POLITECNICO DI TORINO
Repository ISTITUZIONALE

Immersive virtual reality-based interfaces for character animation

Original
Immersive virtual reality-based interfaces for character animation / Cannavo', A., Demartini, C.G., Morra, L., Lamberti, F.. - In: IEEE ACCESS. - ISSN 2169-3536. - ELETTRONICO. - 7:(2019), pp. 125463-125480. [10.1109/ACCESS.2019.2939427]

Availability:
This version is available at: 11583/2749203 since: 2019-12-13T09:41:28Z

Publisher:
IEEE

Published
DOI:10.1109/ACCESS.2019.2939427

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

26 November 2026

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 6, 2019, accepted September 1, 2019, date of publication September 4, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939427

Immersive Virtual Reality-Based Interfaces for Character Animation

ALBERTO CANNAVÒ, (Student Member, IEEE), CLAUDIO DEMARTINI, (Senior Member, IEEE), LIA MORRA, (Senior Member, IEEE), AND FABRIZIO LAMBERTI, (Senior Member, IEEE)
Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Turin, Italy
Corresponding author: Fabrizio Lamberti (fabrizio.lamberti@polito.it)

This work was supported in part by the VR@Polito Initiative.

ABSTRACT Virtual Reality (VR) has increasingly attracted the attention of the computer animation community in search of more intuitive and effective alternatives to the current sophisticated user interfaces. Previous works in the literature already demonstrated the higher affordances offered by VR interaction, as well as the enhanced spatial understanding that arises thanks to the strong sense of immersion guaranteed by virtual environments. These factors have the potential to improve the animators' job, which is tremendously skill-intensive and time-consuming. The present paper explores the opportunities provided by VR-based interfaces for the generation of 3D animations via armature deformation. To the best of the authors' knowledge, for the first time a tool is presented which allows users to manage a complete pipeline supporting the above animation method, by letting them execute key tasks such as rigging, skinning and posing within a well-known animation suite using a customizable interface. Moreover, it is the first work to validate, in both objective and subjective terms, character animation performance in the above tasks and under realistic work conditions involving different user categories. In our experiments, task completion time was reduced by 26%, on average, while maintaining almost the same levels of accuracy and precision for both novice and experienced users.

INDEX TERMS 3D graphics, computer animation, virtual reality, Blender, user interface, evaluation.

I. INTRODUCTION

Today, a growing number of fields, including movie and video game production, architecture, industrial design, product advertising, and training, to name a few, make a massive use of computer-generated 3D animation [1], [2]. Despite recent huge technological improvements in the field, the creation of animations still requires a significant amount of diverse skills and is a very time consuming task, even for professional animators [3], [4]. To deal with the scenario depicted above, more and more users with different backgrounds like, e.g., digital artists, filmmakers and storytellers, recently started to consider the possibility to use Virtual Reality (VR) not only as medium for visualizing their animated contents, but also as a tool for creating them [5], [6].

One of the main advantage in using VR for producing animations is the sense of presence experienced by the users, which let them feel as a part of the virtual environment being animated [3]. As stated in [5], the sense of presence can boost animators' creativity and productivity. Another reason to consider VR for animation is the possibility to interact with 3D objects by using input and output devices (like controllers and head-mounted displays, or HMDs) that are natively 3D. These 3D interfaces are usually characterized by higher affordances compared to traditional 2D interaction method based, e.g., on mouse and keyboard [7].

Professionals are increasingly interested into the development of immersive environments, e.g., for articulating and animating virtual characters: PoseVR,[1] developed by the Walt Disney Animation Studios, is a recent example in this direction. Most commercial products and research prototypes, however, target mostly non-professional users. On the one hand, although these latter solutions are generally intuitive and easy to use, they lack many of the advanced functionalities offered by common animation tools like Autodesk Maya,[2] Blender,[3] etc. On the other hand, tools targeted to

The associate editor coordinating the review of this article and approving it for publication was Tai-Hoon Kim.

[1]PoseVR:https://www.technology.disneyanimation.com/projects/PoseVR
[2]Autodesk Maya: https://www.autodesk.com/products/maya/overview
[3]Blender: https://www.blender.org/

 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

professional users often miss support for a seamless integration with common animation suites [3], [4], [8]. In particular, in order to edit or reuse the animations generated with such tools, animators are asked to continuously perform import/export operations. These additional operations slow down the animation process and are highly distracting to the animators.

To complete the picture, it is worth noticing that, to the best of authors' knowledge, the literature presents only a few works reporting experiments aimed to validate the use of VR for creating animations, often considering just some of the possible animators' tasks [4], [6], [9]. Hence user studies aimed to investigate the effectiveness of VR over complete animation pipelines are becoming of particular interest, as they would be of great help in designing next-generation animation tools.

By moving from the above considerations, the goal of this paper is to investigate to what extent the use of VR can improve the various stages of a complete animation pipeline. The paper builds upon the outcomes of a previous work by the same authors [9]. In that work, a VR-based add-on for the well-know open source Blender animation suite was presented. The add-on, named VR Blender, allows users to immerse in a virtual environment and execute a set of animation tasks covered by the ACM's Computer Science Body of Knowledge for Computer Animation [10] and supported by most of today's animation suites (namely, the creation/editing of keyframing and performance animation, spline manipulation for animating objects along paths and character posing). Although it was proved that most of the tasks would actually benefit from the use of VR compared to traditional interfaces, in the experiments users were not allowed to produce a complete animation. In fact, since the goal was not to implement all the steps of an animation pipeline but the most common ones, at some point users had to remove the headset and do part of the work using mouse and keyboard as required functionalities were not available in VR.

The present work, in turn, specifically focuses on character animation, a process that typically encompasses three stages, known as *rigging*, *skinning* and *posing*. In the first stage, the character's mesh is endowed with a so-called ''rig'' or ''armature'', a kind of skeleton which will then be used by the animator to articulate the geometry, giving it the intended shape at a given frame. In the second stage, the animator sets the influence of each skeleton's bone on the character's ''skin'', i.e., on vertices of the 3D geometry [11]. Finally, in the last step the animator applies rotations and translations to bones in order to pose the character as needed.

According to [12], these three stages involve tedious tasks, which ask for significant skills and require animators to devote a lot of time in order to reach the intended quality. Thus, they represent good candidates for experimenting with VR. Moreover, they allow the animator to produce a complete animation: hence, they are perfect for studying the impact of said technology on a complete pipeline.

To investigate the execution of these operations in VR, new functionalities had to be developed and included in the add-on. However, while extending the VR-based interface, it was realized that the growing richness of available interaction patterns could increase the user's mental load and learning cost, possibly leading to a reduction in the effectiveness of immersive animation. For the same reason, all the features available in the native interface of the selected animation suite could never be brought to VR. Hence, for some operations, the user might still need to leave the VR environment.

In order to deal with these issues, two directions were explored in this paper.

On the one side, effectiveness and usability of the new VR-based functionalities for the specific pipeline tackled in this work was evaluated using both subjective and objective measurements, through a user study which involved both non-professional and professional users (animators). Various dimensions were explored, including animation time, accuracy, precision, interaction naturalness, etc.

On the other side, a new capability was developed for the add-on in order to let the users extend the interface of the VR Blender tool with custom functionalities.

A trivial way to achieve this latter goal could be to display the native windows of the Blender's interface (or parts of them) in the VR environment, and let the user interact with them using ray casting. Operations performed on graphics components like buttons, sliders, or combo boxes in these ''flat'' virtual windows could be then redirected onto the native interface to obtain the intended effect. Unfortunately, this approach would break the sense of presence and could present serious interaction issues due to the use of 2D interface elements in a 3D environment, to the limited resolution of the VR display, etc. [13]. Moreover, although the adoption of a familiar interface could simplify operations for users with previous experience with Blender, a direct replication of complex tools available in the considered animation suite that are notoriously hard to use for novice users would automatically translate their complexity to VR.

Considering the above limitations, the approach pursued in this work was to develop a new script that allows the users to pick native functionalities to be made available in VR, and customize the way to access them by choosing between various types of visual and controller-based interactions.

The new features developed in the present work were integrated with the original possibility for multiple users to simultaneously work on the same scene by exploiting at the same time the VR-based and the traditional interfaces (customized or not). The code of the new add-on is available for download as open source from the GitHub repository of VR Blender (folder ''New functionalities'').[4]

The rest of the paper is organized as follows. Section II reviews relevant literature pertaining techniques and interfaces for computer animation. Section III briefly introduces the VR Blender add-on and then describes the new features

[4]New VR Blender add-on: https://github.com/grainsgroup/VR-Blender

for managing the pipeline and supporting customization, by also providing implementation details. Section IV and Section V illustrate the experimental setup and report on the outcomes of the user study. Lastly, Section VI concludes the paper by also suggesting possible directions worth investigating in the future.

II. RELATED WORK

The idea of letting animators express their creativity using VR and related technologies, including Augmented Reality (AR), 3D user interfaces (3DUIs), etc., is not new.

For instance, more than 20 years ago, the authors of [14] proposed a method to extend the 2D sketch-draw animation paradigm to 3D using VR. Simple animations could be created using a wand manipulator and visualized on a desktop display with a pair of stereoscopic shutter glasses. The limitations of early works were strictly related to the technological characteristics of the hardware used, which constrained the complexity of imagery that could be created and introduced serious interaction issues due, e.g., to tracking accuracy, latency, etc. A decade later, the above issues could be effectively solved by ''simply'' upgrading to more sophisticated (and expensive) VR setups, e.g., relying on multi-wall projection-based immersive environments with multi-camera tracking systems [15].

More recently, developments in consumer electronics brought an incredible revolution to the considered markets, which made HMDs, gloves and similar accessories largely accessible, thus opening the way for a whole new set of animation tools leveraging the above technologies. By considering both commercial products as well as designs presented in research papers (in some cases accompanied by prototype implementations), two major approaches can be identified.

The first approach consists in *extending existing animation suites* with plugins which basically let the user see the viewport of the particular tool and interact with it in 3D. The second approach, which is by far more common, consists in the creation of ad hoc animation tools which can leverage novel interaction paradigms in VR.

A. EXTENDING EXISTING ANIMATION SUITES

An example of this type of systems is VR Viewport,[5] a software add-on which can be used to visualize the Blender's animation windows in VR using various types of HMDs. A similar strategy is adopted by MARUI,[6] a plugin for Autodesk Maya designed for creating low poly models, sketches and animations in VR. In MARUI, animation functionalities for manipulating bones, inserting/deleting keyframes and navigating the timeline are managed by means of 3D widgets. However, all the other Maya's features are made available via a set of controllers-activated shortcuts (which can be customized to activate frequently used functionalities) and menus resembling the native 2D interface.

[5]VR Viewport: https://github.com/dfelinto/virtual_reality_viewport
[6]MARUI: https://www.marui-plugin.com/

Because of the complexity of the underlying software (whose functionalities are mostly preserved) and the interfaces used, the above solutions are probably more accessible to professional than casual users. Furthermore, since the affordances of VR are not fully exploited, usability and performance are often negatively affected for both the user categories. Recently, a tool which partially copes with the above limitations has been presented [9]. The tool, devised as an add-on for the Blender modeling and animation suite, provides users with 3DUIs supporting the creation/editing of keyframing as well as the management of performance animation, character posing and spline manipulation for animating objects along paths. Extensive experiments carried out on separate animation tasks and encompassing both objective and subjective observations showed where VR technology can actually overcome traditional interaction methods for different user categories.

