A sketch-based interface for facial animation in immersive virtual reality

A Sketch-based Interface for Facial Animation in Immersive Virtual Reality

Alberto Cannavò[a], Emanuele Stellini[a], Congyi Zhang[b] and Fabrizio Lamberti[a]

[a] Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin, Italy;[b]Department of Computer Science, The University of Hong Kong, Pokfulam Road 4/F, Hong Kong

ARTICLE HISTORY
Compiled February 1, 2023

ABSTRACT
Creating facial animations using 3D computer graphics represents a very laborious and time-consuming task. Among the numberless approaches for animating faces, the use of blendshapes remains the most common solution because of their simplicity and the ability to produce high-quality results. This approach, however, is also characterized by important drawbacks. With the traditional animation suites, in order to select the blendshapes to be activated animators are generally requested to memorize the mapping between the blendshapes and the influenced mesh vertices; alternatively, they need to adopt a trial-and-error search within the whole library of available blendshapes. Moreover, the level of expressiveness that can be reached may be lower than expected; this is due to the fact that the possibility to apply transformations to vertices different than just linear translations and mechanisms for adding, e.g., exaggerations, are typically not integrated into the same animation environment. In order to tackle these issues, this paper proposes an immersive virtual reality-based interface that leverages sketches for the direct manipulation of blendshapes. Animators can draw both linear and curved strokes, which are used to automatically extract information about the blendshape to be activated and its weight, the trajectories that associated vertices have to follow, as well as the timing of the overall animation. A user study was carried out with the aim of evaluating the proposed approach onto several representative animations tasks. Both objective and subjective measurements were collected. Experimental results showed the benefits of the devised interface in terms of task completion time, animation accuracy, and usability.

KEYWORDS
Facial animation; blendshape; direct manipulation; sketch-based interface; virtual reality

1. Introduction

Facial animation in computer graphics plays a crucial role in enriching, among others, the entertainment, education and communication domains (N. Liu et al. (2021)), since it allows digital characters to come to life. The production of movies, video games and other interactive content is thus pushing the development of new interfaces able to make the creation of 3D animations more intuitive, fast and effective (Miranda et al. (2012)). Nevertheless, software tools for animating faces are still very complex

CONTACT F. Lamberti. Email: fabrizio.lamberti@polito.it

to use. This fact makes the production of animations with the intended quality a very laborious and time-consuming task, requiring both artistic and technical skills (Cetinaslan and Orvalho (2020a)).

Among the many techniques devised for facial animation, the approach based on *blendshapes* is one of the most commonly adopted (Cetinaslan and Orvalho (2020b)) because of its simplicity and the possibility to produce also high-quality results (Lewis et al. (2014)). This technique makes use of a weighted linear combination of parameterized "target faces", each represented by a blendshape and describing a specific expression. Once target faces have been created, e.g., by a skilled artist or scanning a human actor's face, and stored in a blendshape library, their weights in the intended facial pose (i.e., the intensity of the deformation) can be defined through dedicated editing interfaces. In existing animation tools, these interfaces are generally based on sliders and numeric fields (Cetinaslan and Orvalho (2020b))

Alternatively to these user interface (UI) widgets, animators can take advantage of more sophisticated facial "armatures" or "rigs", which allow them to control the blendshapes and/or the mesh vertices by leveraging elements referred to as "bones".

Although the manipulation of widgets and bones relies on simple operations (e.g., moving sliders or grabbing bones around in the 3D space), the creation of the sequence of facial poses defining the animation via traditional tools is still challenging for the animators. In fact, they are requested to either memorize a number of items from rather high-dimensional spaces, or to perform a trial-and-error search into them (Cetinaslan and Orvalho (2020a)), since realistic facial models may leverage thousands of vertices and more than one hundred blendshapes may have to be combined for shaping a given expression (Cetinaslan and Orvalho (2020b)).

Furthermore, using only the approach based on blendshapes could be limiting for the animators for two main reasons. The first reason is related to the fact that the movements generated by animating the weight of blendshapes typically include only linear translational trajectories between two poses. In order to obtain more complex movements (involving, for instance, rotations), animators need to combine blendshapes with free-form deformations (FFD) that are typically obtained by means of lattices or additional bones (Barr (1984)). The second reason concerns the level of expressiveness that can be achieved. Using blendshapes, new expressions can be obtained only as a weighted combination of the existing blendshapes. This fact makes it impossible, for instance, to generate expressions with exaggerated characteristics like caricatures (N. Liu et al. (2021)), as they are usually not included in the blendshape library (and are generated via FFD). As the complexity of the model or the animation to create increases, the combination of blendshapes with FFD could be difficult to pursue.

Researchers are continually experimenting with alternative interaction paradigms to overcome the above limitations. Among the various methods proposed in the literature, sketch-based interfaces are becoming quite popular, and represent today a promising means to intuitively create, deform and control 3D models (Miranda et al. (2012)). The idea behind the use of sketch-based interfaces is to allow artists and animators to operate with a familiar gesture (drawing of strokes) instead of dealing with hundreds of different rigging and/or deformation control parameters (Cetinaslan and Orvalho (2018)), which makes them suitable to users with different levels of expertise (Miranda et al. (2012), Cetinaslan and Orvalho (2018)). The use of these interfaces has been investigated, e.g., in shape and 3D surface creation (Jiang et al. (2021), Bae, Balakrishnan, and Singh (2008), Igarashi, Matsuoka, and Tanaka (2006)), modelling (Li et al. (2018), Jackson and Keefe (2016)), character posing (Hahn et al. (2015), Cannavò, Zhang, Wang, and Lamberti (2020)), and animating tasks (Thorne, Burke,

and van de Panne (2004), Mao, Qin, and Wright (2007)).

Despite the benefits possibly coming from the adoption of sketch-based interfaces, their usage still remains limited. This is due to the fact that the massive use of 2D devices for working with traditional computer graphics tools introduces issues like, e.g., interpreting 2D lines in 3D or estimating depth while drawing (Choi et al. (2016), Barbieri, Garau, Hu, Xiao, and Yang (2016)). To overcome the constrained dimensionality of the input and output devices, the research community started to look at the possibility to leverage virtual reality (VR) as a means not only for visualizing animated content but also to generate them (Gipson et al. (2018)). A key benefit of using VR is the possibility to visualize and manipulate elements through interfaces that are natively 3D (Lamberti, Cannavo, and Montuschi (2020)). Moreover, the improved sense of presence offered by VR can make the animators feel as part of the virtual environment in which the animation takes place, thus enhancing their creativity and productivity (Henrikson, Araujo, Chevalier, Singh, and Balakrishnan (2016)). The high interest in VR tools for creating 3D character animations is witnessed, e.g., by the launch of initiatives like PoseVR[1], a project managed by Walt Disney Animation Studios.

Research prototypes (e.g., in Vogel, Lubos, and Steinicke (2018), Osawa and Asai (2003), and Pantuwong (2016)) and commercial products (e.g., AnimVR[2], Facebook Quill[3], Tvori[4] and Masterpiece[5]) also started to be developed, targeting both professional users and beginners. To the best of the authors' knowledge, however, most of these tools feature simplified interfaces offering only a limited set of functionalities. Moreover, they are not designed to be integrated within common animation suites, preventing animators to re-use created content or edit it using state-of-the-art animation functionalities that are natively provided by those suites.

Moving from considering the challenges mentioned above, this paper presents a novel interface that combines the advantages offered by sketch-based interaction and VR for the production of facial animations. With the proposed interface, the users can animate blendshapes by means of strokes that are drawn within an immersive environment. Strokes are used not only to automatically select the blendshapes to be activated and set their weight, as done in previous works (N. Liu et al. (2021), Miranda et al. (2012), Cetinaslan and Orvalho (2020a)), but also to specify the trajectories that the mesh vertices have to follow during the transition between two expressions (i.e., *position interpolation*). In this way, the users can have more control over the generated animation, since they can define trajectories containing also rotations in addition to generally used linear translations. Besides spatial control, the proposed interface offers the opportunity to control the timing of the animation, by capturing the movement of the users' hand while drawing. In particular, the time taken to trace the stroke defines the duration of the animation, whereas the speed variations provide control over the so-called "easing" of the animation (*timing interpolation*). In this way, the users have the opportunity to create complete animations with a single gesture.

The proposed animation method was developed as an add-on for a well-known modeling and animation suite (Blender[6]), by providing native integration with its interface. The code of the add-on is available on Github at `https://github.com/`

[1]PoseVR: `https://disneyanimation.com/technology/posevr/`
[2]AnimVR: `https://nvrmind.io/`
[3]Facebook Quill: `https://quill.fb.com/`
[4]Tvori: `https://tvori.co/`
[5]Masterpiece VR: `https://masterpiecestudio.com/`
[6]Blender: `https://www.blender.org/`

`lab2atpolito/FaceAnimationVR`.

In order to assess the performance of the proposed system, a user study was carried out by involving 35 subjects. According to obtained results, the proposed interface allowed the participants to be faster in completing all the four tasks than with the native interface, and more accurate (in terms of being able to reproduce, spatially and temporally, the reference animation) on two of them. Moreover, advantages were observed in terms of usability.

The rest of the paper is organized as follows. Section 2 provides a review of works concerning the use of sketch- and VR-based interfaces for facial animation. In Section 3, a detailed description of the proposed system is provided. Section 4 illustrates the experimental setup that was used to evaluate the system. Section 5 reports the results obtained in the user study. Lastly, in Section 6, conclusions and possible directions for future research in this field are discussed.

