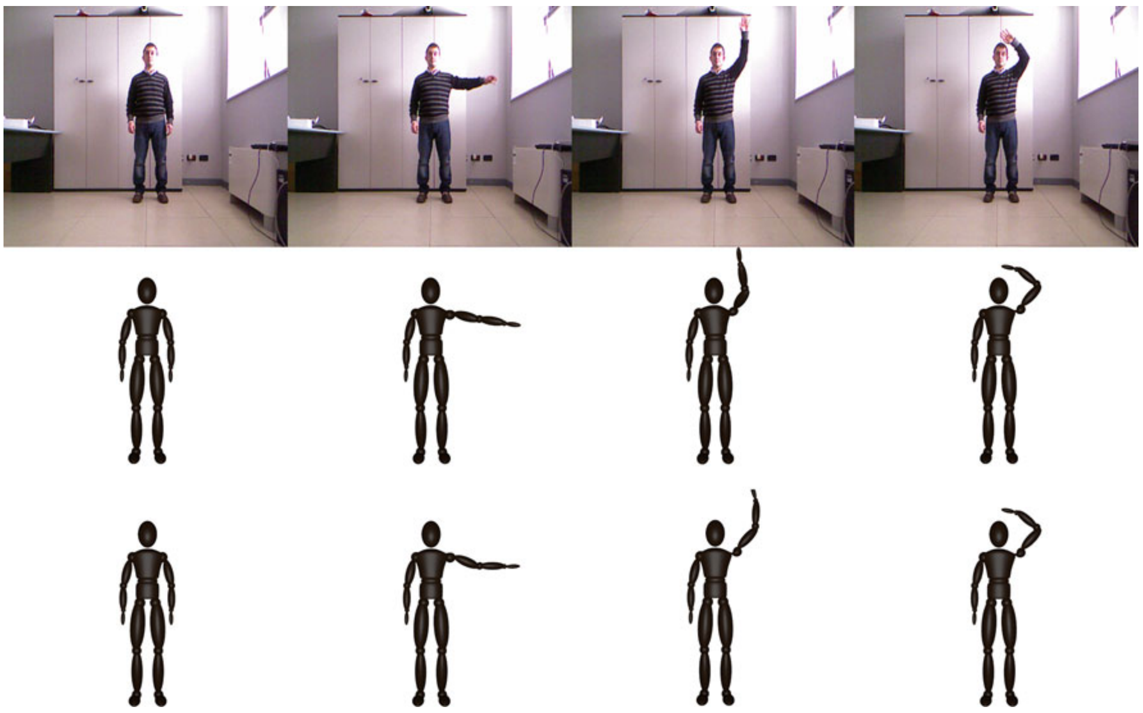


Fig. 7 Wave: *the first row* show four snapshots of the reference video, *the second row* shows the corresponding frames of the Kinect-based animation and *the third row* the corresponding frames of the keyframing-based animation