In rare cases, VR and related technologies have been used in combination with non-conventional interfaces. For instance, in [16], a reconfigurable Tangible User Interface (TUI) for virtual character articulation assembled using hub-and-strut construction bricks is presented. Users can build a tangible prop that mimics as much as possible the armature of the character to be animated (in terms of degrees of freedom) and see brick-to-bone mappings and real-time character's shape deformations in VR. Although the authors demonstrated possible advantages offered by the devised method in terms of time required to carry out character's posing for different user categories, overall effectiveness was limited by the fact that focus was on a single animation task (which, according to the users, was the one that could actually benefit from TUIs' affordances).

B. AD HOC VR AND AR-BASED ANIMATION TOOLS

Creating ad hoc animation tools can leverage interaction paradigms capable to make the most out of today's immersive VR systems. The above advantage is generally offset by the impossibility to rely on functionalities natively available in existing animation suites, which need to be recreated from scratch. Hence, tools that pursue this approach often provide only a subset of common animation functionalities, generally addressing the needs/requirements of specific animation tasks/techniques via redesigned interfaces. In particular, many works address armature deformation-based animation (focusing in particular on characters), since the availability of 3D interfaces is regarded as particularly beneficial for this task.

As a matter of example, the work in [17] deals with the creation of cyclic movements for 3D rigged characters using performance animation in VR. Users equipped with a HMD and pinch gloves can select the part of the model to be animated with one hand and record the animation by simply moving that part with the other hand. Authors demonstrated that users with no computer animation skill could create realistic layered animations for models with arbitrary topologies in a very short time. However, the system provides

a very limited set of functionalities, basically consisting in the interactive recording of parts' orientation over time using forward kinematics. Thus, according to the authors, it would hardly be of interest for professional animators. Marionette VR[7] has a similar focus (and similar limitations). The tool lets animators see their hands in the virtual environment and use them to intuitively animate rigged characters in real time within a physics-enabled interactive space. Although a different approach is adopted (based on inverse kinematics), rig deformations can only be obtained by manipulating marionette's ropes or applying forces to its geometry.

In [18], character animation is addressed through the design of a multimodal interface enabling VR- and multi-touch device-based collaboration. Through the proposed solution, the movement of a performer's body wearing a HMD is transferred to a virtual character, while an animator can refine it using gestures on a touch screen. Besides motion capture, the system supports basic editing operations for adjusting position, orientation and scaling. A similar approach is adopted in Mindshow,[8] a tool that is meant to create VR stories by letting multiple users (actually, actors) animate predefined virtual characters with their body and speech. A further example is represented by Merper VR,[9] a tool that allows users to pose and animate virtual characters by importing rigged meshes and exporting produced animations in the standard FBX file format.

Special-purpose solutions focusing on specialized tasks are also found in the literature. For instance, in [8], the authors present a VR application for manipulating the control points of a transition path in 3D keyframe-based animations. The proposed method addresses a very specific animation task and combines the intuitiveness of direct hand manipulation for the rough positioning of objects in space with the interaction scheme based on virtual handles for the fine adjustment of animation parameters. A comparable approach was adopted in [19] to support the creation and editing of displacement animations.

Other works have explored the execution of character animation tasks in AR. For instance, in [4] and [20], the authors show how to control the animation of virtual characters using a cube-shaped TUI. Animations to be activated are selected from a predefined set by framing AR markers on cube's faces with a mobile device's camera, whereas character's position and orientation are managed interactively by grabbing the cube itself. Results of a user study confirmed that the system can make the interaction with 3D characters easier than with traditional interfaces, especially for casual users. However, the poor flexibility due to the limited set of characters and animations that can be selected/activated with the devised interface could represent a severe limitation for general-purpose and/or professional animation scenarios. Mobile AR-based animation is investigated also in [21]. In this case, the keyframing technique is used. Transformations to be applied to virtual characters are chosen using an on-screen interface, whereas a selected set of manipulations is activated/controlled by framing user's fingers with the device's camera.

Recently, few ad hoc tools offering a richer set of functionalities have been presented. For instance, AnimationVR is a tool designed to create stories with animated objects and characters in VR [3], similarly to [9]. Characters and objects can be animated using Inverse Kinematics (IK) and performance animation. The tool supports the creation of layered animations and lets the user exploit a virtual camera to create effective shoots. No import/export functionalities are provided (the tool is developed to be used with Unity's assets). Authors performed a qualitative assessment with professional animators, showing that the tool can reduce time required for generating animations, but results could be less accurate than with common suites.

Tvori[10] and AnimVR[11] offer similar functionalities. In particular, with Tvori users can import raw characters from a library and personalize them. They can also bring to the scene other simple (unarticulated) 3D objects and sounds available in the library (by feeding it with imported assets, if needed). Characters and objects can be animated using inverse kinematics and performance animation. With respect to [3], in Tvori keyframe and path animation are also available. AnimVR focuses on hand-drawn animations, and lets users sketch scenes, objects and characters and to intuitively animate them (or part of them) by applying deformations and transformations. Another noteworthy example is Masterpiece Motion VR.[12] Differently from other works targeted to character animation, this standalone tool allows users to create the rig and manage the skinning process in an immersive virtual environment. Unfortunately, the tool is not integrated in common animation suites; hence, poses created by deforming the skeleton need to be exported to other suites if further animation steps are required.

C. CONSIDERATIONS

The review above shows that, on the one side, VR animation tools integrated with common suites used by professional animators are very rare and, in most of the cases, only partially exploit the availability of 3D input/output interfaces. On the other side, ad hoc tools generally offer an extremely heterogeneous set of functionalities, with each tool serving only parts of the needs of a specific production. Most importantly, even when a richer set of functionalities is available, animators are still requested to import and export contents to/from other suites (e.g., for modeling, rendering, etc.) with frequent context switches that limit the value of VR in the overall animation pipeline [22]. Finally, in both cases, depending on the tool considered, usability may not be adequate for casual

[7]Marionette VR: https://github.com/pushmatrix/marionettevr
[8]Mindshow: https://mindshow.com/
[9]Merper VR: https://store.steampowered.com/app/725510/Merper_VR/
[10]Tvori: http://tvori.co/
[11]AnimVR: http://nvrmind.io/
[12]Masterpiece Motion VR: https://www.masterpiecevr.com/motion

users because of the high learning cost and/or flexibility may be too limited for professional users; furthermore, quantitative evaluations concerning the various factors that could influence performance and user experience are available only in a few cases (and not on complete pipelines), making it hard to choose the tools to work with (and determine the actual impact of context switches).

III. SYSTEM OVERVIEW

This section introduces the architecture of the VR Blender add-on, its components and the new features introduced in the present work.

A. VR BLENDER

As said, the system exploited in this paper builds upon an existing software package named VR Blender [9], which was designed to let users operate some of most common tools of a well-known animation suite into an immersive virtual environment. Development and testing were carried out using a HTC Vive VR kit, though the system could in principle be used on any VR kit supported by the VR Viewport library.

The user can activate the VR interaction directly from Blender by pressing a button, grabbing the controllers and wearing the headset. No further operations, like importing/exporting models or opening external software and tools, are required to work in VR. When in VR, user is immersed in a virtual environment whose origin corresponds to the position of the Blender's main camera and is set to the center of the HTC Vive's room. Camera's orientation provides the direction for the reference system, whereas the scaling along the three axes defines the ratio between virtual displacements (measured in Blender's units) and real displacements in the tracked space (measured in meters). By default, a one-to-one mapping is configured, meaning that one Blender unit correspond to one meter in the real world; however, the mapping can be changed by the user, depending on the best view for the intended task.

The interaction with the system is managed through a state machine with four states: *Idle*, *Selection*, *Interaction* and *Navigation*. The Idle state is the condition in which the system is not receiving any specific input from the user, who, for example, is moving in the VR environment to change his or her point of view. The Selection state, which is activated with the controllers' Grip buttons, lets the user specify the element(s) of the scene (the head or the tail of an armature's bone, an object, a vertex group, etc.) to interact with. In the Interaction state, the user issues commands through the Trigger buttons with the aim to, e.g., alter the position and orientation of the selected elements, interact with the components of the graphics user interface, or use a specific system tool. While in this state, the configuration of the controllers and the available commands are dependent on the specific mode or tool activated. The Menu button of the right controller activates/deactivates the Navigation state, which allows the user to manipulate the position, orientation and scale of the Blender's main camera with the final effect of changing the VR reference system.

When producing animations, the user can choose among different modes (currently, *Performance* and *Keyframing*) and tools supporting the various animation creation and editing steps. More details about states, modes and tools included in the original version of VR Blender can be found in [9].

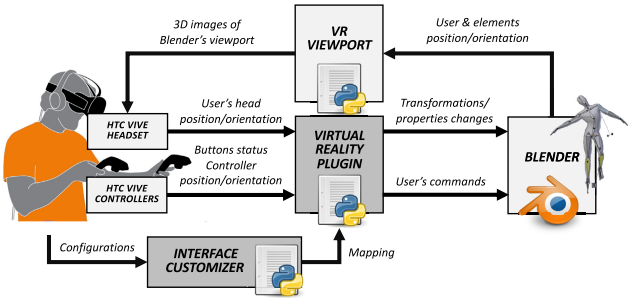


FIGURE 1. Architecture of the VR Blender system.

Fig. 1 shows an overview of the system's architecture. Changes that have been introduced in the present work to enable the new functionalities affected blocks colored dark gray. As said, new functionalities support the creation and the configuration of armature objects (which can be used, among others, to articulate virtual characters), and the customization of additional native features of the selected animation tool to be made available in VR.

In particular, to support the first functionality mentioned above three additional tools have been introduced, named *Rigging*, *Skinning* and *Constraints*. Posing was already implicitly possible with the original version of the add-on, as armatures' bones could be manipulated like any other object; hence, a *Posing* mode was not envisaged.

The Rigging tool lets the user create an armature for a given mesh in the scene, e.g., by defining its topology, assembling bone chains, subdividing bones, and defining parent relationships. The Skinning tool allows the user to adjust the bones' influence on the mesh's vertices by setting weights for them. Vertex weight is shown by using different colors according to the Blender's weight painting visualization. The red color indicates a weight equal to 1 (maximum weight), whereas vertices colored blue are assigned a weight equal to 0 (minimum weight). Finally, with the Constraints tool, the user can define the parameters of an IK chain, e.g., by setting a target bone, adjusting the chain's length, etc. These tools can be controlled through the *Settings* panel embedded in the VR interface, which is shown on the left side of Fig. 2 (new functionalities are collected under the *Armatures* label). A dedicated button (*Bone selection*), has been added to let the user choose whether to limit a given operation (e.g., the insertion of a keyframe) only to the selected bone or to apply it to the whole armature. The design of controller-based interactions will be discussed in the next sub-section.