2. Related work

Facial animation has always been extremely stimulating for the research community, since producing synthetic faces for humanoid and non-humanoid characters and providing them with the intended expressiveness come with a countless number of challenges (Lewis and Anjyo (2010)).

The literature offers a variety of approaches that allow animators to manipulate and/or deform meshes to create facial animations. For instance, there are techniques based on parametric models that implement deformations by means of algorithms specifically designed for the considered geometries (Parke (1974)), or approaches building onto physically based models that approximate the mechanical properties of the face like, e.g., muscles, tissues, bones (Sifakis, Neverov, and Fedkiw (2005)). Other works reported approaches leveraging principal component analysis (PCA) to animate expressions in face scans (Blanz, Basso, Poggio, and Vetter (2003)), or systems for face modeling and animation based on high-resolution scans obtained through synchronized video cameras and structured light projectors (Zhang, Snavely, Curless, and Seitz (2004)). The literature also contains examples of methods that apply deformations driven by motion capture data (Beeler et al. (2011)), or tackle the generation (Karras, Aila, Laine, Herva, and Lehtinen (2017)) and the editing (Berson, Soladie, Barrielle, and Stoiber (2019)) of facial animations by using machine learning. Finally, algorithms for the automatic creation of in-situ UI control layouts for deformable face rigs (J. Kim and Singh (2021)) have also been explored.

Research on facial animation also studied new interaction paradigms and input devices capable of easing the job of animators. In this context, it is not surprising that particular attention has been devoted to sketch-based interfaces. In fact, sketches are already adopted in many steps of the creative process, e.g., for defining shapes, exploring motion with rough key poses, drawing storyboards, etc. (Gouvatsos, Xiao, Marsden, and Zhang (2017), Guay, Cani, and Ronfard (2013). Sketch-based interfaces have already been used also for manipulating facial meshes. For instance, the authors of Chang and Jenkins (2006) presented a methodology for articulating and posing facial meshes through 2D sketching. The devised methodology requests the users to draw two different strokes, i.e., the reference and the target curve. The former is used to select the features of the face mesh to be manipulated, whereas the latter specifies the deformations to be matched. The pose that the face has to assume is computed by finding the amount of deformation that minimizes the distance between the reference

and the target curve. The animation system proposed in that paper supported different types of curve interpolations, encompassing not only translations, but also rotations and scaling. Given the use of 2D sketching, strokes representing the reference and target curves were drawn on a plane (named the image plane) and then projected onto the mesh. The system used only FFD, which could be a drawback since it may lead to non-realistic results (N. Liu et al. (2021)). In Gunnarsson and Maddock (2010), another sketch-based interface was proposed for the generation of facial animations through interpolation of a sequence of drawn key poses. In this case, only one 2D stroke is provided as input, and the proposed system is able to automatically reconstruct the 3D features that fit the stroke. However, the use of a 2D input could be a limiting factor for the animators' expressiveness, since they have to imagine how the 2D sketch is projected on the 3D surface or apply several rotations to the viewpoint to control different parts of the face.

With the system that was reported in Miranda et al. (2012), the users can sketch their strokes either directly on a 3D character or on two different types of surfaces, i.e., a 2D fixed canvas or a dynamic screen-aligned billboard. The system assumes that the face model is provided with a rig, and the strokes are used by the animation system not to directly deform the geometry, but rather to modify the rigging parameters and pose the model. The system is also able to retarget the strokes to different models. A similar sketch-based posing system was proposed in Hahn et al. (2015) to support the articulation of rigged characters and/or faces through an iconographic 2D representation of the model named "sketch abstraction". The abstraction, drawn on top of the actual shape of the model, can represent both a sticky version of the shape or its outline. The system computes the new pose by finding the rigging parameters that best align the sketch abstraction to the input sketch. Although the last two systems support 3D interaction, they work on a rigged model that is deformed by manipulating its rigging parameters.

The analysis of the above works outlines two possible limitations. The first one concerns the constraints introduced by the methodology adopted for deforming the mesh, based on FFD (Chang and Jenkins (2006)) or on rigged models (Hahn et al. (2015); Miranda et al. (2012)). This limitation can translate into a reduced level of expressiveness for the animators and/or lead to poorly realistic results. The second limitation pertains the use of 2D interfaces to operate on 3D contents, which can affect the naturalness and effectiveness of the interaction with the animation system (Gunnarsson and Maddock (2010)).

To overcome the first limitation, this paper proposes a methodology to obtain realistic and expressive results grounded on blendshapes. Blendshapes were selected since they represent the most common approach to facial animation thanks to their simplicity, expressiveness and interoperability (Lewis et al. (2014)). Blendshapes were used, e.g., to animate characters in movies such as The curious case of Benjamin Button (Flueckiger (2011)), King Kong (Sagar (2006)), and The Lord of the Rings (Singer (2003)), to name a few. Blendshapes also support the creation of other graphics content. For instance, they are integrated in common graphics game engines like, e.g., Unity[7] and Unreal Engine[8]. Moreover, blendshapes have been used in social VR platforms like VRChat[9] to generate the talking animations of avatars based on the phonemes uttered by the users.

[7]Unity: `https://docs.unity3d.com/Manual/BlendShapes.html`

[8]Unreal Engine: `https://docs.unrealengine.com/5.0/en-US/fbx-morph-target-pipeline-in-unreal-engine/`

[9]VRChat: `https://hello.vrchat.com/`

In order to ease the cumbersome editing of the blendshapes weight, this paper leverages the methodology named *Direct-Manipulation Blendshapes* (DMB) in Lewis and Anjyo (2010), which allows animators to control the weights with a practical interface based on few degrees of freedom (Anjyo, Todo, and Lewis (2012)). For instance, in Cetinaslan, Orvalho, and Lewis (2015), the DMB technique was used in combination with a sketch-based interface supporting "pin-and-drag" operations for controlling the blendshapes; with the proposed system, once the users have drawn a stroke representing the intended deformations onto a 3D facial model, the weights to be assigned to the blendshapes are automatically computed in order to obtain the desired facial pose. In Cetinaslan and Orvalho (2018, 2020a), the same authors recently proposed a number of improvements to this technique, fixing issues regarding the computation of weights and making it possible to influence only local parts of the 3D model.

The positive results reported in these works show the applicability of DMB to the creation of facial poses. At the same time, the lack of consideration for the animation task indicates that there is space for applying this methodology also in such a context. Moreover, works seen so far are based only on blendshapes: hence, expressions that can be obtained are limited to the library of blendshapes defined for the model. To overcome this constraint and allow animators to add exaggerated features to the face (as shown, e.g., in N. Liu et al. (2021)), the approach proposed in this paper combines DMB with FFD, thus letting animators not just to generate the poses, but to animate the face models with ad hoc functionalities.

In order to use DMB and FFD for animation, a mechanism for defining motions had to be defined. To this purpose, this paper builds on previous literature and investigates the use of sketches not only as a way for specifying the deformations to be applied to the face model, i.e., for defining its poses (as done in Cetinaslan and Orvalho (2018, 2020a, 2020b)), but also to setup the parameters (for position and timing) to be used for animating the transitions between poses. A method tackling a similar problem was proposed in Ciccone, Öztireli, and Sumner (2019) to simplify the generation of smooth animations. In that work, once the animators had defined the key poses of a character, the animation system provided them with an interface to easily edit the in-between frames with curves showing the trajectories of the involved joints. In this way, the animators could control the motion of joints by imposing positional constraints over time. Inspired by this method designed for body animations, in this paper a methodology is defined to control the interpolation of facial poses.

As mentioned, the second limitation observed in previous works regards the impact of 2D interfaces on the intuitiveness of interaction with 3D content. To cope with this issue, this paper looks at the benefits possibly brought by the use of VR technology. In fact, a number of studies were carried out to assess the use of VR for character animations, e.g., Lamberti et al. (2020), Vogel et al. (2018), and Bernier, Chellali, Thouvenin, and Blom (2012). Focusing on works that leveraged 3D sketches, an example of a VR system for posing virtual characters was presented in Cannavò, Zhang, et al. (2020). With such a system, animators can articulate a rigged character by drawing strokes into an immersive environment. Once sketches are drawn, the system tries to solve an optimization problem to minimize the distance between the input strokes and character pose. Differently than in other works, that system was integrated into a well-known modeling and animation suite, thus letting animators reuse the articulated characters exploiting the advanced animation functionalities of the underlying software without the need of additional import/export operations. This feature has been considered also in the design of the animation interface that is proposed in this paper.

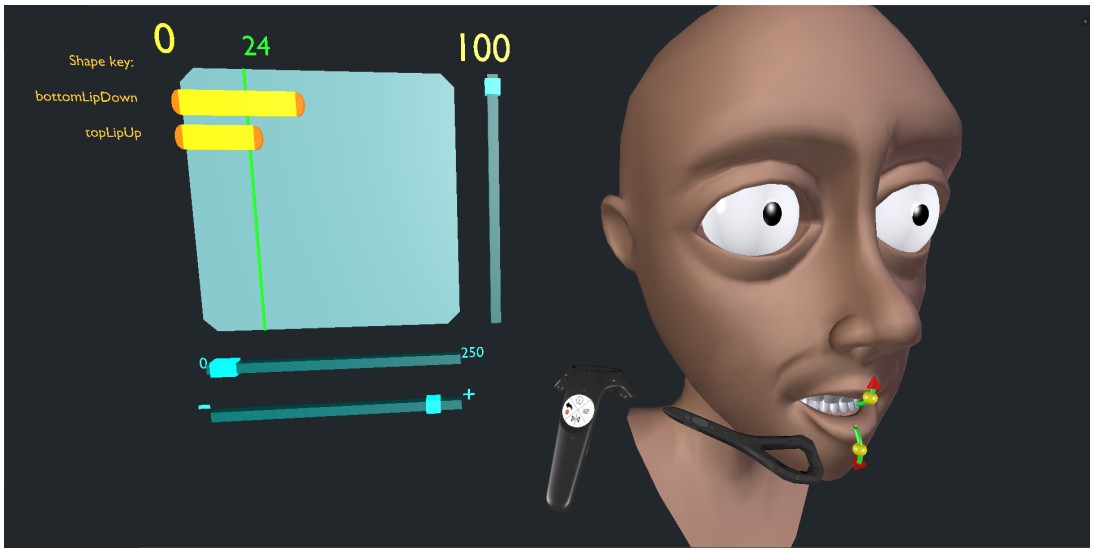


Figure 1. Overview of the proposed interface: components of the GUI (on the left), virtual representation in the immersive environment of the input devices (center), and digital character being animated (on the right).