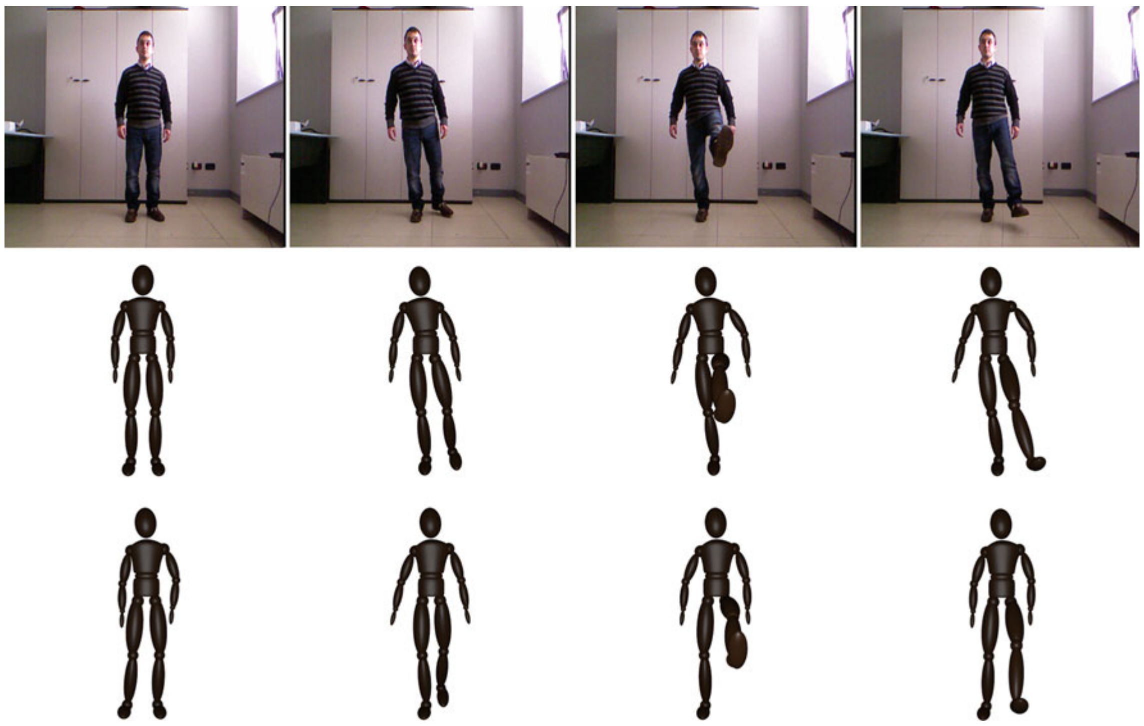


Fig. 8 Kick: *the first row* show four snapshots of the reference video, *the second row* shows the corresponding frames of the Kinect-based animation and *the third row* the corresponding frames of the keyframing-based animation

Table 3 Subjective results: the table shows the number of preferences and corresponding motivations

	Motivation
Wave by Microsoft Kinect	
27	The body's movements seems to be more natural involving in the motion more parts
18	The animation of the virtual character is more similar (therefore more realistic) to the animator's movements
5	More body's parts are involved in the motion even if the arm action seems to be less fluid
3	The motion of the arm/elbow/hand seems to be more natural
Wave by Keyframing	
10	The motion of wrist/hand/arm seems to be more natural (i.e., more realistic)
2	The animation of the virtual character is more similar to the animator's movements
1	Positions of shoulder and elbow seem to change in a more continuous way
Kick by Microsoft Kinect	
26	The body's movements seems to be more natural involving in the motion more parts (e.g., movements of chest and knee are more realistic)
19	The motion is more natural, whereas in the other video (i.e., kick by keyframing) the body seems to be imbalanced
16	The animation of the virtual character is more similar (therefore more realistic) to the animator's movements
Kick by Keyframing	
2	The motion seems to be more natural and fluid.
2	The animation of the virtual character is more similar (therefore more realistic) to the animator's movements
1	The motion of the knee is more realistic

to be more natural/realistic (45 preferences). Moreover, 27 motivations were specifically related to the involvement of all the body's parts in the animation of the virtual character. Results obtained for the kick are also more decisive: more than 92 % of students chose the animation generated by the Kinect-based NUI. It is very interesting observing as nineteen students selected the proposed solution as the motion of the character reconstructed by the keyframing approach seemed to be imbalanced, thus not natural/realistic. These results support the possibility to use Kinect-based

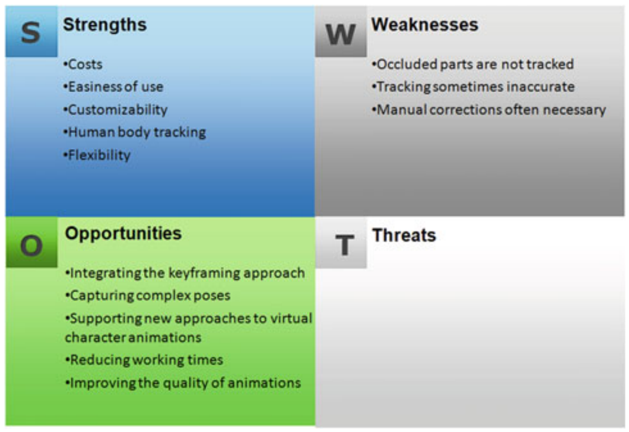

S
Strengths
•Costs
•Easiness of use
•Customizability
•Human body tracking
•Flexibility
W
Weaknesses
•Occluded parts are not tracked
•Tracking sometimes inaccurate
•Manual corrections often necessary
O
Opportunities
•Integrating the keyframing approach
•Capturing complex poses
•Supporting new approaches to virtual character animations
•Reducing working times
•Improving the quality of animations
T
Threats

Fig. 9 A SWOT analysis of the proposed solution

NUIs as affordable motion capture tools able to manage subtle movement details that make a virtual character looks real.

5 Remarks

This section presents a critical discussion on the positive and negative aspects of the proposed solution. In particular, a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis has been conducted in Sect. 5.1). Moreover, a technical analysis has been performed in Sect. 5.2, where the robustness of the system under varying conditions is discussed.