As said, thanks to the strict integration with the selected animation suite, the above operations could be performed

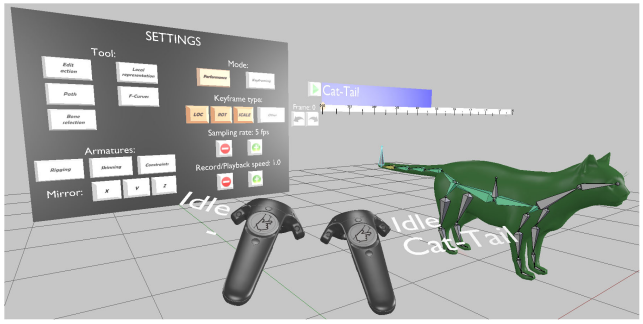


FIGURE 2. First-person view of the VR Blender immersive interface during user interaction; on the left side, widgets introduced to manage armature deformation-based animation tasks are shown. More information on other elements can be found in [9].

also in the original VR Blender tool, but the user had to remove the headset, leave the VR modality and start working with mouse and keyboard on the desktop-based native Blender's interface, thus losing the advantages brought by the immersive environment and by the use of a 3DUI. In the new version of the add-on discussed in this paper, a complete animation pipeline particularly suited to character animation can be fully managed in VR.

Furthermore, in oder to manage the second new feature introduced in the present work concerning interface customization, a new script called *Interface Customizer* was developed; the script lets the user select the native functionalities to be made available in VR, and to configure the way they can be accessed (that is, the way they are mapped to VR-based interaction modalities).

B. INTERACTION DESIGN

As illustrated in the previous sub-section, the user interacts with the system's functionalities, either standard, or customized, through graphics elements displayed in the 3D environment (like the Settings panel) and the VR controllers.

The controllers' configuration (i.e., functionalities mapped on buttons, textures and labels displayed onto them in the virtual environment) depend on the user's actions. For instance, when the Settings panel is visible, the virtual representation of the controllers is that illustrated in Fig. 3a. The texture on Trackpad buttons of both the controllers indicates that the Trigger button can be used at this time to interact with the widgets in the Settings panel. The Menu button of the left controller can be used to display/hide the panel (whose position is set to that of the controller when the button is pressed), whereas that of the right controller activates/deactivates the Navigation state.

When the Settings panel is hidden, configuration depends on the state, as well as on the mode or tool selected. The configuration of the controllers when the Navigation state is activated is depicted in Fig. 3b. Concerning the right controller, the Grip button allows the user to grab the origin of the virtual environment (i.e., of the Blender's main camera) into a new position, whereas the Trigger button can be used to apply a rotation. The Trackpad buttons let the user zoom-in/-out the entire scene. When this state is deactivated, the system automatically moves to the Idle state. With the Trackpad buttons of the left controller, the user can switch between perspective and orthogonal view. The configuration of the controllers in the Selection and Interaction state depends on the specific mode or tool.

In the following sub-sections, configurations for the new functionalities that have been integrated in the VR Blender tool with the present work are discussed.

1) RIGGING TOOL

Fig. 3c illustrates the configuration of the controllers when using the Rigging tool. A press on the Grip button of the right controller makes the system enter the Selection state, where the user can choose the part of the bone, i.e., tail, head or entire bone, to be controlled. The movement of the right controller is actually transferred to the bone part selected only when the Interaction state is activated by pressing the Trigger button of the right controller. The Trackpad of the right controller is used to manage the roll of the selected bone. In particular, a press on the Trackpad Up-Left button decreases the bone's roll, whereas a press on the Up-Right button increases it. The Down-Right/Left buttons can be used to add and delete bones, respectively. The newly inserted bone is set as a child of the currently selected bone. If the current selection does not contain any bone, the new bone is set at level zero of the armature's hierarchy.

For what it concerns the left controller, the Trackpad Up buttons can be used to create the association between the armature and the mesh, by exploiting the Blender's automatic weight assignment (with the Up-Right button) or creating for each bone an empty group of vertices influenced by the bone (with the Up-Left button). The Down-Left button allows the user to specify if the selected bone is directly connected to its parent (in this case, only the orientation of the bone can be manipulated in the posing step) or not (in this case, also the location can be controlled). A press on the Down-Right button subdivides the selected bone, creating a sequence of two bones: this operation allows to speed up the creation of long chains with same-length bones.

While rigging, the user can mirror the operations performed on the armature by selecting the symmetry axes using the corresponding widgets in the Settings panel. When mirroring is activated and a bone is selected, a mirrored copy is created, if not existing. Grabbing a bone also moves the mirrored copy by applying a transformation that considers the mirror axes selected by the user. For instance, if the mirror along the X-axis is enabled and the user applies a translation of 0.5 units along this axis to a bone's head, the system translates also the mirrored bone's head by -0.5 units.

2) SKINNING TOOL

Controllers' configuration when the Skinning tool is activated is illustrated in Fig. 3d. For what it concerns the right controller, the Trigger button is used to move to the Interaction

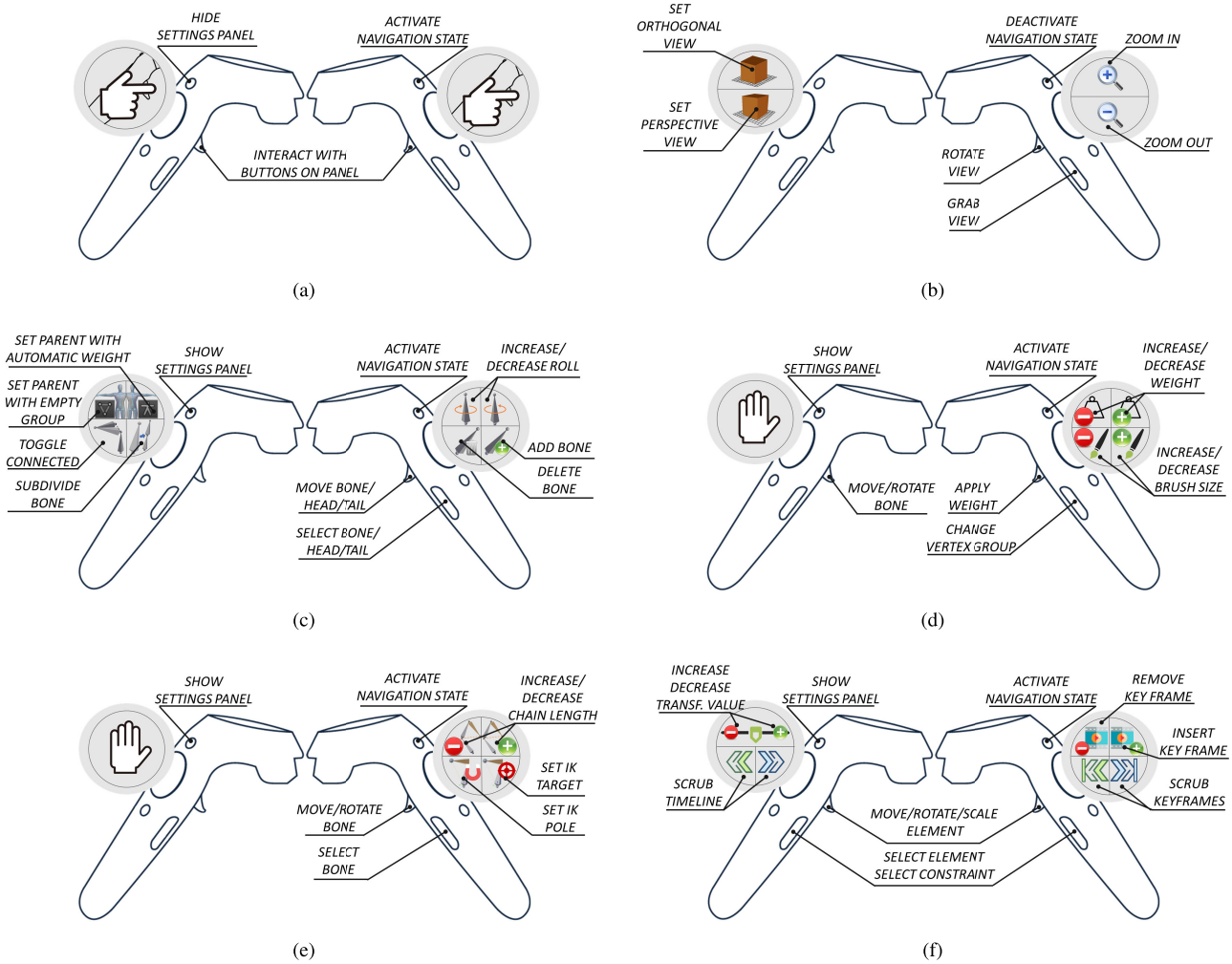


FIGURE 3. **Configurations of the left and right controllers (textures and buttons) based on system's active state, mode and tool. Focus is put on the new functionalities introduced to manage armature deformation-based animation. The full set of functionalities is detailed in [9] and in the accompanying user manual.**

state, where vertices' weight can be set. In this state, all the vertices span by a so-called 3D brush (i.e., spherical object attached to the right controller, which mimics the circular 2D selector used in the native Blender's interface) are added to the current vertex group with a weight specified by the user, as illustrated in Fig. 4. The text displayed in the VR interface to the right of the controller reports the current weight. During interaction, all the mesh's vertices are displayed as black dots (as shown in the zoom of Fig. 4). The vertex group actually modified during interaction can be chosen from the list of groups already defined for the selected object; selection is implemented by moving the right controller close to the bone and pressing the Grip button. The Trackpad Up-Left/Right buttons decrease and increase the weight to be applied to the vertices involved in the interaction. The Trackpad Down buttons can be used to increase (the Down-Right button) and decrease (the Down-Left button) the size of the brush. With the Trigger button of the left controller, the user can move/rotate the select bone to visualize in real time the effect of the current weights assigned. The Trackpad and Grip buttons have no associated functionality.

3) CONSTRAINTS TOOL

Controllers' configuration when the Constraints tool is activated is illustrated in Fig. 3e. For what it concerns the right controller, the Grip button lets the user choose a bone of the selected armature. The Trigger button is used to move and/or rotate the selected bone. The Trackpad Down buttons allow the user to define an IK chain by setting a target and pole for it. First, the user selects a bone, which will become the chain's end-effector.[13] Then, the user moves the right controller close to another bone and presses the Trackpad Down-Left/Right button to set it as a pole (that controls the direction of the chain) or as a target (that controls the position of the end-effector), respectively. With the Trackpad Up-Left/Right buttons, the user can increase/decrease the IK

[13] In IK animation, the end-effector is the element which drives the motion of the chain, making each bone assume the proper pose [23].