In conclusion, a key contribution of this paper is to present a novel interface for facial animation able to combine DMB with FFD, thus increasing the number of expressions that can be obtained. Moreover, the devised interface allows animators to control both the position and time interpolation of facial poses by leveraging a sketch-based interface in VR. In this way, 3D sketching can be used to improve the naturalness and effectiveness of the interaction with the animation system. Finally, the interface is integrated into a well-know animation suite; this choice makes the reuse of created content and their modification (using the native functionalities of the considered suite) easy to accomplish, without any import/export or pre-/post-processing operation.

3. System overview

As discussed, the aim of this paper is to present a sketch-based interface, depicted in Fig. 1, which allows the users to create facial animations by manipulating blendshapes in an immersive virtual environment. The devised interface is integrated into an animation system whose architecture is illustrated in Fig. 2.

The animators can interact with the system using different *Input/Output devices*. The core of the system is represented by the Blender modeling and animation suite. This tool was selected for hosting the proposed interface since it offers most of the functionalities required to animate faces based on blendshapes and FFD. Moreover, being an open-source project, it provides a fully scriptable environment that lets developers create dedicated add-ons that can be used to extend its functionalities. The possibility to add custom scripts made it possible to combine into a single environment the natively available modeling/animation features with the new, sketch-based interface. Finally, since Blender is a well-know tool used by a large number of artists, operating directly into it makes the application of further changes to created content extremely straightforward. The add-on implementing the proposed interface was developed targeting Blender 2.79.

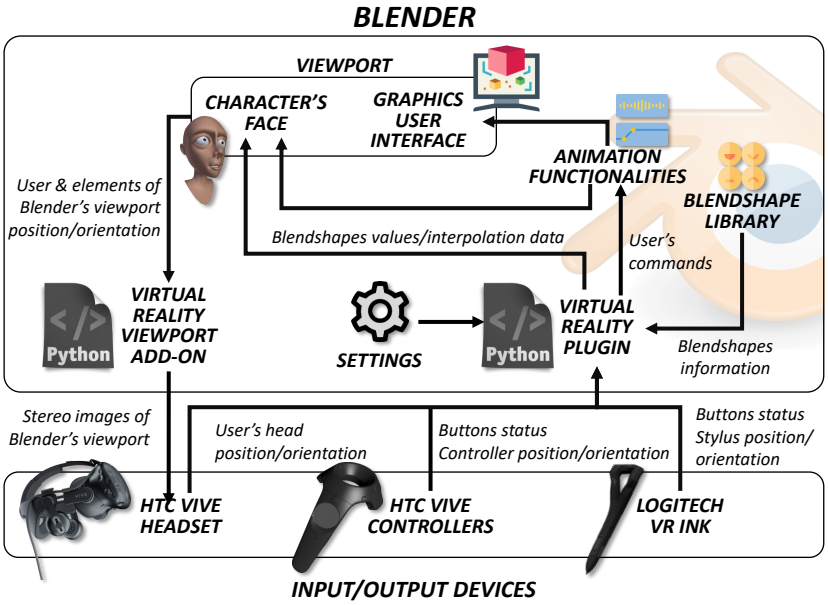


Figure 2. Overall architecture of the animation system.

In order to enable VR interactions, the HTC Vive VR system[10] was used. This system was chosen since it is able to track a VR headset plus a number of additional devices (controllers and other props) in a room-scale environment with high accuracy and low latency. The headset can be used to let the animator visualize the 3D models to be animated into an immersive environment, and observe them from the most appropriate viewpoints. The origin of the virtual environment corresponds to the position of the Blender main camera in the 3D viewport. This position is mapped on the center of the HTC Vive's room, whereas its orientation is used to specify the orientation of the reference system. The scale factor along the three axes of the Blender camera provides the ratio between the real displacements in the tracked space (measured in meters) and the displacements in the virtual space (measured in Blender units). A one-to-one mapping (i.e., one Blender unit corresponds to one meter) is used as a default setting, though the mapping can be changed at anytime by the animator.

In the current implementation, the connection with the HTC Vive system is based on the following libraries: the *Virtual Reality Viewport add-on*[11] that is used to generate the stereo images required to visualize the Blender viewport through the headset, and the *pyopenvr* SDK[12] that enables Python-based access to controllers and other VR devices. In order to support newer Blender versions, some changes would be required. In particular, starting from Blender 2.83 the Virtual Reality Viewport add-on should be replaced with the integrated *VR Scene Inspection* add-on; moreover, a new version of the pyopenvr SDK, named *pyopenxr*[13] and supporting also the OpenXR standard, could be used.

The input devices used in the proposed interface include both a Logitech VR Ink

[10]HTC Vive: `https://www.vive.com`
[11]Virtual Reality Viewport: `https://github.com/dfelinto/`
[12]https://github.com/cmbruns/pyopenvr
[13]`https://github.com/cmbruns/pyopenxr`

stylus[14] and an HTC Vive hand controller. The VR Ink was used as a sketching tool since its shape makes it clearer for the animator the actual starting point of the sketch, letting him or her draw more accurately than using a controller (Cannavò, Calandra, Kehoe, and Lamberti (2020)). However, it was not possible to map all the functionalities needed to manage the animation process onto the controls available on the VR Ink. For this reason, the devised animation interface adopts an interaction paradigm based on both the devices, with the animators being suggested to operate the VR Ink with the dominant hand and to use the controller with the other hand. The literature confirms the advantages of this asymmetric configuration based on a dedicated stylus and a controller. As a matter of example, the pilot study reported in (Zou, Bai, Gao, Fowler, and Billinghurst (2022)) indicated this configuration as the most suitable for artistic applications compared to various symmetric and asymmetric setups, since it obtained the highest number of preferences and very good usability scores from the study participants.

It is worth observing that, although in the selected setup a stylus was used, other input devices like, e.g., a 3D mouse, could be exploited, in principle, to control the sketching. Even the output should not be necessarily based on a VR headset: in fact, with the devised implementation, the users may decide to operate the VR Ink and the HTC Vive controller while sitting, e.g., in front of a desktop display. In this case, they would lose the advantages of immersive manipulation and visualization, but could still benefit of 3D interaction and would not suffer the annoyance associated with the possible need to switch between the native Blender interface (to create the models, the blendshapes, etc.) and the VR environment (to animate the characters). Nevertheless, as witnessed by promising developments in the field (Lamberti et al. (2020), Cannavò, Demartini, Morra, and Lamberti (2019), B. Kim (2019)), in the future modeling and animation functionalities directly available in VR could grow further, progressively reducing the need to unwear the headset for carrying out some of the tasks.

The animation workflow starts with the user drawing sketches in the air through the stylus. After having applied beautification to the strokes, the system uses the provided inputs to select the blendshapes to be activated. From the drawn strokes, it also extracts the weights to set for the blendshapes, the timing as well as the trajectories of the animation. Once the system has created the animation, the user can perform a number of operations like navigating the timeline, applying changes to the animation timing, modifying the trajectories of the generated motions, etc. The *Virtual Reality plugin* is in charge of recognizing the user inputs and of activating the corresponding Blender animation functionalities. More details about each step in this workflow will be provided in the following subsections.

3.1. User inputs

The interaction with the system is managed through a state machine. In each state, the system gives a different meaning to the inputs provided by the user by activating different mappings on the buttons of the HTC Vive controller and the VR Ink. Mappings are reported in Fig. 3. A visual feedback, shown in the VR environment as a text along the side of the controller or the stylus, lets the user recognize the current state. States and related functionalities are described below.