5.1 SWOT analysis

As shown by the SWOT analysis in Fig. 9, one of the main strenghts of the proposed solution concerns the affordability of the presented natural user interface due to the use of off-the shelf components and open source software as its cost is notably lower than traditional motion capture systems. On one side, an important opportunity brought by the system is that it deals with capturing complex poses by affordable hardware. On the other hand, this comes at the cost of lower robustness that leads sometimes to an inaccurate tracking phase; the use of a single depth sensor as envisioned in the current solution is an obstacle to the precision of the system, indeed occluded parts of the body are not tracked at all due to an indetermination in the position of specific limbs. Nevertheless, this fact does not affect the bones of the armature that are correctly tracked. The inaccuracy of the current system makes necessary the animator intervention to manually modify the recorded tracking data in order to be successfully used for animating a character. However, as pointed out by the subjective tests in Sects. 4.2 and 4.3, the proposed interface still remains a valid solution to speed up the process of animating virtual characters in a more natural/realistic way. In fact, it is possible to capture specific poses of the body that can be easily translated into keyframes. As such, the quality of the produced animations increases. In particular, this solution has to be considered as an integration of the keyframing approach, where manual intervention is needed only when there is indetermination in the position of joints. In addition to the previous *strengths*, the proposed solution is easy to use because it does not require any particular sensor to be worn by the animator. Lastly, its flexibility is due to the possibility of animating anthropomorphic and non-anthropomorphic bodies by customizing the bone mapping between human and virtual characters.

5.2 Technical aspects

As emphasized in Sect. 5.1, currently one of the main limits of the proposed solution concerns the robustness of the system when occlusions of body parts occur. These issues arise from the fact that a single depth sensor, as envisioned in the current system architecture, is not able to provide enough information about the position of specific limbs/joints when they are hidden. Although it is possible to make assumptions on the position and attitude of single bones basing on their direction and speed of movement, in general this critical aspect leads to an indetermination of these parameters. It is worth to note that lateral framings of a body necessarily introduce occlusions, thus limiting the usability of the motion capture system. Although some types of movements can be correctly captured by carefully positioning the sensor with respect to the actor that performs the movement (e.g., a football player kicking a ball can be captured by placing the sensor in front of him rather than laterally), other motion types are particularly challenging to catch (e.g., a dancer spinning around can not be properly captured). At this stage, these problems are tackled and overcame manually: the animator inspects, and eventually corrects, frame-by-frame the animations generated by the Kinect-based motion capture system. The time needed to carry out such corrections directly depends on the type of recorded motion and therefore the overall number of inaccurate frames.

Whereas the time needed to record a good movement in front of the Microsoft Kinect is negligibile, the time previously spent during the mapping phase is highly changing. Basically, it depends on the complexity of the target armature and its difformity with respect to an anthropomorphic kinematics chain. It is easy to imagine that the mapping process is almost immediate when dealing with anthropomorphic virtual characters; more degrees of freedom are available when dealing with simple kinematic chains (e.g., a Pixar-like lamp), but basically it is still quite straightforward to map the human skeleton to the target kinematic chain. Moreover, in these first two cases the mapping that will be chosen by different users will be essentially the same. On the other hand, in the case of complex non-anthropomorphic kinematic chains (e.g., animals, animated vehicles and so on) this process translates into a huge setup phase, which could last hours for each animator due to a large number of trials that are needed to decide the most suitable mapping schema.

In summary, the robustness of the system is mostly influenced by occlusions and issues related to tracking. In general, it is not particularly influenced by the speed of movement since tests revealed a sampling rate near 30 Hz (cf. the screenshot of the GUI in Fig. 3); this value can be considered satisfactory for most of human motions to be reproduced, since it is similar to standard frame rates in the TV and movie-making business, unless the animator does not want to carry out high speed data acquisition to simulate slow motion. In that case, an insufficient capture frame rate can be experienced. Depending on the motion type, using the system at this frequency can be problematic only for very fast movements. For example, for periodic movements such as moving the hand very quickly when waving, the perfect reconstruction of the motion is possible only when the frequency of waving is less than half the sampling frequency according to the Nyquist-Shannon sampling theorem: in the current set up, this means waving with a frequency higher than 15 Hz. For non-periodic movements, problems arise when the full range of a movement takes place in a time comparable to an adequate multiple of the sampling period. For example, consider a very fast movement of the foot when kicking. If 7 poses (correponding to as many keyframes) are supposed to be sufficient to reconstruct the entire gesture by the animator, kicks performed in a time quicker than 233 ms cannot be properly reproduced.

The precision of the system heavily relies on the initial calibration procedure. Indeed, currently the implementation of the set up phase plans to measure the initial position of the user and then associates the tracked human skeleton with the rest position of the armature to be controlled in Blender. Therefore, an initial error between tracked joints and rest position is propagated along the whole tracking phase.