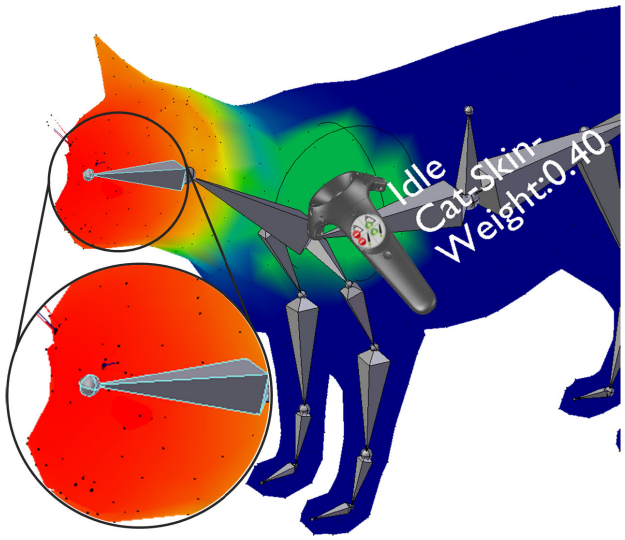


FIGURE 4. Interaction state while using the Skinning tool. 3D brush, mesh's vertices and weights are shown (using colors). Currently manipulated bone and weight being set are displayed to the right of the virtual controller. All the mesh's vertices are displayed as black dots, as shown in the zoom on the character's head.

chain's length, supposed that the selected bone has an IK constraint associated; otherwise, the length value set will be assigned to the next selected bone with that constraint. A length equal to 0 means that all the bones belonging to the chain for which a parent relationship with the selected bone has been set will be affected by the IK constraint; if length is larger than 0, only that number of bones will be influenced by the constraint. The Trackpad, Grip and Trigger buttons of the left controller have no associated functionality.

4) KEYFRAMING MODE

A complete description of all the functionalities in the original version of VR Blender not discussed in this paper is provided in a user manual which is available for download.[14] Notwithstanding, for sake of completeness, the configuration of the controllers in Keyframing mode is provided in Fig. 3f. Although this functionality was already included in the original version, it will be exploited also in the present work to perform the experimental evaluation on the new tasks.

A press on the Grip button of any of the controllers makes the system enter the Selection state, where the user can select the element(s) to be animated (one per controller). During animation, the movement of the controllers is actually transferred to animated properties (transformations or other properties) only when the Interaction state is activated by pressing the Trigger buttons (separately for the elements selected with the left and right controller).

Controllers' movement can be limited to specific axes. To this purpose, a set of movement constraints have been defined, which can be chosen from a scrollable list (activated by pressing the Grip button while in the Interaction state). If the user selects the same element with both the controllers and interacts with it by pressing the Trigger button of both the controllers at the same time (by basically executing a 3D pinch gesture), a scaling transformation is applied. The Trackpad of the right controller lets the user insert (with the Trackpad Up-Right button) or delete (with the Up-Left button) a key frame for the selected element(s). The Trackpad Down-Left/Right buttons allow the user to scrub the keyframes by setting the current frame to the next/previous key frame defined for the selected element(s). For what it concerns the left controller, the Trackpad Down-Right/Left buttons can be used to scrub the timeline by increasing/decreasing the current frame by one at each press. The Up-Left/Right buttons increase the scalar value that controls the property been animated.

[14] VR Blender user manual: http://tiny.cc/wm6l8y

C. INTERFACE CUSTOMIZATION

As said, the second major feature introduced in this paper is meant to let each user choose additional Blender's functionalities he or she wants to have access to in VR. This way, the set of functionalities available in the VR Blender tool can be extended, by at the same time letting the user choose the preferred way to work with them, thus reducing mental load and learning cost.

This new features relies on a script named Interface Customizer, which creates a configuration panel in the native Blender's interface, named *Custom VR Mapping*. Through this panel, the user can define a mapping between the selected functionalities and the interaction modalities that can be used to manage them in VR. Interaction is then handled, like for all the states, modes and tools available in VR Blender, by the Virtual Reality Plugin.

The script currently lets the user configure two types of Blender's elements, later referred to as *functions* and *controls*. Functions are single operations (like the playback of an animation, the addition of an object to the current scene or of a modifier/constraint to the selected object/bone, etc.) that, in the Blender's interface, can be executed by pressing a button or using a keyboard shortcut. Controls are variables (like the location/orientation/scale of an object along a given axis, the parameter of a constraint, etc.) that can be set using common components of the Blender's interface, like sliders, text boxes, check boxes, etc. (depending on variable type).

For instance, Fig. 5a shows the panel allows the user to set a *Limit Rotation* bone constraint in the Blender's native interface. This operation can be considered as a function, whereas the different parameters which can be configured for the constraint can be regarded as controls; some of the parameters accept binary values, whereas other parameters requires numerical or categorical inputs (see interaction components highlighted in red in the figure).

As said, in order to configure a VR-based access to elements like those above, the user can work with the Custom VR Mapping panel depicted in Fig. 5b. This panel is added to the Blender's Tool Shelf (the set of panels on the left of the Blender's 3D View).

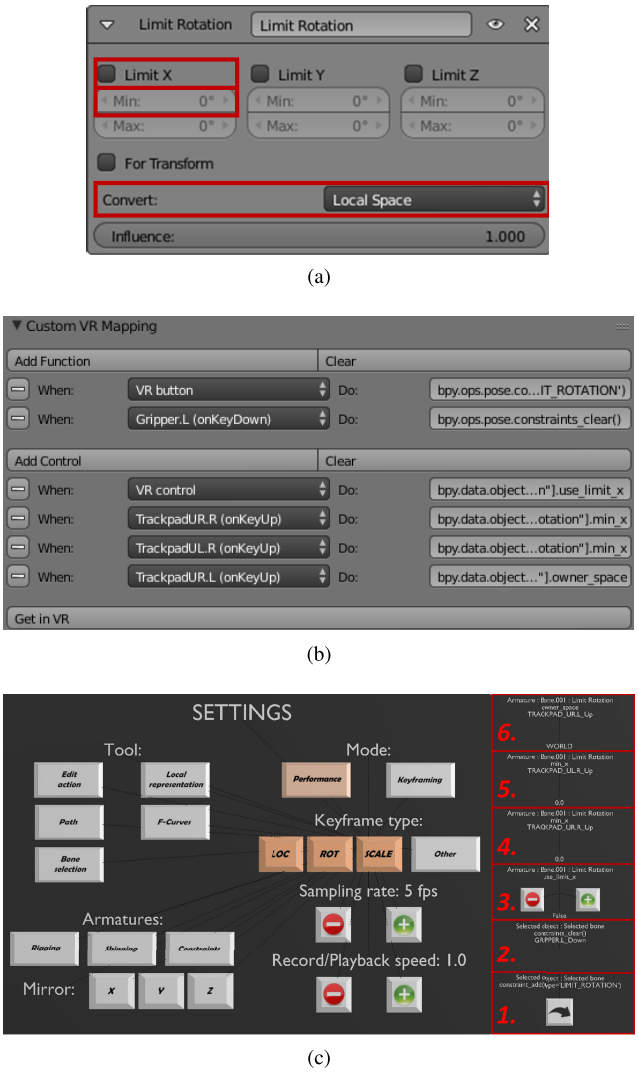


FIGURE 5. **Interface customization: a) a panel of the native Blender's interface used to configure, in the particular case, a Limit Rotation bone constraint, b) custom VR Mapping panel created by the Interface Configurator, which lets the user define custom mappings for the native Blender's functionalities associated with such constraint using a When-Do logic, and c) corresponding elements for accessing the above functionalities in VR.**

Initially, the panel presents only the *Add Function*, *Add Control*, *Clear* and the *Get in VR* buttons.

The first two buttons allow the user define a new mapping between a function/control and the way to manage it in VR, whereas a click on the Clear button deletes all the mappings created. Each mapping follows a "when this happens, do that" logic, and is defined by setting the desired values in the *When* and *Do* boxes underneath.

The When combo boxes allow the user to set the way to activate a function or modify a control in VR. For functions, the user can choose between two alternatives: a press on a *VR Button* or the recognition of a specific event (*Key down*, *Key press* or *Key up*) generated by the Grip button of one of the hand controllers.

A VR Button is a 3D button component which is added on the right side of the Settings panel displayed in the VR environment (Fig. 5c, block highlighted in red, numbered 1). To help the user remember the mapping when in VR, several labels indicating the function activated by pressing the button and the target are automatically generated (in white, above the button): as a matter of example, the labels *Selected Object: Selected bone* and *constraints_add(type='LIMIT_ROTATION')* in the considered figure indicate that a Limit Rotation bone constraint can be added to the currently selected bone of the active object (an armature) by interacting with such a button. The buttons are visible only when the Settings panel is displayed. The user can interact with them using the controller in the same way he or she interacts with the other buttons of the panel, i.e., by using the Trigger button onto them.

The alternative to the VR buttons when configuring the When combo box for functions is represented the three events associated with the Grip button of the hand controllers. The Key down event is fired once when the user presses the button; the Key pressed event is generated continuously as the button is pressed; when the user releases the button, a Key up event is fired. When the user configures a mapping using one of these events, a new block is added to the right side of the Settings panel (Fig. 5c, block 2): in this case, since there is no interactable 3D component in the interface, only the labels recalling the associated function and its target are shown. For instance, the mapping depicted in the figure allows the user to remove all the constraints (function *constraints_clear()*) for the currently selected bone in the active armature by pressing the Grip button of the left controller.

Concerning controls, the user can choose between two alternatives for defining the When part of the mapping: a 3D interaction component referred to as a *VR Control*, or one of the events generated by using the Trackpads (Up-Right/Left and Down-Right/Left buttons).

When the VR Control is selected, a new block is added in the Settings panel (Fig. 5c, block 3) with some labels (showing the name of the associated variable and its current value) and two buttons (to increase/decrease the value).

When the mapping is based on Trackpads' events, only the labels are shown (Fig. 5c, blocks 4–6).

The types that can be handled by a control are binary, integer, floating point, enumeration, and object. When the user tries to modify the value of a control by interacting with the two buttons of the VR Control or with the Trackpads, the system automatically understands the variable type and sets a new valid value for it. For instance, if the type is binary, then the systems inverts the value; if it is an enumeration, then it sets the value to the next valid item in the list of categorical alternatives; if is an integer, the value is increased or decreased, etc.

In the example shown in Fig. 5c, blocks 3–6, a VR Control is used to switch the value of the binary variable named *use_limit_x*, which corresponds to checking/unchecking the *Limit X* check box for the Limit Rotation constraint in Fig 5a associated with bone *Bone.001* in the *Armature* object; the Trackpad's Up-Right/Left buttons of the right controller

manage the value of the *min_x* variable controlling the minimum value along the above axis, whereas the Up-Right button of the left controller allows the user to select a different value for the *Convert* enumeration (which sets the coordinate system used by the constraint).

In the Do text box, the user is expected to specify the reference to the function/control to be executed/set when the condition specified in the associated When block is verified. References are expressed as Python commands, which are then automatically processed to extract the labels used in the blocks discussed above. With the aim to support different user categories (including those with no expertise with Python), various ways to specify the command are supported:

- using the online Blender's API documentation[15];
- executing the function or modifying the value to be controlled by acting on the Blender's interface, then copying & pasting the last row which appears in the Blender's Info Editor;
- right clicking with the mouse on the corresponding graphics component of the Blender's interface and choosing *Copy Python Command* from the contextual menu that appears, or manually copying the string.