- **Waiting**: The system is waiting for a specific input from the user, who, for

[14]VR Ink: `https://www.logitech.com/en-eu/promo/vr-ink.html`

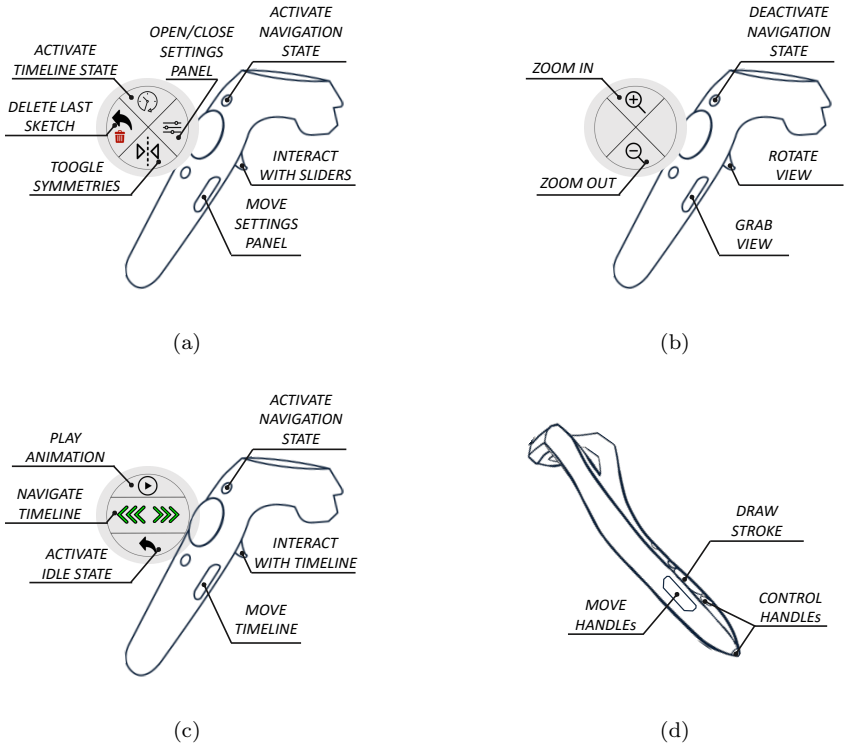


Figure 3. Functionalities accessible through the buttons of the HTC Vive controller in the a) Waiting, b) Navigation, and c) Timeline state, and of the VR Ink in the d) Interaction state.

instance, may be exploring the VR environment changing his or her point of view.

- **Interaction**: By using the button of the VR Ink (i.e., Trackpad, Gripper, and Trigger), the user can interact with the animation system, e.g., drawing new strokes, modifying their trajectories, applying changes to the time interpolations, etc.
- **Navigation**: By pressing the Menu button of the controller, the system moves to the Navigation state, where the user can apply transformations (i.e., translations, rotations, and scaling) to the Blender main camera with the final goal of altering the mapping between the tracked space and the virtual environment.
- **Timeline**: This state, which is activated with the Trackpad Up button of the controller, lets the user interact with the timeline, e.g., to change the visualization of the frames, modify the duration of the animation, synchronize the movement of different parts of the model, etc.

As shown in Fig. 3, the buttons of the VR Ink are used to draw new strokes or interact with their handles with the goal of manipulating the resulting animation. While drawing, the system collects the 3D coordinates of the points belonging to the strokes as well as the timing of the drawing gesture. These data are used by the system to control the trajectories to be used for the transition between two blendshapes (i.e., to perform position interpolation) and the timing of the animation (i.e., to perform time interpolation). With the buttons of the controller, the user can issue commands to the system, e.g., to insert keyframes, open the panels or interact with them, control the playback of the animation, etc. More details on the supported functionalities as

well as on interpolation mechanisms will be given hereafter.

3.2. Blendshapes selection

As depicted in Fig. 3d, a new stroke can be added by pressing the Trackpad button on the top of the VR Ink. The system continues to sample the 3D position of the stylus tip with a frequency of 90Hz until the user releases the button. Once the drawing gesture has ended, the system calculates a curve by forcing it to interpolate all the collected points. In Blender, each stroke is converted into a Bezier curve whose control points are defined by the collected samples.

The mechanism adopted by the proposed system to select a blendshape assumes that the user starts to draw close to the part of the face he or she wants to animate and terminates the stroke near the intended deformation for which a blendshape exists. Under the above hypothesis, it is possible to use the starting and ending points of the curve to control the animation (Fig. 4).

In particular, the starting point of the stroke is used to find the part of the geometry that is involved in the animation (Fig. 4a). By leveraging the K-Dimensional tree optimization method (Bentley (1975)), the vertex of the mesh that is closest to the starting point of the stroke is first identified (in the following, this vertex will be referred to as *reference vertex*). Once the reference vertex has been identified, it is possible to use it together with the ending point of the stroke to select the blendshape to be activated (later referred to as the *target blendshape*). More specifically, the target blendshape is determined by using the following equation:

$$target_{blendshape} = argmin_i \{d(s_{end\_point}, t_i)\} \quad i \in [1, N]\,. \tag{1}$$

According to Eq. 1, the target blendshape ($target_{blendshape}$) is determined by computing the distance $d(\cdot)$ between the ending point of the stroke (s_{end_point}) and the positions of the reference vertex with the maximum deformation applied for each blendshape i (in the following named *target position*, t_i), with $i \in [1, N]$ and N being the number of blendshapes altering the position of the reference vertex.

In order to help the user to identify the blendshapes that could be activated, a visual feedback is provided by the system. More specifically, as he or she starts drawing, several red spheres appear, representing the target positions reached by the reference vertex in all the considered blendshapes (Fig. 4b). These spheres suggest to the user possible ending points for the stroke (as illustrated in the next subsection, by choosing where to end the stroke he or she will be able to control the weight of the blendshape).

The blendshape for which the minimum distance is calculated represents the target blendshape. For instance, in the figure, t_1 is selected as the target blendshape since it is the one minimizing the distance to the ending point of the stroke among the four target positions shown.

Possible errors in the selection of the blendshape could occur, especially when the target positions are particularly close to each other (e.g., t_3 and t_4 in Fig. 4b). It is worth observing that the presence of close target positions is an indication of the fact that the selected reference vertex is not well-representative of the deformation. For example, t_3 and t_4 represent the maximum deformation of the "LipSpursed" and "TopLipUp" blendshapes, respectively. A reference vertex chosen on the middle of the lip would have been more representative of the movement performed by the mouth when deformed with these blendshape than a vertice on the corner of the lip. In these

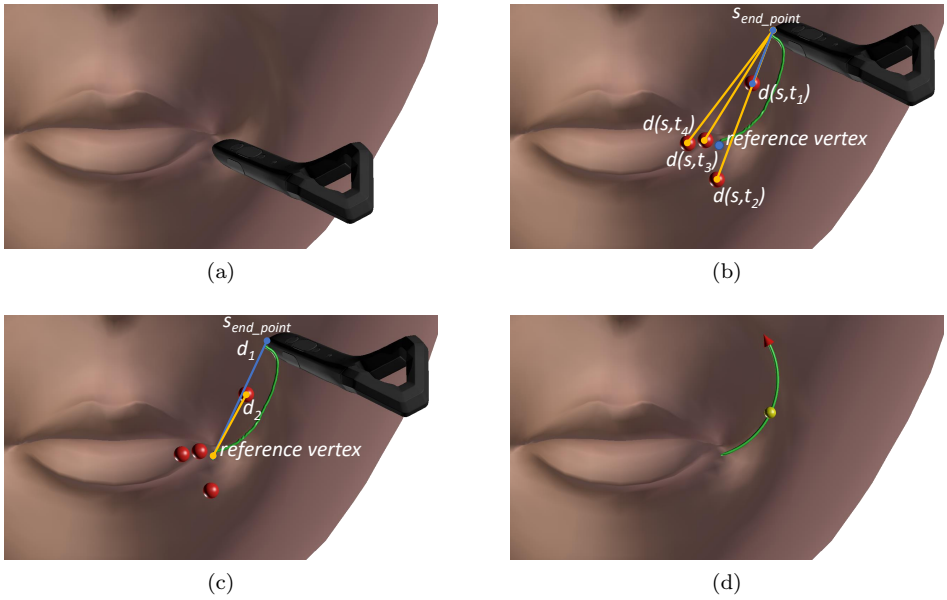


Figure 4. Drawing operations: a) the user first approaches the mesh with the VR Ink. While drawing several graphics elements appear, namely the green curve representing the stroke and the red spheres showing the target positions reached by the reference vertex. When the user terminates the stroke, the system b) computes the distances between the stoke ending point and the target positions in order to select the blendshape to activate, and c) determines its weight and the amount of FFD by comparing the stoke ending point to reference vertex and the stoke ending point to target position distances. Finally, d) a beautification step is applied that generates the 3D representation of the (beautified) stroke, and graphics elements (a red arrow and a yellow handle) for manipulating the stroke and controlling the animation are displayed.

situations, based on the visual feedback received, the user may choose a different reference vertex or avoid ending the stroke close to those target positions.

3.3. Computing weights and applying exaggerations

Once the target blendshape has been identified for the drawn stroke, the system has to determine the weight w to be applied to it. The weight is the parameter controlling the deformation, from none to the maximum one (hence, $w \in [0, 1]$). The user can decide the weight of the activated blendshape by choosing the ending point of the stroke: an ending point close to the reference vertex will set a low weight for the blendhshape, whereas an ending point close to the target position will set a high weight.

To this aim, two distances, referred to as d_1 and d_2, are computed. The first one (d_1) corresponds to the distance between the reference vertex and the ending point of the stroke, whereas the second one (d_2) corresponds to the distance between the reference vertex and the closest target position. Distances computed, e.g., for the target position t_1 are shown in Fig. 4c. Depending on the values of d_1 and d_2, the following cases are considered for determining w:

$$\begin{cases} d_1 < d_2 & w = d_1/d_2 \\ d_1 \geq d_2 & w = 1. \end{cases} \tag{2}$$

When $d_1 \geq d_2$, the higher the distance ratio, the higher the exaggeration that the

system applies to the facial expression by using a FFD transformation in order to make the reference vertex reach the target position. Deformation is also applied to the reference vertex neighborhood, by applying to close vertices transformations that are attenuated with a proportional editing function $S(j)$ defined as

$$\begin{cases} S(j) = 1 - \left(\frac{1}{n+1}\right)^{k+1} & k \geq 1 \\ S(j) = 1 - \left(\frac{1}{n+1}\right)^{-k} & \text{otherwise,} \end{cases} \tag{3}$$

where j is the vertex index (with $j \in [1, M]$), M is the number of vertices belonging to the neighborhood, n is the number of edges belonging to the shortest path that connects the j-th vertex and the reference vertex, and k is a scaling factor (with $k \in [-\infty, \infty]$) that can be configured to obtain rigid ($k < 0$) or elastic ($k > 0$) deformations (in the experiments, it was set to $k = 0$ for linear deformations).