6 Conclusion and future work

This paper presents a Kinect-based NUI to animate, in real-time, virtual characters. The armature of a character is animated using body movements tracked by the Microsoft Kinect; moreover, the user can choose an arbitrary bone mapping between bones tracked by the Microsoft Kinect and bones of the armature to be animated, thus enabling the control of almost any kind of character. Provided examples show the effectiveness of the proposed interface that enables users to intuitively and efficiently generate complex animations, thus saving a lot of time and possibly improving the quality with respect to a traditional keyframing approach.

This work is only the first step of a more articulated, challenging, and intriguing project. Future work will be aimed at improving the accuracy of the system. In particular, an interface based on two, or more, Microsoft Kinects that work concurrently could minimize, or at least reduce, issues related to the motion capture of occluded body parts. Currently the system works fine with human skeletons and simple kinematics chains. Non-anthropomorphic bodies are supported and they can be animated thanks to a customization phase of the mapping between the bones tracked by the Kinect application and the target armature bones. However, the mapping between bones tracked by the Kinect application and the armature animated in Blender is a manual procedure completely in charge to the user. Although this step allows animators to customize the control over the armature to be animated according to their own preferences, it can be time consuming to find a suitable mapping. In regard to this, it has been identified the need for a completely or partially automatized mapping phase, thus relieving user to find a suitable bone mapping. The developed methodology moves in the direction of efficiently dealing also with non-anthropomorphic shapes as the ultimate goal; in this respect, currently it has to be considered as a first attempt to animating also non-human characters with an affordable and flexible solution. Therefore, future research directions will concern also the study of an automatic or semi-automatic mapping phase.

References

1. Blender web site: http://www.blender.org
2. The Bloop project: http://dm.tzi.de/research/hci/bloop
3. Game Engine Manual: http://wiki.blender.org/index.php/Doc:2.4/Manual/Game_Engine
4. Boulos Kamel et al (2011) Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. Int J Health Geogr 10:45. doi:10.1186/1476-072X-10-45
5. Brekelmans Jasper web site: http://www.brekel.com
6. Burtnyk N, Wein M (1971) Computer generated key frame animation. J Soc Motion Pict Telev Eng 8(3):149–153
7. Burtnyk N, Wein M (1976) Interactive skeleton techniques for enhancing motion dynamics in key frame animation. Commun ACM 19(10):564–569
8. DeVincenzi A, Yao L, Ishii H, Raskar R (2011) Kinected conference: augmenting video imaging with calibrated depth and audio. In: Proceedings of the ACM (2011) Conference on Computer Supported Cooperative Work. ACM, New York
9. Editorial (2010) The Kinect revolution. The New Scientist 208(2789):5. doi:10.1016/S0262-4079(10)62966-1
10. FAAST web site: http://projects.ict.usc.edu/mxr/faast/
11. Giles J (2010) Inside the race to hack the Kinect. New Sci 208(2789):22–23. doi:10.1016/S0262-4079(10)62989-2
12. Gleicher M (1998) Retargeting motion to new characters. In: Proceedings of the ACM Siggraph'98, pp. 33–42.
13. Kinect web site: http://www.xbox.com/kinect/

14. Kinect for Windows SDK: http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/
15. Oliver Kreylos web page: http://idav.ucdavis.edu/okreylos/index.html
16. Martin JC (1998) TYCOON: theoretical framework and software tools for multimodal interfaces. In: Intelligence and multimodality in multimedia interfaces. AAAI Press, Portland
17. Menache A (2000) Understanding motion capture for computer animation and video games. Morgan Kaufmann, New York
18. NITE web site: http://www.primesense.com/?p=515
19. OpenKinect we site (libFreeNect): http://openkinect.org/
20. OpenNI web site: http://www.openni.org
21. Parent R (2007) Computer animation: algorithms and techniques, 2nd edn. Morgan Kaufmann, New York
22. Phan T (2011) Using Kinect and OpenNI to Embody an Avatar in second life: gesture & emotion transference. http://ict.usc.edu/projects/
23. Polli AM (2011) Natural User Interfaces in home rehabilitation using Microsoft Kinect. http://bth.academia.edu/AnnaMariaPolli/Papers
24. Sanna A, Lamberti F, Paravati G, Henao Ramirez EA, Manuri F (2011) A Kinect-based natural interface for quadrotor control. In: Proceedings of Intetain 2011
25. Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A (2011) Real-time human pose recognition in parts from single depth images. In: Proceedings of CVPR'2011
26. The Virtual-Reality Peripheral Network: http://www.cs.unc.edu/Research/vrpn/