The system also recognizes specific keywords which allow the user to extend the effect of a specific Python command to other objects besides the intended target. For instance, the Python command to change the floating point value of *Min* text box for the Limit Rotation constraint in Fig. 5a (determined using one of the three methods listed above) is *bpy.context.object.pose.bones['Bone.001'].constraints[Limit Rotation].min_x*. By analyzing the command, it appears that a specific bone (namely, *Bone.001*) is defined as target for the command. To extend this command to currently selected bone, the user simply has to replace the bone name above with the keyword *SELECTED_BONE*. Currently, the other supported keyword is *SELECTED_OBJECT*, to apply the specified command to currently selected object. As it can be seen, in Fig. 5c there are blocks for which a precise target is already specified, and blocks for which it is not (target will be the object or bone currently selected in VR, in these cases).

In the Custom VR Mapping panel, the button to the left of each When-Do pair allows the user to remove that mapping.

Once the mappings have been defined, the user can press the Get in VR button at the very bottom of the panel to create the blocks attached to the Settings panel or update previously created blocks.

The mappings configured through the procedure described above become automatically active when the user opens the Settings panel. When the panel is hidden, the custom configuration has no effect since the default mappings of the controllers are restored depending on the currently active state, mode or tool. User is not allowed to use default mappings in defining his or her custom configurations, in order not to interfere with predefined VR Blender functionalities.

Several videos have been recorded to illustrate the flexibility of the devised interface customization mechanism.[16]

The first video shows how to add a *Track to* constraint to make the selected object follow another object; both objects and constraint can be added using custom When-Do blocks. The second video shows how to precisely limit the rotations of a rigged character's bones by setting the proper constraints on the armature. The third video shows how to manage very heterogeneous Blender's functionalities by creating custom functions and controls for, e.g., setting objects' shading, changing the timeline start and end frames, as well as configuring a sub-surfacing modifier. It can be easily seen that methodology works both with animation- and modeling-oriented functionalities.

IV. EXPERIMENTAL SETUP

The user performance with a full animation pipeline into an integrated, immersive environment can be influenced both by the precision of interaction, as well as by the user experience associated with the functionalities offered by the VR interface. In order to thoroughly investigate the above perspectives, two user studies were designed involving students and academic staff at the authors' University. The following subsections discuss the setup adopted in each of the studies; experimental results are then presented in the Section V.

A. INTERACTION PRECISION

This section describes the first study designed to explore the precision of interaction with the VR Blender tool.

1) OVERVIEW

The factors that can negatively influence the precision of interaction with VR elements (and, hence, the way users can control 3D objects' parameters, like position, orientation, etc.) are the precision of the mechanism adopted to track the hand controllers (and the headset) and the difficulties that users can face while operating in VR because of the characteristics of the visualization and interaction technologies.

Considering the first factor, previous works like [24] and [25] experimentally investigated the precision of the HTC Vive's tracking. As reported in these works, the precision depends on the region of the tracked 3D space where interactions occur (in the center of the tracked space or on its boundaries), as well on the size of the tracked space. In the best conditions, the HTC Vive's tracking system is characterized by a sub-millimetric precision [24].

Regarding the second factor, first of all it is worth observing that, as illustrated in the previous sections, the users are allowed to dynamically change the mapping between the real and virtual reference systems by exploiting the functionalities available in the Navigation state. That is, the precision of the tracking system can have a different impact on users' operations (like reaching a specific position, setting a given orientation, etc.) depending on the actual zoom level.

[15]Blender API Documentation: https://docs.blender.org/api/2.79/

[16]Videos on interface customization: http://tiny.cc/ipvsaz

Moreover, a higher zoom allows the users to visualize the VR environment more in detail; this latter fact makes the defocusing problem for the objects that are very close to the users' point of view less relevant, letting them achieve better performance.

2) TASKS

Based on the above considerations, a study was designed to analyze the correlation between the zoom level set in the VR environment and the corresponding interaction precision.

Taking into account the basic operations involved in the use of the considered animation pipeline, two tasks were developed, referred to as *Positioning* and *Rotating* tasks.

The goal of the Positioning task was to move a controlled bone (could be any other object) as close as possible to a reference bone. The goal of the Rotating task was to apply a specific rotation to the given bone. In order to exclude factors that could influence the measurements, rotation (translation) of the controlled bone was disabled in the first (second) task.

The approach used in [24] was adapted to the context of the specific experiment: in particular, a sampling methodology was developed requesting a single user to perform each task with five different zoom levels. Overall, the user repeated the two operations 500 times for each level of zoom.

For both the tasks, manipulations had to be performed in the center of the tracked space, where precision of the tracking technology is known to be the highest possible.

3) EVALUATION CRITERIA

The Euclidean distance (for the Positioning task) and the angular distance (for the Rotating task) between the reference and the controlled bone were computed. Like in [24], the precision was then estimated by first calculating the mean and the standard deviation of the distances between the samples collected for a specific zoom level and their centroid. Moreover, like in [25], the sample-to-sample jitter was assessed by calculating the root mean square (RMS) of the differences in the measured positions and orientations as:

$$RMS_m = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}\Delta m_i^2} \tag{1}$$

where m is the considered measure (position or orientation, for a particular zoom level), n is the number of samples, and Δm_i is the difference between sample i and $i+1$. By following the methodology in [25], RMS was computed for each axis.

B. USER EXPERIENCE

This section introduces the second study, which was designed to specifically investigate the effectiveness of using the novel functionalities implemented to support rigging, skinning and posing of virtual characters in the VR Blender tool.

1) OVERVIEW

The study involved 23 volunteers (14 males and 9 females, aged between 21 and 34, $\mu = 25.61$, $\sigma = 4.03$), who were categorized in two groups based on their experience with computer animation software as well as VR and related technologies (as done in [9]). In particular, 13 volunteers could be considered as non-professional users (NPUs), since they were students attending a course on computer animation with Blender. The remaining 10 volunteers could be considered as professional users (PRUs), because of their expertise in teaching similar courses or because working in the field. PRUs also reported a much higher experience with the considered technologies, and most were accustomed to working in VR environments.

2) TASKS

Each user was asked to carry out four different animation tasks using both the native Blender's interface (in the following referenced as BNI) and the VR-based interface of the devised tool (VRI). Tasks were designed to cover the key stages of the pipeline for armature deformation-based virtual character animation, from rigging to skinning, and posing. The first three tasks requested the users to work with a cat model,[17] and were developed with the aim to "isolate" the user experience with each of the three animation stages: hence, they included 3D references which are generally not available to animators. The fourth task recreated more closely the real animation work over a complete pipeline, by asking the users to work with a human model[18] and a 2D reference.

The tasks, shown in Fig. 6, were tested in a random order, balancing the number of users who started with the BNI and the VRI, in order to limit bias due to learning effects. In the following, the four tasks are described in detail. Tasks were executed always in this order, which is the standard one for the considered animation method.

- *Rigging*: the goal of this task was to create an armature for the provided model. As shown in Fig. 6a, users were provided with a reference armature (shown with sticky bones), and had to use the Rigging tool to create new bones (displayed using octahedral shapes). User-created armature should match as much as possible the reference one.
- *Skinning*: in this task, the weight of vertices assigned to a selected set of bones had to be modified, in order to correct evident mistakes (as shown, for instance, in Fig. 6b and Fig. 6c, where the neck and the left hind leg bones are mistakenly set to influence also vertices on the tail and the right hind leg of the mesh, respectively). For this task, users had to use the Skinning tool. It is worth pointing out that it was decided to provide users with an initial set of colored weights to change, rather than making them work starting from the outcomes of the first task. This way, it was then possible to compare users'

[17] Cat model: https://free3d.com/3d-model/low-poly-cat-46138.html
[18] Human model: https://free3d.com/3d-model/body-mesh-28679.html

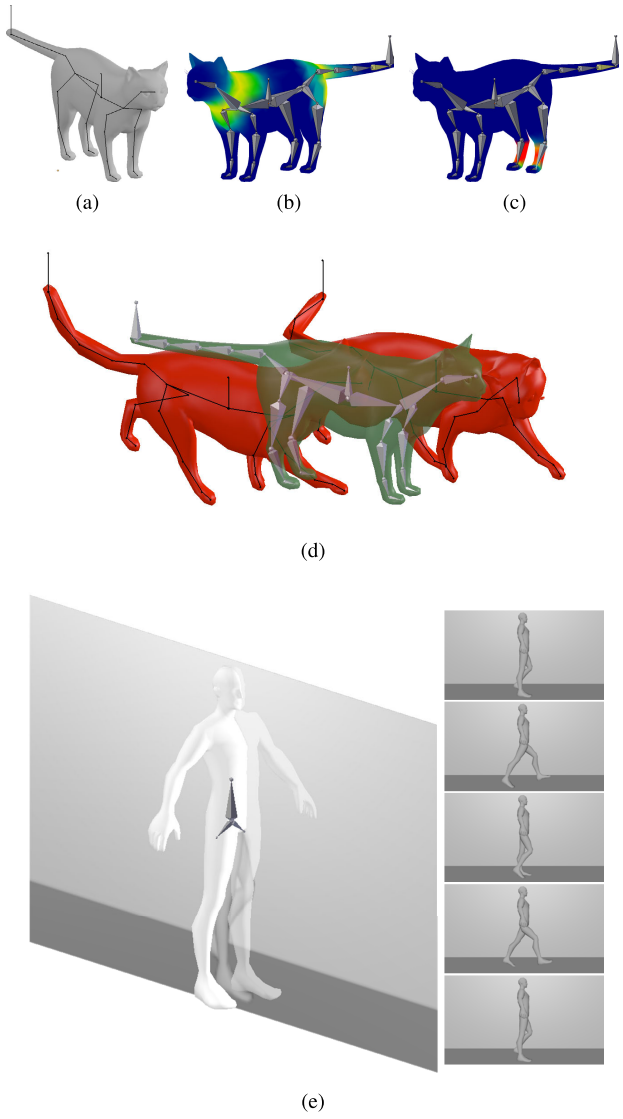


FIGURE 6. Tasks considered in the user study: a) rigging, b)–c) skinning, d) posing on a cat character, and e) complete pipeline on a human character.

performance on this specific task independent of the quality of the armatures they created. Notwithstanding, more sophisticated skinning operations that would have to be carried out in a complete animation pipeline were investigated in the fourth task.