3.4. Position interpolation

Once the target blendshape and the amount of deformation have been determined, the system computes the trajectory to be followed by the geometry for passing from the current to the target blendshape. As discussed, differently than in previous works where the users were allowed to animate only the weight of the blendshape (with the result that only linear translations in space could be obtained), with the proposed approach it is possible to control the trajectories of the transitions by means of the drawn strokes.

As suggested also by well-known principles regarding the use of arcs in animating living things (Thomas, Johnston, and Thomas (1995)), the possibility to leverage (blendshape-based, in the specific case) deformations encompassing both translations and rotations can be particularly important when animating specific parts of the face like, e.g., the motion of the jaw in a mouth opening animation or the motion of the eyelids in an eyes closing animation, since it allows animators to obtain greatest generality and fidelity of facial expressions compared to simplified, linear-only movements (X. Liu, Xia, Fan, and Wang (2011)). Fig. 5 compares the animation of the said parts using linear translations (red lines) and trajectories containing (also) rotations (blue lines). A video showing the two alternatives for the eyes closing animation is also available at `https://bit.ly/3Of3hcI`.

Due to the high sampling rate, strokes are usually characterized by noisy points. For this reason, before using a drawn stroke for defining a trajectory, a beautification step is applied. To this purpose, in order to support non-linear deformations, a circle that contains three of the stroke points (which are not aligned) and that best approximates it is computed. The three points that are considered are the first one, the last one, as well as the one that occupies the middle position in the array containing all the control points of the Bezier curve (points A, B and C in Fig. 6). The algorithm for finding the equation of the circle that goes through three points is then applied. From the equation, it is also possible to retrieve the center of the circle. Once the center of the circle has been determined, the system computes the arc that contains the three points A, B and, C by using the normal of plane Π_3 (i.e., the plane to which the three points belong to, as shown in Fig. 6) as rotation axis.

This arc indicates the trajectory to be followed by the reference vertex interpolating

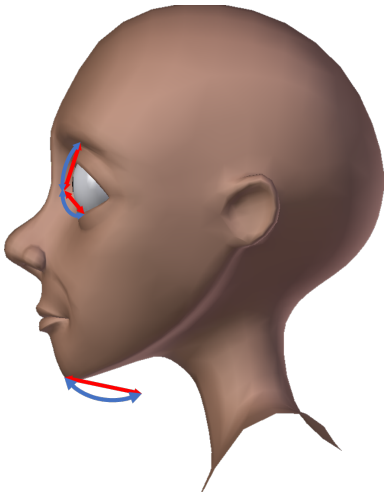


Figure 5. Examples of linear translations (red lines) and of trajectories containing rotations (blue lines) that can be used to control deformations in a mouth opening and eyes closing animation.

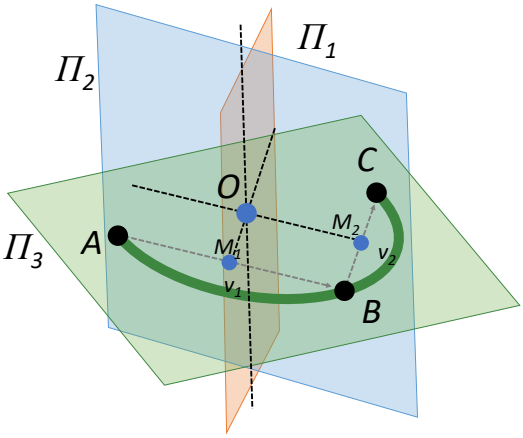


Figure 6. Computing the center of the circle representing the stroke.

between the current and the target blendshape. The effect of applying the beautification step can be observed by comparing the strokes in Fig. 4c and Fig. 4d.

For all the vertices affected by the target blendshape, a different center is computed by applying a translation equal to the distance between the reference vertex and the vertex v_j in the current blendshape. The use of a translated center, instead of the same center computed for the reference vertex, avoids the generation of spherical movements on entire parts of the mesh, which would lead to unpleasant effects.

After computing the individual centers and knowing the initial and end positions for each vertex affected by the target blendshape, it is possible to determine the rotation (θ_j) to be applied to the reference vertex neighborhood using the law of cosines (Fig. 7). Applying only the rotation would make a vertice v_j reach a position (indicated by v_{ij} in Fig. 7) that probably differs from the target one (v_{tj}), which is determined by activating the target blendshape and applying the FFD transformation described above. For this reason, an offset d_j, computed as the distance between the two positions v_{ij} and v_{tj}, is used for animating the trajectory. In particular, to obtain the in-between

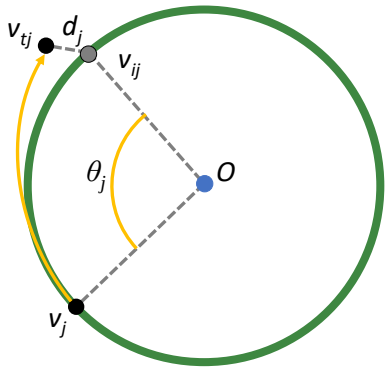


Figure 7. Computing the rotation to be applied to the reference vertex neighborhood.

positions of vertice v_j, a scaling factor $l_j \in [0, 1]$ is first computed by considering the ratio between the current frame and the final frame. This scaling factor is then applied to both the amount of rotation θ_j and the amount of translation d_j in order to define the trajectory between the two frames.

3.5. Time interpolation

As illustrated above, while a stroke is drawn the system records the timing of the stylus movements, which is used to define the time interpolation. In this way, the speed of the animation, as well as the ease-in/out (i.e., acceleration/deceleration at the beginning/end of the animation), can be controlled by the user in a natural way.

The time interval between the triggering of the drawing operation with the Trackpad button of the VR Ink and its release are used to establish the duration of the animation. Afterwards, the arc length of the curve is determined by approximating it to the sum of the distances of its control points. The ratio between the arc length computed for each vertex and the total length is mapped on the weight w that the blendshape has to assume at that point. In this way, the first point is assigned a weight $w = 0$, whereas the last point is assigned the weight that the blendshape has to assume for that stroke (e.g., $w = 1$ if the ending point of the stroke corresponds to the target position of a blendshape).

As shown in Fig. 8, slow drawings (the regions with a blue background in the left part of figure) result in short arc lengths (plot on top) that are mapped on close values of the blendshape weights (plot at the bottom), thus producing smooth variations of these values. Conversely, fast drawings (the regions with a yellow background, to the right) are mapped on longer arcs that produce steep variations in the weight values. For each sample, a keyframe is defined in the Blender Timeline to follow the timing of the user's gesture as close as possible. The number of keyframes can be configured by the user, setting a different sampling rate.

3.6. Handles manipulation and animation control

Once the strokes have been drawn and the animation has been created, the user can control both the position and time interpolation by manipulating the handles that appear on the strokes, i.e., the red arrow and the yellow sphere depicted in Fig. 4d.

The red arrow indicates the direction of the movement performed during the ani-

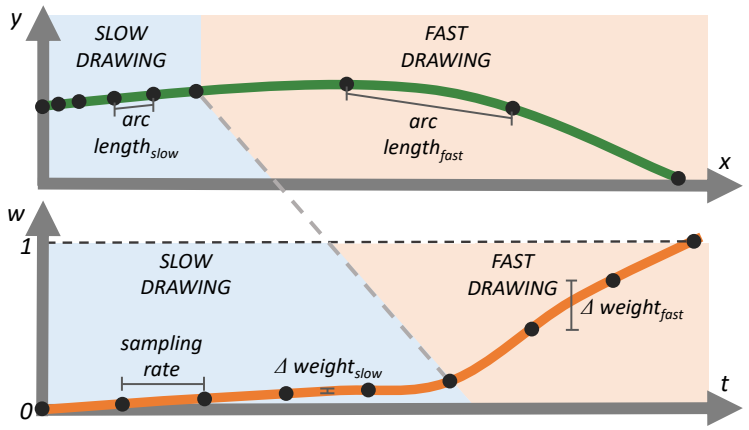


Figure 8. Time interpolation: 2D representation of the stroke (plot on top) and resulting interpolation curve (plot at the bottom).

mation. By using the Gripper button of the VR Ink, it is possible to grab/move this point into a new position, thus changing the ending position of the stroke. The same operations described in the previous subsections are repeated to suggest other possible blendshapes to be activated or alter the amount of the FFD transformation applied, thus increasing/decreasing the exaggeration.

The yellow sphere at the center of the stroke lets the user control the curvature. By using the Gripper button of the VR Ink, it is possible to change the position of this handle in the 3D space to obtain different degrees of curvature. The operations concerning position interpolation described above are repeated taking into account the new position of the middle point. In case the user moves this handle to a position where the starting, ending, and middle points are aligned, the generated transition between the two blendshapes does not consider any rotation, thus obtaining the same effect of animating only the weights of the blendshape without providing any trajectory to follow. When this happens, the color of the sphere changes from yellow to green.

To make it easier for the user to position the sphere handle on the same line of the starting and ending points, a snapping effect is applied based on the distance between the handle and the exact middle position. When the distance is under a configurable threshold, the handle is automatically snapped to the middle position. In the case of aligned points, the position of the vertices in the in-between frames is computed with linear interpolation.

Besides letting the user manipulate the position interpolation, the yellow sphere handle also allows him or her to control the time interpolation. As discussed, the timing of the animation is controlled by the speed of the drawing gesture. The curves that define the interpolation of the blendshape weights, however, can be controlled by means of this handle. In particular, the position of the sphere along the trajectory of the stroke can be changed by pressing the Trigger button of the VR Ink. The ratio (in the following referred to as α) computed as the length of the arc connecting the initial point to the sphere handle over the total arc length is used to move the keyframes along the timeline. When the user stops drawing, the sphere is automatically placed at the middle of the stroke ($\alpha = 0.5$). Using the Trigger button, the user can move it close to the starting ($\alpha = 0$) or ending point ($\alpha = 1$). Once handle manipulation has finished, the keyframes are mapped to new positions according to the function $y = x^p$, where y is the new position of the keyframe, x is the original position, and p

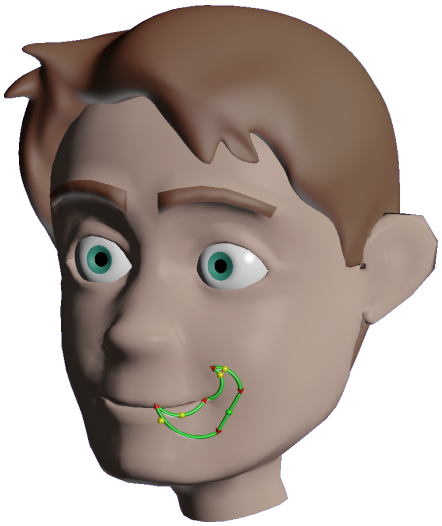


Figure 9. Result of concatenating multiple strokes.

a parameter defined as

$$p = \begin{cases} \alpha \left(\frac{St_{Max}-1}{-St_{Max}} \right) + St_{Max} & \alpha < 0.5 \\ 1 & \alpha = 0.5 \\ (\alpha - 0.5) \left(\frac{St_{min}-1}{1-0.5} \right) + 1 & \alpha > 0.5. \end{cases} \tag{4}$$

For instance, by using this definition, an acceleration of the initial keyframes and a deceleration of the last ones can be obtained by setting $\alpha < 0.5$. In Eq. 4, St_{Max} and St_{min} are configurable parameters that define a minimum and maximum value used to control the steepness of keyframes acceleration and deceleration, respectively.

3.7. Strokes concatenation

As illustrated in previous subsections, the system automatically converts the stroke drawn by the user into an arc or a line. In order to create more complex trajectories and make the face assume different expressions, it is possible to concatenate multiple strokes. To this aim, the user can start drawing a new stroke close to the ending point of the previous one. If the distance is below a configurable threshold, the two strokes are concatenated so that a modification applied to the first stroke influences also the second one, and vice-versa. To merge the strokes, the initial point used by the beautification algorithm is snapped to the ending point of the previous stroke, creating a single curve made up of two arcs. Fig. 9 shows the result of concatenating four strokes. In this example, the user created a cyclic animation in which the face assumes several expressions passing from the first blendshape to the next ones and returning to the initial configuration at the end.

Figure 10. Animations with symmetrical blendshapes.

3.8. Symmetries

Symmetries are clearly common in facial animation. For this reason, the system supports a symmetry modality that allows the user to easily obtain animations including this feature. When this modality is activated (by pressing the Trackpad Down button of the HTC Vive controller), the operations that involve symmetric blendshapes (in Blender named by convention with the suffix ".L" and ".R") are automatically mirrored. This operation applies to the drawing of new strokes, the manipulation of trajectories, the modification of the animation timing, and the concatenation of multiple strokes. For instance, if this modality is active and the user draws a stroke to move the right eyebrow down, the system automatically creates a new stroke influencing the left part of the face (as shown in Fig. 10). Disabling the symmetry modality, the animations acting on the left and right parts of the face can be individually controlled (as shown for the stroke manipulating the mouth corners in Fig. 10), e.g., to add small changes in the final animation.

3.9. GUI

In order to let the user interact with the system, a Graphics User Interface (GUI) has been integrated within the immersive environment. The GUI takes inspiration from a number of existing animation tools, both traditional and VR-based, that were analyzed during the design of the proposed system. The aim was to extract key aspects of such tools and to recreate them in VR, thus providing the animators with graphics elements they are already familiar with. The GUI elements are grouped into panels floating in the environment, as done in previous research works (Lamberti et al. (2020); Vogel et al. (2018)) and in tools like AnimVR or Facebook Quill.

The GUI includes two main elements: the timeline and the settings panels. The timeline panel (depicted in Fig. 11) lets the user visualize and manipulate the so-called "actions", i.e., the animations associated with a given stroke. On top of the timeline are displayed the current temporal window (i.e., the start and end frame of the system)

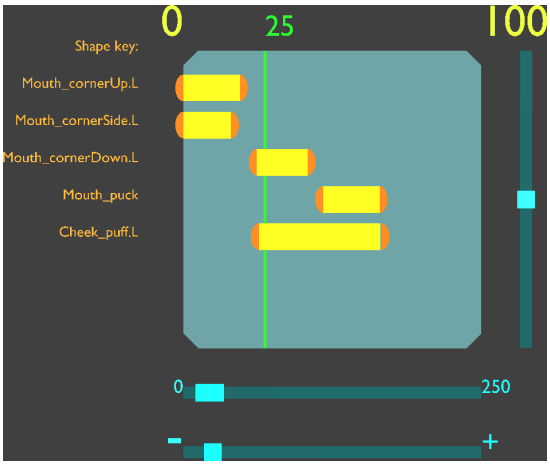


Figure 11. The GUI for the timeline.

available for recording an animation, as well as the current frame, indicated with a green font and a vertical bar. When the Timeline state is activated, the current frame is determined by the position of the user's finger on the Trackpad. Alternatively, the user can start the playback of the animation by pressing the Trackpad Up button. Actions are represented with yellow boxes with a horizontal size proportional to the length of the drawing gesture, i.e., to the duration of the animation. The colors and shapes of these elements are derived from Blender Timeline and Non-linear Animation Editor. The action name is provided on the left side of the timeline. In case of multiple actions, the user can scroll the list by using the vertical slider. The orange handles placed on the side of each box allow the user to move the first and the last frame of the action, rescaling and/or moving it in time using the same paradigm adopted in Tvori. It is possible to use the two horizontal sliders at the bottom of the timeline to manipulate the temporal window, i.e., resizing it or moving it to a new position, using the same approach adopted in Masterpiece.

The settings panel includes a number of sliders that can be used to configure the following system parameters:

- thickness of the stroke;
- minimum radius for choosing the target blendshape;
- minimum distance between the closest vertex of the mesh and the first point of the stroke to activate the blendshape selection;
- editing radius of the FFD (i.e., parameter n in Eq. 3);
- minimum distance between two strokes for concatenating them;
- sampling frequency.

This list of parameters was defined while designing the tool and firstly experimenting with it. Indeed, additional items could be added also thanks to the scripting capabilities of the underlying animation suite.

4. Experimental setup

In order to evaluate the effectiveness of the proposed system, as done, e.g., in Cetinaslan and Orvalho (2018) and Serra, Cetinaslan, Ravikumar, Orvalho, and Cosker

(2018), a user study was carried out by involving students at the authors' university with intermediate animation skills.