- *Posing*: in this task, users had to articulate the armature of a rigged character including 29 bones and 66 degrees of freedom using both forward and inverse kinematics. Like in the previous task, it was chosen to work with a reference (in this case, a rigged model) in order to collect results that could be compared across users. The final goal of the task was to make the pose of the green cat in Fig. 6d match as much as possible the reference pose (the red cat in the same figure). Users had to recreate two poses, one at frame 0, the other at frame 20 (with the system configured in Keyframing mode). This way, an animation was actually produced, with intermediate frames created by Blender through interpolation. As for the second task, simplifications introduced in this tasks were meant to investigate user experience in isolation. More sophisticated animation patterns were analyzed through the fourth task.
- *Complete pipeline*: in this task, users had to traverse the whole animation animation pipeline, iterating the rigging, skinning and posing stages as needed. At first, users were requested to use the Rigging tool for creating the armature to be articulated in the remaining of the task. Once the rigging was completed, users had to create the relationships between the armature and the mesh by choosing the preferred parenting method (with automatic assignment or with empty vertex groups). Regardless of the method chosen, users were forced to use the Skinning tool in order to make sure that current assignments did not present any flaw (to be corrected, in the case of automatic assignment) or to define the weights to be assigned to vertices (in the case of empty vertex groups). Users were then requested to create an entire walking cycle animation for the provided character. At any time, users were free to return to a previous pipeline stage to fix possible errors (e.g., to add a new bone to the armature or to change the weights) in order to improve the quality of the animation. Differently than in the first three tasks, the reference animation to be recreated was presented through a 2D video clip displayed in the virtual environment: in this way, much more realistic usage conditions were recreated, since the use of a video reference is very common in the creation of animations with traditional suites (a further alternative could have been to let the users free to create any animation, but then it would have been impossible to quantify performance in objective terms). The video did not show neither the topology of the character's armature nor the exact position of the keyframes to be inserted: this information had to be inferred by changes in the video reference. Fig. 6e shows, on the left side, the character to be animated and how it appears in VR at the beginning of the task (only a basic armature with three bones was provided, which had to be modified); on the right side, same frames of the video reference are depicted. The duration of the animation (of the video reference) was set to 60 frames.

Before starting the experiments, users were introduced to the BNI and the VRI, focusing on functionalities needed for executing the tasks. In order to familiarize with the two interfaces, users were given time to perform the same operations required for the given tasks, but on different scene elements. Additional time was allocated for users without previous experience in VR environments. In order to balance the low experience in the use of the VRI with respect to the BNI, users were allowed to ask for help during the execution of the tasks with both the interfaces (in most of the cases, those who looked for help wanted to have a confirmation that they were executing operations in the proper way). Like in [9], no time limit nor minimum thresholds on quality or other

metrics were set a priori. Users were left free to decide when to consider the given task as completed. Videos showing the execution of the four tasks are available for download.[19]

3) EVALUATION CRITERIA

To evaluate users' performance, both objective and subjective measurements were collected during and after the experiments, respectively.

As for objective measurements, two metrics already exploited in previous works were considered [9], [16]. The first metric, named *completion time* (T), accounts for the time needed to complete the task and could be calculated on all the four tasks.

The second metric, named *animation accuracy* (A), measures the difference between the reference result and that of the user's. This metric is applicable only in the first three tasks, where a ''homogenous'' reference is available (in the fourth task it is not possible to collect data that could be compared across different users, since the number of bones in the created armature as well as the number and position of inserted keyframes were arbitrary). It also needs to be separately defined for each task, as the end result is different. In all the cases, it was rescaled from 0 to 100, where 100 represents a perfect match. For the Rigging task, the Euclidean distance between the bones of the reference and of the user-created armature was considered (averaged on all the bones).

For the Skinning task, the metric, based on the weights distances, was computed as follows:

$$A = 1 - \frac{1}{\Omega} \cdot \sum_{i=0}^{N} \left(\sum_{j=0}^{M} weight_i(j) - \sum_{j=0}^{M} weight_i^*(j) \right) \quad (2)$$

where N and M are the number of vertex groups and vertices, respectively, and $weight_i(j)$ represents the weight assigned to the j^{th} vertex in the i^{th} group. The * symbol is used to indicate reference weights (i.e., weights in the cat character with no mistake in the vertex groups). Ω is a normalization factor computed as follow:

$$\Omega = \sum_{i=0}^{N} \left(\sum_{j=0}^{M} weight_i^0(j) - \sum_{j=0}^{M} weight_i^*(j) \right) \quad (3)$$

where $weight_i^0(j)$ are the weights assigned to the vertices at the beginning of the task.

For the Posing task, two alternatives were considered. The first alternative is represented by the metric used in [9], [16], which considers the Euclidean (for the position) and angular (for the orientation) distances of bones averaged on frames. The second alternative is an adaptation of the metric used for the Rigging task, which is computed as:

$$A = 1 - \frac{1}{2} \cdot \sum_{i=0}^{N} \left(\frac{\delta_i}{\Delta} \right) \quad (4)$$

where N is the number of bones in the armature, δ_i is the Euclidean distance between the center of the bones in the reference and the user-controlled armatures, and Δ is a normalization factor calculated similarly to δ_i, but considering the controlled armature in the rest position (for frame 0) and in the pose of frame 0 (for frame 20). The first metric was deemed as not appropriate for the particular task considered in this work. In fact, it was designed for posing tasks not involving displacements of the whole object (only deformations). In this case, users had to move the cat character to create the animation. Hence, in Section V, only the second metric will be considered.

A third metric, named *animation precision* (P), was then defined to quantify, in numerical terms, the precision of animation results obtained by the users. This metric is different than the one defined for measuring interaction precision: in fact, the present metric also accounts for the impact of user experience with the given animation functionalities.

Like the second metric, it is applicable only when a homogenous reference is available, and has to be defined in different terms depending on the task considered. In particular, for the Rigging and Posing tasks, the metric was computed by first calculating a centroid for the coordinates of corresponding bones as set by different users; then, the distances between each bone and its centroid was calculated and averaged among the various users. Standard deviation was also provided. For the Skinning task, first the sum of the weights assigned to all the vertices in a given vertex group was measured for each group; then, a centroid was computed by averaging the sums among the users; finally, the mean and the standard deviation of the distances between the sums and the centroid were measured for each group.

Subjective measurements were collected by asking users to fill a post-test questionnaire split in three sections. The aim of the first section was to evaluate the overall usability of the two interfaces based on the System Usability Scale (SUS) [26]. The second section assessed users' satisfaction after the execution of the tasks; questions were adapted from previous work on 3D animation with non-traditional interfaces [4]. In the last section, users' preference for the two interfaces was investigated by considering each of the four task separately. Since in the last task the whole animation pipeline is traversed, scores assigned to it could be regarded as an indication of overall users' preference. The questionnaire is available for download.[20]

V. RESULTS

In the following, results obtained by applying the above criteria are presented and discussed with the aim to estimate the precision of the VR-based interaction, as well as to compare the BNI and the VRI in terms of user experience.

A. INTERACTION PRECISION

Results about interaction precision are reported in Table 1 and Table 2. As expected, an increase in the zoom level

[19] Videos of the experiments: http://tiny.cc/1hxsaz

[20] Questionnaire: http://tiny.cc/hf2saz

TABLE 1. Interaction precision: mean values and standard deviations of the Euclidean and angular distances between the samples and the corresponding centroid for the considered measures.

Zoom level	Location (*m*)		Rotation (*deg*)	
	μ	σ	μ	σ
0.25×	0.0069	0.0038	1.0133	0.5730
0.50×	0.0029	0.0013	0.6739	0.3566
1.00×	0.0017	0.0016	0.5828	0.4008
2.00×	0.0010	0.0005	0.26530	0.1720
4.00×	0.0006	0.0003	0.2257	0.1622
Pearson's corr. coeff. ρ	−0.9034	−0.8855	−0.9711	−0.9269

TABLE 2. Interaction precision: RMS values for the considered measures.

Zoom level	Location (*m*)		
	RMS(x)	*RMS(y)*	*RMS(z)*
0.25×	0.0006	0.0007	0.0007
0.50×	0.0002	0.0002	0.0003
1.00×	0.0002	0.0001	0.0002
2.00×	0.0001	0.0001	0.0001
4.00×	0.0001	0.0001	0.0001
Pearson's corr. coeff. ρ	−0.8712	−0.8716	−0.8890

Zoom level	Rotation (*deg*)		
	RMS(x)	*RMS(y)*	*RMS(z)*
0.25×	0.0552	0.1444	0.0532
0.50×	0.0302	0.0894	0.0342
1.00×	0.0220	0.0746	0.0229
2.00×	0.0105	0.0392	0.0092
4.00×	0.00732	0.0350	0.0067
Pearson's corr. coeff. ρ	−0.9524	−0.9574	−0.9736

made the distances of sampled values from the corresponding centroid get progressively lower. The correlation between the zoom level and the values of the metrics was evaluated by computing the Pearson's correlation coefficient. Before making the above computation, a visual analysis of the plotted data suggested to apply a transformation to the zoom variable in order to obtain a liner scale. To this aim, a logarithmic function was applied to remap values 0.25, 0.5, 1, 2, 4 to −0.6, −0.3, 0.0, 0.3, 0.6. The values of the correlation coefficient ρ for the various measures confirms a high inverse correlation between the zoom level and all the considered metrics.

B. USER EXPERIENCE

In this section, results concerning user experience are discussed. Objective results are considered first, followed by subjective results.

1) OBJECTIVE RESULTS

Results concerning completion time, animation accuracy and animation precision for the two user categories are reported in Fig. 7, Fig. 8, and Fig. 9, respectively. Statistically significant results (verified using paired t-tests, $p = 0.05$) are marked with *.

Considering first the results collected for NPUs, it can be observed that the VRI exceeded the BNI in terms of completion time for all tasks. In particular, speedup was 33%

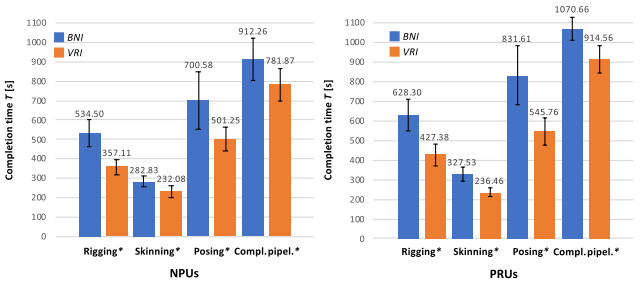


FIGURE 7. Objective results in terms of completion time for a) NPUs and b) PRUs. Mean values (bar height) and standard deviations (error bars) are shown.

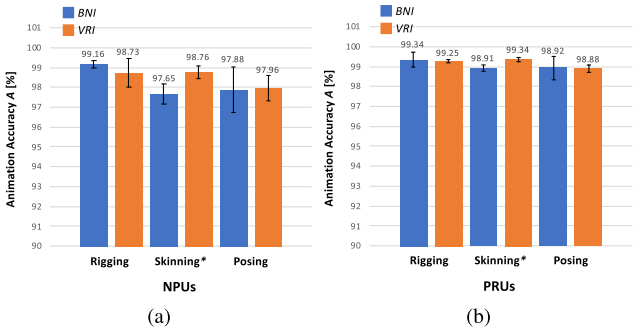


FIGURE 8. Objective results in terms of animation accuracy for a) NPUs and b) PRUs. Mean values (bar height) and standard deviations (error bars) are shown.