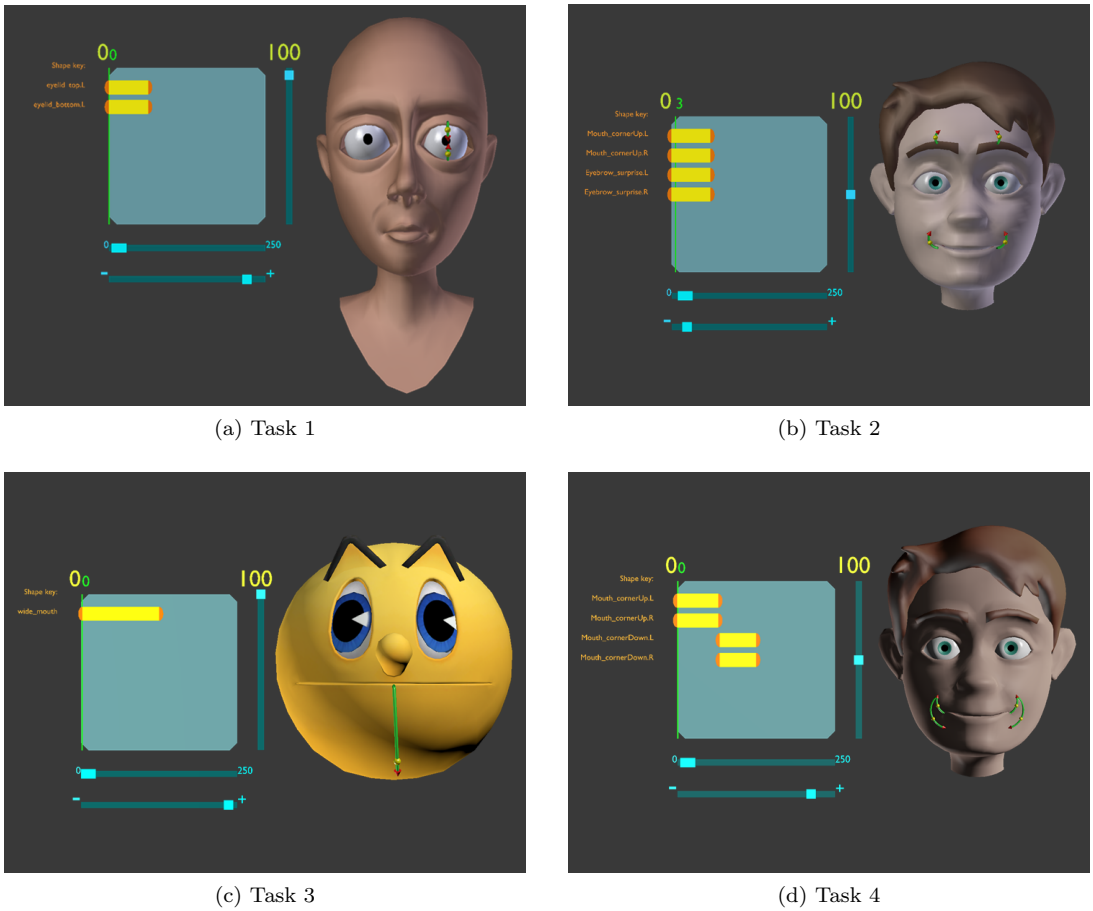
4.1. Participants

The 35 volunteers (24 males and 11 females) involved in the study were aged between 21 and 31 (M=25.51, SD=2.58). They can be considered as non-professional animators, even though they were rather skilled on Blender, since they had attended already at least one course on computer modeling and one course on computer animation with practical project works using Blender. Information collected with a demographic questionnaire at the beginning of the experiment revealed that only 20.00% of the participants used VR devices regularly, 14.29% used them sometimes, whereas 65.71% never used them. The sample size was determined with the aim of observing effect sizes of medium entity (Cohen's $d = 0.5$) with a study power $(1 - \beta_{err})$ of 80% for a non-parametric repeated measures test (Wilcoxon signed-rank test for paired samples).

4.2. Tasks

Each participant was requested to carry out four animation tasks using both the proposed VR interface (in the following referred to as VRI) and the native Blender interface (BNI); for the BNI, mouse and keyboard were used. The four tasks, exemplified in Fig. 12, were designed to make the participants perform specific operations that are typically needed in facial animation, with the aim to highlight both strengths and weaknesses of the proposed interface. The two interfaces were tested in a random order to limit as much as possible the learning effect. In all the tasks, the participants were requested to recreate a reference animation. The reference animation was shown on a mesh (representing the same face model) separated from the one to be animated by the participants by a small spatial offset. The operations to be performed in the four tasks are described below.

- *Task 1* (Fig. 12a): This task required the participants to recreate a simple animation of a left eye blinking. The reference animation was realized by manipulating two blendshapes. With the BNI, the participants had to identify the correct blendshapes from a list including 22 items with self-explaining names, and animate them inserting at least four keyframes on their weights. The Blender Timeline had to be used to define the timing of the animation. With the VRI, the participants had to draw two strokes and use the tools in the timeline panel to align the created animation with the reference.
- *Task 2* (Fig. 12b): This task focused on spatial interpolation. In particular, the participants had to make the character smile raising up the eyebrows. A nonlinear trajectory had to be defined for the mouth corners passing from the "base" to the smile blendshape. In Blender, the character was equipped with a facial rig including eight bones that allowed the participants to manipulate blendshapes by working with bones. Predefined Blender drivers were used to map the positions of the bones to the weights of blendshapes. With the BNI, in order to make the mouth follow a given trajectory, the participants had to manipulate 19 hook elements deforming the mesh through a lattice. To complete the task, the participants needed to animate the positions of the bones and of the hook elements by inserting at least 14 keyframes (four for the bones controlling the eyebrows, four for the mouth, and six to define the trajectories). With the VRI,



(a) Task 1

(b) Task 2

(c) Task 3

(d) Task 4

Figure 12. Tasks considered in the experiments.

the trajectories can be defined using strokes. In particular, the participants were requested to draw at least four strokes and manipulate the timeline to align the actions. The character used in this task had symmetrical blendshapes; hence, the participants had the possibility to reduce the number of operations to be performed by making use of Blender functionalities regarding symmetric copy & paste with the BNI or the symmetry modality with the VRI.

- *Task 3* (Fig. 12c): This task concerned time interpolation. The participants were asked to make the character open the mouth with a movement that starts slow and then speeds up in the last frames of the animation. When operating with the BNI, the participants had to insert two keyframes for animating the weight of a blendshape (among the 11 available in the library). Afterwards, the Blender F-curve Editor had to be used to adjust the interpolation curve in order to fit the timing of the reference animation. With the VRI, a draft movement of the mouth could be generated while drawing with the stylus, as the participants had the possibility to perform a gesture that imitates the acceleration of the reference animation. Then, the timing could be adjusted by using the timeline panel and/or the yellow sphere handle on the stroke.
- *Task 4* (Fig. 12d): In this task, the reference animation was composed of a sequence of three blendshapes (basic shape, smiling and sad face) and made use of the facial rig described for Task 2. With the BNI, the animation could be

recreated by setting at least six keyframes for animating the activation of the blendshapes, and eight keyframes for controlling the trajectory of the mouth using the hooks. With the VRI, the participants were requested to concatenate multiple strokes alternating the drawing with operations on the timeline panel in order to make the strokes for the sad face start after the end of the stroke created for the smiling face. Like for Task 2, symmetries could be used to reduce the number of strokes to be drawn.

Four videos showing a user performing each of the tasks are available at `https://bit.ly/3V4vp46`.

In order to foster replicability, in this work we used publicly available models, which can be downloaded at `https://bit.ly/3Ege959`. Although their complexity does not reach that of professional productions, it is comparable to that of recent works on animation interfaces (N. Liu et al. (2021), Cetinaslan and Orvalho (2020a), Serra et al. (2018)).

Before executing the four tasks, the participants were left free to familiarize with the VRI. In particular, a training scenario was devised to let the participants become confident with the functionalities of the system, trying to compensate for their previous experience with the BNI. During the training, the participants were also allowed to operate with the BNI to refresh their knowledge and try the specific functionalities required to complete the tasks, if needed. The training session lasted 15 minutes, on average.

Afterwards, the experiment was started. During the execution of the tasks, the participants were allowed to ask for help with both the BNI and the VRI. No time limit or accuracy threshold was set. That is, the participants were allowed to continue working on the animation until they were satisfied with the obtained result.

4.3. Evaluation criteria

The two interfaces were compared by making use of both objective and subjective measurements. The objective measurements included three metrics. The first metric considered the time needed by each participant to complete the task. The second metric considered spatial animation accuracy that, similarly to what done in Paravati, Lamberti, Gatteschi, Demartini, and Montuschi (2016), was obtained by computing the *frame-by-frame* difference between the user-generated and the reference animation. To make the two animations comparable on a frame-by-frame basis (i.e., to get rid of possible differences in duration and focus on spatial distances), the Dynamic Time Warping algorithm (Müller (2007)) was used, aligning the two sequences containing the list of values assumed by the weights in the two animations for all the blendshapes. Then, the metric was calculated as the sum of the differences between the weights contained in the aligned sequences for the two animations. The third metric considered temporal animation accuracy, defined as the *frame-to-frame* jitter of a given animation. Like in Niehorster, Li, and Lappe (2017), this metric was obtained by calculating the root mean square (RMS) of the sum of the differences between the weights at two consecutive frames for the whole animation.

The subjective measurements were collected at the end of the experiment by requesting each participant to fill in a post-test questionnaire (`https://bit.ly/3ECRYrq`). The questionnaire included two sections. The first section evaluated the usability of the two interfaces by means of the System Usability Scale (SUS) in Brooke (1996). The second section was aimed at assessing the participant's satisfaction by means of

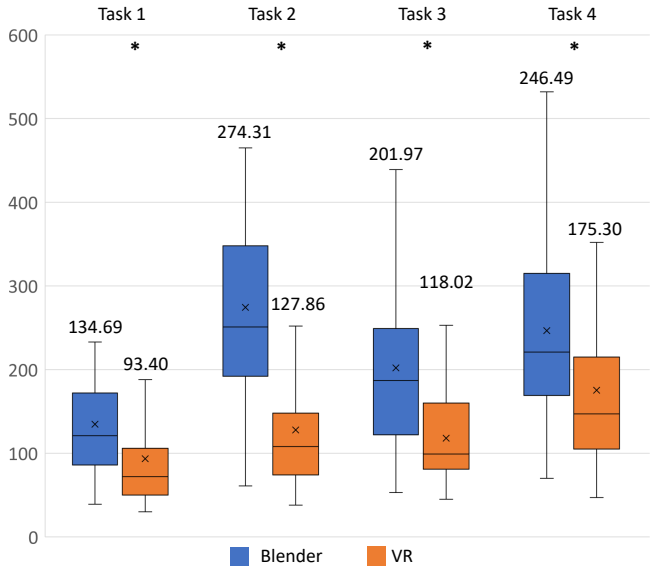


Figure 13. Completion time (in seconds) for the four tasks (the lower, the better). Mean values ('x' symbol), minimum and maximum values (whisker height), first and third quartiles (box size), and medians (horizontal line) are shown. The '*' symbol indicates statistically significant differences.

a questionnaire that was proposed in Pantuwong (2016) for evaluating 3D animation systems based on non-traditional interfaces. The participant's preferences for each task as well as for the overall experience were also collected.

5. Results

In the following, the results obtained by measuring the above metrics in the execution of the described tasks are presented and discussed, with the aim to compare the BNI and VRI performance.

5.1. Objective results

Measurements regarding the completion time, the spatial animation accuracy and the temporal animation accuracy are reported in Fig. 13, Fig. 14 and Fig. 15, respectively. Statistically significant results (verified using the mentioned test, $p < 0.05$) are marked with the '*' symbol.