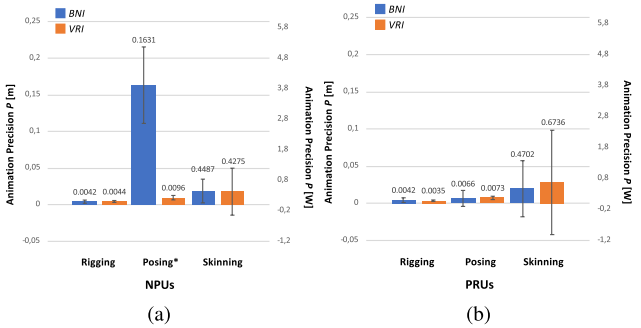


FIGURE 9. Objective results in terms of animation precision for a) NPUs and b) PRUs. Mean values (bar height) and standard deviations (error bars) are shown.

in the Rigging task ($p = 0.0001$), 18% in the Skinning task ($p = 0.0039$), 34% in the Posing task ($p = 0.0021$), and 14% in the Complete pipeline task ($p = 0.0022$). Differences in accuracy were statistically significant only for the Skinning task, in which the VRI allowed users to be more accurate (47%) than with the BNI ($p = 0.0005$). For the other two tasks, no significant difference was found between the BNI and the VRI (either considering, for the Posing task, the first or the second metric). As said, data for the Complete pipeline task are not available because of the lack of a homogeneous reference. Concerning animation precision, statistically significance was found only for the Posing task,

TABLE 3. A comparison of objective results for the NPUs and the PRUs (when using the same interface). Times, percentages and distances are shown for the three metrics.

Task	$T_{NPUs} - T_{PRUs}$		$A_{NPUs} - A_{PRUs}$		$P_{NPUs} - P_{PRUs}$	
	BNI	*VRI*	*BNI*	*VRI*	*BNI*	*VRI*
Rigging	93.81s p=0.2141	70.30s p=0.1005	–0.18% p=0.4503	–0.51% p=0.2378	–0.0001m p=0.9748	**0.0009m** p=0.0493*
Skinning	44.70s p=0.1927	4.37s p=0.8578	**–1.26%** p=0.0004*	**–0.58%** p=0.0182*	–0.0215 p=0.8323	–0,2461 p=0.3143
Posing	131.03s p=0.2781	44.52s p=0.3599	–1.04% p=0.1398	–0.91% p=0.0061	**0,1566m** p=0.0002*	**0.0024m** p=0.0008*
Compl. Pipel	158.39s p=0.0681	132.69s p=0.0841	Not available	Not available	Not available	Not available

where users were more precise (94%) with the VRI than with the BNI ($p = 0.0001$). The high variances observed in this task are strictly related to results reported in Fig. 8a, and could be related to the fact that, when working in VR, it was easier for the users to ensure that bones were properly aligned with the reference along all the axes.

By focusing on PRUs, trends for completion time and animation accuracy were confirmed. With the VRI, users were 32% faster in the Rigging task, 28% faster in the Skinning task, 34% faster in the Posing task, and 15% faster in the Complete pipeline task. Like for NPUs, differences in accuracy were statistically significant only for the Skinning task (either considering the first or the second metric for the Posing task). The improvement with the VRI was 39% ($p = 0.0011$) with respect to the BNI. This result is particularly interesting, since PRUs had a significant expertise with Blender, but it was the first time that they worked with the VRI. Motivations appeared to be related to a higher learnability and usability of the VRI (which were investigated more in detail through the subjective measurements discussed in the next sub-section). Differences concerning animation precision were non statistically significant.

Comparing the two user categories based on figures conveniently reported in Table 3, PRUs were apparently faster and more accurate than NPUs with both the interfaces. However, these differences proved to be statistically significant (with unpaired t-tests, $p = 0.05$) only for the animation accuracy in the Skinning task. In particular, PRUs increased by 1.26% ($p = 0.0004$) and 0.58% ($p = 0.0182$) compared to NPUs when using the BNI and the VRI, respectively. With respect to the animation precision, NPUs were less precise than PRUs (the average distances from the centroids were higher) in the Rigging task using the VRI (0.0009m, $p = 0.0493$), and in the Posing task using both BNI (0.1533m, $p = 0.0002$) and VRI (0.0024m $p = 0.0008$).

Finally, based on results reported in Table 4, it can be observed that NPUs using the VRI were faster than PRUs using the BNI in both the Skinning ($p = 0.0492$) and the Posing ($p = 0.0481$) tasks (no statistically significant difference was found neither for the the other two tasks nor for animation accuracy). Concerning animation precision, statistically significant differences were obtained only for the Posing task, where NPUs were less precise than PRUs (0.0030m, $p = 0.0121$). These findings suggest that VRI could be helpful in leveling computer animation skills among groups of users characterized by different levels of expertise.

TABLE 4. A comparison of objective results for the NPUs using the VRI and the PRUs using the BNI. Times, percentages and distances are shown for the three metrics.

Task	$T_{NPUs}(VRI) - T_{PRUs}(BNI)$	$A_{NPUs}(VRI) - A_{PRUs}(BNI)$	$P_{NPUs}(VRI) - P_{PRUs}(BNI)$
Rigging	–107.12s p=0.0751	–0.61% p=0.2242	0.0002m p=0.8345
Skinning	**–46.37s** p=0.0492*	–0.15% p=0.4237	–0.04273741 p=0.8414
Posing	**–154.81s** p=0.0481*	–0.95% p=0.0930	**0.0030m** p=0.0121*
Compl. Pipel	2.29s p=0.1037	Not available	Not available

2) SUBJECTIVE RESULTS

In the first section of the questionnaire users were asked to evaluate usability based on the SUS scale [26]. Questions were expressed in the form of statements to be evaluated on a 1-to-5 scale (from strong disagreement to strong agreement). Both NPUs and PRUs found that the VRI was characterized by a higher usability than the BNI (72.03 vs 45.57 for NPUs, 81.00 vs 67.20 for the PRUs). For both user categories the VRI was rated as ''acceptable'', whereas the BNI fall in the ''not acceptable'' range for NPUs and in the ''marginally acceptable'' range for PRUs. As expected, there is a small difference between the SUS scores of the two user categories for the VRI (probably due to the greater confidence of most users with VR technologies), whereas the greater experience with Blender made the PRUs assign to the BNI a usability significantly higher than that assigned by NPUs.

More insights about these results can be obtained by analyzing scores assigned to individual statements reported in Table 5 (bold font is used to identify the interface which was evaluated more positively). Focusing on statistically significant results (marked with the * symbol), it can be observed that the VRI was perceived as easier to learn than the BNI by both the user categories (less things to learn, which can be learnt in a faster way by most people). The two categories also found the VRI as characterized by a more appropriate level of complexity than the BNI, which made the interface easier to use. Moreover, NPUs said they would like to (frequently) use the VRI more than the BNI. They also perceived the system as less cumbersome with the VRI than with the BNI.

TABLE 5. Subjective results concerning usability based on SUS statements [26]).

Statement	NPUs			PRUs		
	BNI	*VRI*	*p*	*BNI*	*VRI*	*p*
1) I think that I would like to use this system frequently	3.22	**4.24**	0.0029*	4.13	4.44	0.6712
2) I found the system unnecessarily complex	3.04	**2.16**	0.0215*	2.43	**1.79**	0.0420*
3) I thought the system was easy to use	3.01	**4.21**	0.0012*	3.15	**4.23**	0.0128*
4) I think that I would need the support of a technical person to be able to use this system	2.36	2.69	0.4119	2.03	1.75	0.0943
5) I found the various functions in this system were well integrated	3.16	3.62	0.3412	4.56	4.47	0.6497
6) I thought there was too much inconsistency in this system	1.92	1.46	0.0870	1.42	1.26	0.2641
7) I would imagine that most people would learn to use this system very quickly	1.55	**4.23**	0.0001*	2.78	**4.12**	0.0003*
8) I found the system very cumbersome to use	3.26	**1.87**	0.0001*	2.20	2.01	0.2451
9) I felt very confident using the system	2.19	**3.45**	0.0318*	3.99	4.02	0.9001
10) I needed to learn a lot of things before I could get going with this system	4.32	**2.76**	0.0003*	3.65	**2.07**	0.0099*

Lastly, NPUs felt more confident in using the VRI than the BNI. No statistically significant difference was found for the remaining statements.

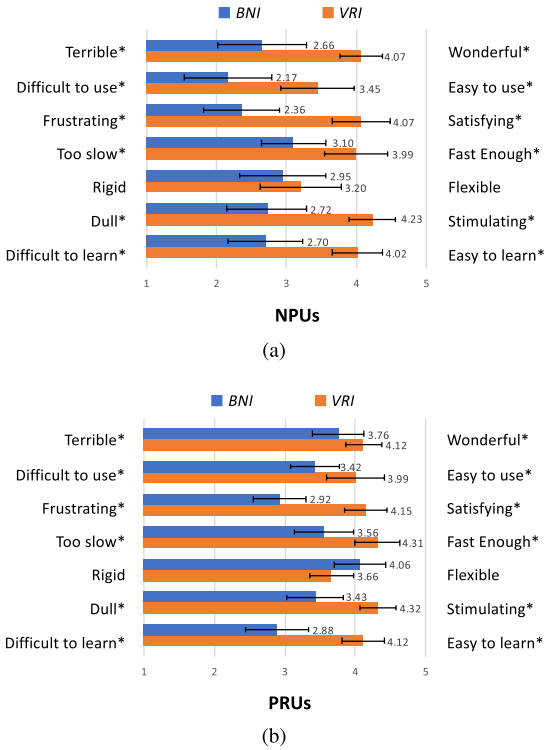


FIGURE 10. Subjective results concerning satisfaction based on criteria (items) in [4] for a) NPUs, and b) PRUs. Average values (bar height) and standard deviations (error bars) are shown.

The second section of the questionnaire requested users to rate several aspects pertaining their satisfaction with the two interfaces, by rating a set of items (used in [4]) on a 1-to-5 scale (adapted from the original 0-to-10 scale, for coherency with the other sections). Average values are reported in Fig. 10. Results confirmed the findings obtained through the first section of the questionnaire. Focusing on statistically significant results (marked with the * symbol), the VRI was perceived as easier to use ($p = 0.0296$ for NPUs and $p = 0.0223$ for PRUs) and to learn ($p = 0.0007$ for NPUs and $p = 0.0001$ for PRUs). These results suggest that VRI can be used by users with different levels of expertise in the field of computer animation and VR. Regarding the remaining aspects evaluated, both NPUs and PRUs expressed a higher appreciation for the VRI than for the BNI. The VRI was perceived as closer to the "wonderful" item ($p = 0.0082$ for NPUs and $p = 0.0095$ for PRUs), more satisfying ($p = 0.0034$ for NPUs and $p = 0.0004$ for PRUs) and stimulating ($p = 0.0001$ for NPUs and $p = 0.0012$ for PRUs). Moreover, as found also through objective measurements, differences in terms of perceived operation speed were statistically significant ($p = 0.0075$ for NPUs and $p = 0.0103$ for PRUs) with higher scores assigned to the VRI. No significant difference was found between the two interfaces for what it concerns flexibility.

TABLE 6. Preferences expressed by the users for the three tasks and for the production of the whole animation (percentages).

Task	NPUs		PRUs	
	BNI	*VRI*	*BNI*	*VRI*
Rigging	21.74%	**78.26%**	30.43%	**69.57%**
Skinning	4.35%	**95.65%**	17.39%	**82.61%**
Posing	26.09%	**73.91%**	21.74%	**78.26%**
Compl. pipel.	34.78%	**65.22%**	21.74%	**78.26%**

Finally, in the last section of the questionnaire, users were asked to indicate their preference between the two interfaces for the execution of each task. Average percentages are reported in Table 6. Both the user categories preferred the VRI to the BNI for the first three tasks executed in isolation and for the last task requesting them to iterate on the rigging, skinning and posing stages. Analyzing in detail individual tasks, it can be observed that the Skinning task was the one which benefited more from using the VRI for both NPUs and PRUs (reasonably due to the simplifications introduced for measurement and comparability purposes, which requested the users to just change weights instead of fully configuring vertex groups). For the remaining tasks, the percentage of NPUs who preferred the VRI was higher in the Rigging task than in the Posing and the Complete pipeline tasks. In the case of PRUs, this order is inverted. This change could be explained by the fact that using the VR headset for a long period (Posing and Complete pipeline tasks are the longest tasks of the experiment) was perceived as more demanding

in term of physical effort by users who were not accustomed with VR. This aspect could have led to a reduced number of preferences for that interface. The percentage of users who preferred the VRI in the Complete pipeline task was lower or equal to the same percentage for the Posing task. Based also on feedback collected, this results was partially due to the fact that users (especially those with limited experience with animation techniques) found it difficult to work with a 2D video reference in a 3D environment.

VI. CONCLUSION

In this paper, the possible advantages and disadvantages associated with the use of VR for the production of character animations have been investigated. By building onto an existing VR-based animation tool integrated in a well-known modeling and animation suite, users have been provided with interface customization capabilities and with VR-based functionalities to deal with the key stages of a complete animation pipeline based on armature deformation, i.e., rigging, skinning and posing. An in-depth comparison of the VR interface with the traditional Blender interface has been performed by considering objective and subjective metrics, and taking into account aspects that pertain both interaction and user experience.

Based on objective measurements collected it has been observed that, through a VR-based interface, users achieve the intended outcome faster than with a traditional mouse and keyboard interface, while maintaining comparable quality in terms of animation accuracy and precision. Subjective observations indicate that both professional and non-professional users were more satisfied with the VR-based interface, which was found to be very easy to use and to learn. These findings further confirm the appropriateness of the proposed interaction method for the considered use case.

Comments gathered from the users at the end of the experiments provide additional insights for future developments. In particular, users suggested to introduce new mechanisms for activating features that are generally accessed via keyboard shortcuts in the Blender's native interface (e.g., to quickly switch between different predefined views, to fine-tune or discretely adjust properties' value, etc.). To this purpose, ad hoc 3DUIs or voice-/gesture-based commands may be exploited. Several users proposed to change some aspects of current interaction design, by introducing, for instance, the possibility to manipulate position, orientation and scale of the Settings panel, or using ray casting for selecting objects, to name a few. Other users, especially NPUs, suggested to replace 2D textures on the Trackpad buttons with 3D icons, in order to immediately recognize the functionality currently activated and partially cope with the limited resolution of the HMD. Finally, some users said they would be interested in the possibility to manage also the modeling steps in VR, benefiting, e.g., of advantages offered by natural interfaces and immersiveness while working with tools for creating meshes that are natively 3D (e.g., the Blender's sculpting mode). This latter comment further stresses the importance of integrated approaches, like the one presented in this paper, highlighting the need for tools which implement a full animation pipeline, possibly blending modeling with animation.

This latter aspect is partially taken into account through the customization approach introduced in this work, as functionalities that can be selected for the custom mapping do not pertain necessarily only animation. However, further work is needed in this direction, aimed to further simplify the configuration process and improve custom VR-based representation (e.g., letting the user choose the textures to be displayed on the controllers' buttons, group functions and controls into visual containers, etc.). Moreover, a dedicated user study shall be designed to quantify the benefits possibly brought by customization in terms of (reduced) mental load and learning cost.

REFERENCES

[1] M. Zyda, "From visual simulation to virtual reality to games," *Computer*, vol. 38, no. 9, pp. 25–32, Sep. 2005.

[2] Y. Usui, K. Sato, and S. Watabe, "Computer graphics animation for objective self-evaluation," *IEEE Comput. Graph. Appl.*, vol. 37, no. 6, pp. 5–9, Nov./Dec. 2017.

[3] D. Vogel, P. Lubos, and F. Steinicke, "AnimationVR—Interactive controller-based animating in virtual reality," in *Proc. IEEE 1st Workshop Animation Virtual Augmented Environ. (ANIVAE)*, Mar. 2018, pp. 1–6.

[4] N. Pantuwong, "A tangible interface for 3D character animation using augmented reality technology," in *Proc. 8th Int. Conf. Inf. Technol. Elect. Eng. (ICITEE)*, Oct. 2016, pp. 1–6.

[5] R. Henrikson, B. Araujo, F. Chevalier, K. Singh, and R. Balakrishnan, "Multi-device storyboards for cinematic narratives in VR," in *Proc. 29th Annu. Symp. Interface Softw. Technol.*, 2016, pp. 787–796.

[6] R. Pausch, J. Snoddy, R. Taylor, S. Watson, and E. Haseltine, "Disney's Aladdin: First steps toward storytelling in virtual reality," in *Proc. 23rd Annu. Conf. Comput. Graph. Interact. Techn.*, 1996, pp. 193–203.

[7] F. Lamberti, G. Paravati, V. Gatteschi, A. Cannavò, and P. Montuschi, "Virtual character animation based on affordable motion capture and reconfigurable tangible interfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 5, pp. 1742–1755, May 2017.

[8] N. Osawa and K. Asai, "An immersive path editor for keyframe animation using hand direct manipulation and 3D gearbox widgets," in *Proc. 7th Int. Conf. Inf. Vis.*, Jul. 2003, pp. 524–529.

[9] F. Lamberti, A. Cannavò, and P. Montuschi, "Is immersive virtual reality the ultimate interface for 3D animators?" *IEEE Comput. Mag.*, to be published.

[10] B. M. Lunt, J. J. Ekstrom, S. Gorka, G. Hislop, R. Kamali, E. Lawson, R. LeBlanc, J. Miller, and H. Reichgelt, "Curriculum guidelines for undergraduate degree programs in information technology," Tech. Rep., Nov. 2008, pp. 1–139. [Online]. Available: https://dl.acm.org/citation.cfm?id=2593311

[11] I. Baran and J. Popović, "Automatic rigging and animation of 3D characters," *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 72.

[12] J. Pan, X. Yang, X. Xie, P. Willis, and J. J. Zhang, "Automatic rigging for animation characters with 3D silhouette," *Comput. Animation Virtual Worlds*, vol. 20, nos. 2–3, pp. 121–131, 2009.

[13] A. C. Silva, L. R. Mattioli, G. de Paula, A. Cardoso, E. A. Lamounier, G. F. M. de Lima, P. R. M. do Prado, and J. N. Ferreira, "A strategy to present 2D information within a virtual reality application," in *Proc. 16th Symp. Virtual Augmented Reality*, May 2014, pp. 143–147.

[14] M. F. Deering, "HoloSketch: A virtual reality sketching/animation tool," *ACM Trans. Comput.-Hum. Interact.*, vol. 2, no. 3, pp. 220–238, 1995.

[15] B. Jackson and D. F. Keefe, "Lift-off: Using reference imagery and freehand sketching to create 3D models in VR," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 4, pp. 1442–1451, Apr. 2016.

[16] A. Cannavò and F. Lamberti, "A virtual character posing system based on reconfigurable tangible user interfaces and immersive virtual reality," in *Proc. Smart Tools Apps Graph.-Eurographics Italian Chapter Conf.*, 2018, pp. 1–11.

[17] A. Fender, J. Müller, and D. Lindlbauer, "Creature teacher: A performance-based animation system for creating cyclic movements," in *Proc. 3rd ACM Symp. Spatial Interact.*, 2015, pp. 113–122.

[18] R. Beimler, G. Bruder, and F. Steinicke, "SmurVEbox: A smart multi-user real-time virtual environment for generating character animations," in *Proc. Virtual Reality Int. Conf., Laval Virtual*, 2013, Art. no. 1.
[19] E. Bernier, R. Chellali, I. M. Thouvenin, and K. Blom, "The ICEA plug-in for virtual reality, immersive creation and edition of animation," in *Proc. 5th Workshop Softw. Eng. Archit. Realtime Interact. Syst. (SEARIS)*, Mar. 2012, pp. 36–42.
[20] A. Fernández-Baena, L. Salle, and D. Miralles, "Avatar: Tangible interaction and augmented reality in character animation," in *Proc. Interact. Smart Objects*, 2014, p. 28.
[21] M. Eitsuka and M. Hirakawa, "Authoring animations of virtual objects in augmented reality-based 3D space," in *Proc. 2nd IIAI Int. Conf. Adv. Appl. Inform.*, Aug./Sep. 2013, pp. 256–261.
[22] B. Berford, C. Diaz-Padron, T. Kaleas, I. Oz, and D. Penney, "Building an animation pipeline for VR stories," in *Proc. ACM SIGGRAPH Talks*, 2017, pp. 64:1–64:2.
[23] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 35–58, 2018.
[24] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "HTC vive: Analysis and accuracy improvement," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 2610–2615.
[25] D. C. Niehorster, L. Li, and M. Lappe, "The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research," *I-Perception*, vol. 8, no. 3, pp. 1–23, 2017.
[26] J. Brooke, "SUS-A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.



ALBERTO CANNAVÒ received the M.Sc. degree in computer engineering from the Politecnico di Torino, Italy, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Control and Computer Engineering. His research interests include computer graphics and human–computer interaction.



CLAUDIO DEMARTINI is currently a Full Professor with the Politecnico di Torino, where he teaches information systems as well as innovation and product development. He is also the Chair of the Department of Control and Computer Engineering and a member of the Academic Senate of the Politecnico di Torino. His research interests include software engineering, architectures, intelligent systems, and education.



LIA MORRA received the M.Sc. and Ph.D. degrees in computer engineering from the Politecnico di Torino, Italy, in 2002 and 2006, respectively, where she is currently a Senior Postdoctoral Fellow with the Department of Control and Computer Engineering. From 2006 to 2016, she was with im3D, where she served as the Chief Scientific Officer, from 2014 to 2017; she guided the team developing the core medical image analysis and artificial intelligence technology components from design to regulatory approval. She has authored numerous articles in international journals and conferences and holds three patents. Her main research interests include computer vision, machine learning, and computer-assisted medical image interpretation. She served as a member of the AAPM Computer Aided Image Analysis Subcommittee.



FABRIZIO LAMBERTI received the M.Sc. and Ph.D. degrees in computer engineering from the Politecnico di Torino, Italy, where he is currently an Associate Professor with the Department of Control and Computer Engineering. He has published more than 160 articles in international books, journals, and conference proceedings mainly in the areas of computer graphics, human–computer interaction, intelligent systems, and educational computing. He is a Senior Member of the IEEE Computer Society and a member of the IEEE Computer Society's Technical Committee on Visualization and Graphics. He serves as the Secretary/Treasurer for the IEEE Computer Society, Italy Chapter. He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, and the IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES. He is also an Executive Editor of the *IEEE Consumer Electronics Magazine*.