Considering completion time, it can be noticed that the VRI allowed the participants to perform all the tasks faster than with the BNI (differences were significant for all the tasks). In particular, the speedup was 30.65% in Task 1 ($p = .004$, $d = 0.59$), 53.39% in Task 2 ($p < .001$, $d = 1.28$), 41.57% in Task 3 ($p < .001$, $d = 1.01$), and 28.88% in Task 4 ($p = .001$, $d = 0.69$), with a 39.99% time saving, on average. The possibility to control at the same time the trajectories to be followed and the timing of the animation made the participants be faster when using the VRI. This benefit is more evident in those tasks in which they were requested to animate more parts of the face (e.g., Task 2), as the above manipulations had to be repeated for each blendshape.

For what it concerns the spatial animation accuracy, differences were significant only for two tasks. In particular, the VRI let the participants be more accurate in recreating

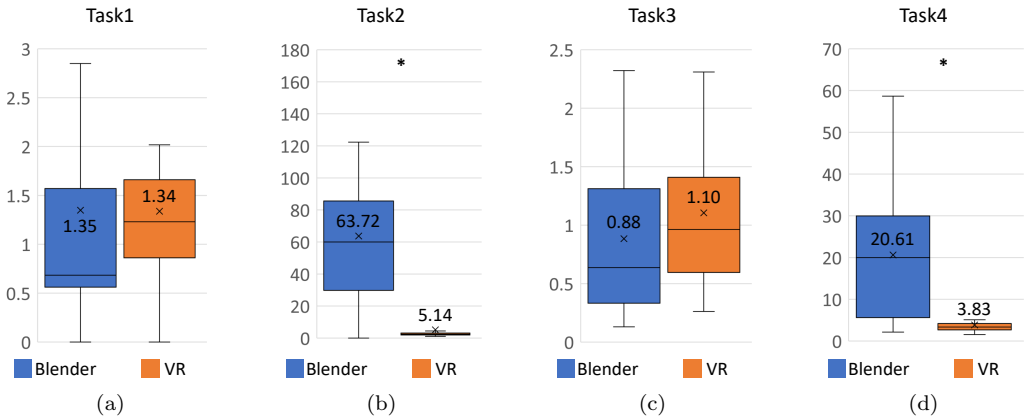


Figure 14. Spatial animation accuracy (dimensionless units) for the four tasks (the lower, the better). Mean values ('x' symbol), minimum and maximum values (whisker height), first and third quartiles (box size), and medians (horizontal line) are shown. The '*' symbol indicates statistically significant differences.

the reference animation than the BNI in Task 2 (91.93%, $p < .001$, $d = 1.46$) and Task 4 (81.37%, $p < .001$, $d = 1.60$). These results can be explained by the fact that, in Task 2 and Task 4, the trajectories to be followed by the mouth corners are longer than the movements of the eyes in Task 1 or of the mouth in Task 3. To obtain the desired trajectories, the participants found it easier and faster to combine the weights of the correct blendshape with those of other ones by using the facial rig, rather than correctly using the hook elements. These operations negatively impacted the spatial animation accuracy, since wrong blendshapes were activated influencing the distances computed after the DTW alignment.

The analysis of temporal accuracy could, in turn, provide insights about the timing of the created animations: in fact, the higher the RMS, the larger the variations in the weights of blendshapes between consecutive frames. Fig. 15 reports the absolute values of the differences between the RMS computed for the user-generated animations and the ideal value computed for the reference animation in the four tasks. Positive/Negative differences indicate that transitions between the starting and ending conditions were faster/slower in the user-generated animations than in the reference ones. Smaller differences indicate a better fit with the reference animation. As already observed for spatial accuracy, differences were significant for two tasks. In particular, the VRI let the participants be more accurate than with the BNI in Task 2 (46.81%, $p = .008$, $d = 1.22$) and Task 4 (84.65%, $p = .002$, $d = 1.98$). Similarly to what observed for spatial accuracy, this result may be related to the fact that the participants, who operated with the rig rather than with the hooks, activated also wrong blendshapes, thus increasing their frame-to-frame jitter.

5.2. Subjective results

In the first section of the questionnaire, the participants were requested to evaluate the usability of the two interfaces based on the SUS scale Brooke (1996). Questions were expressed as statements to be evaluated on a 1-to-5 Likert scale from strongly disagree to strongly agree. Based on collected results, the participants found the VRI (75.86) as characterized by a higher usability than the BNI (51.71, $p < .001$, $d = -1.80$). According to the categorization in Aaron, Philip, and James (2009), the score obtained

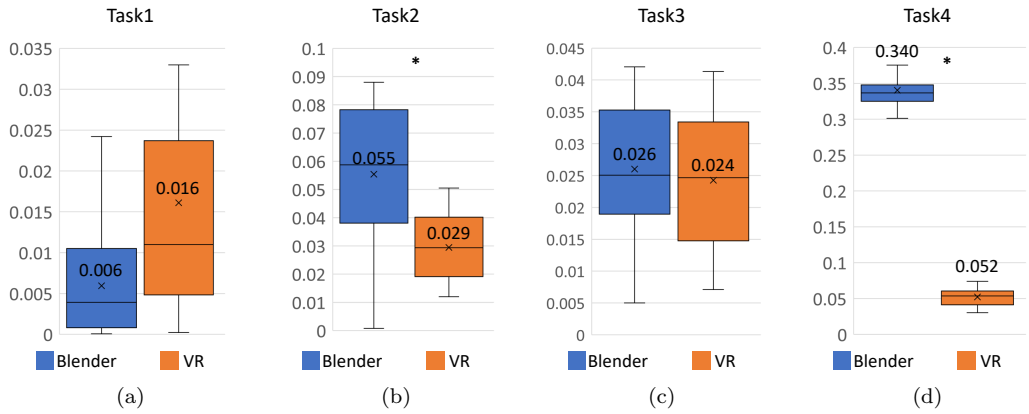


Figure 15. Temporal animation accuracy (dimensionless units) for the four tasks (the lower, the better). Mean values ('x' symbol), minimum and maximum values (whisker height), first and third quartiles (box size), and medians (horizontal line) are shown. The '*' symbol indicates statistically significant differences.

by the VRI corresponds to a B grade, associated to the "Good" class in the adjective rating scale, whereas the BNI was evaluated with a D grade, corresponding to the "Poor" class.

The reasons behind the higher appreciation for the VRI can be found by analyzing in detail the individual statements, which are reported in Table 1. The participants stated that they would be interested in using the VRI more frequently than the BNI (4.34 vs 3.03, $p < .001$, $d = -1.52$), and found the VRI as characterized by a lower complexity (1.86 vs 2.91, $p < .001$, $d = 1.03$) and easier to use (4.26 vs 3.14, $p < .001$, $d = -1.18$). Indeed, the flexibility of Blender, guaranteed by a large set of available functionalities, translates into a huge number of panels and widgets that the participants were requested to interact with to complete the task, resulting into a higher mental workload. Another interesting aspect regards the learnability of the proposed interface, as the participants felt that a lower amount of information had to be learnt to use the VRI than the BNI (2.23 vs 3.66, $p < .001$, $d = 1.41$), making them perceive that the VRI could be easily learnt by a higher number of people (4.06 vs 2.31, $p < .001$, $d = -1.97$). In order to answer this question, the participants may have compared their experience attending the 14-week long computer animation course in which they learned how to use the BNI for animating with the short training session required to get familiar with the VRI. Finally, the participants had the feeling that using the VRI was less cumbersome than interacting with the BNI (2.11 vs 3.09, $p < .001$, $d = 1.02$) and were more confident with operating in the virtual environment than with the Blender native functionalities, despite the fact they already had previous experience with this software (4.17 vs 3.49, $p < .001$, $d = -0.79$); with the VRI, the participants were also more confident that they could complete the task without the support of a technical person than with the BNI (2.26 vs 2.86, $p = .006$, $d = 0.52$). These results are linked to the intuitiveness of the VRI.

In the second section of the questionnaire, which was based on Pantuwong (2016), the participants were asked to rate a number of dimensions concerning their satisfaction with using the two interfaces. The evaluation was based on a 1-to-5 Likert scale, which was adapted from the original 0-to-10 scale to maintain coherency with the first section. The results are reported in Fig. 16. Collected feedback confirmed the outcomes of the first section, since the VRI was considered as easier to use (4.29

Table 1. Subjective results concerning usability based on SUSBrooke (1996). Cells highlighted with a bold font represent the best value between the two interfaces. The '*' symbol indicates statistically significant differences.

Statement	BNI	VRI	*p*	*d*
I think that I would like to use this system frequently*	3.03	**4.34**	< .001	-1.52
I found the system unnecessarily complex*	2.91	**1.86**	< .001	1.03
I thought the system was easy to use*	3.14	**4.26**	< .001	-1.18
I think that I would need the support of a technical person to be able to use this system	2.86	**2.26**	.006	0.52
I found the various functions in this system were well integrated	3.54	4.06	.110	–
I thought there was too much inconsistency in this system	2.37	2.03	.122	–
I would imagine that most people would learn to use this system very quickly*	2.31	**4.06**	< .001	-1.97
I found the system very cumbersome to use*	3.09	**2.11**	< .001	1.02
I felt very confident using the system*	3.49	**4.17**	< .001	-0.79
I needed to learn a lot of things before I could get going with this system*	3.66	**2.23**	< .001	1.41
SUS Score	51.57	**76.00**	< .001	-1.80
Grade	D	B		
Adjective rating	Poor	Good		

vs 3.06, $p < .001\, d = -1.42$) and to learn (4.14 vs 2.66, $p < .001\, d = -1.84$) than the BNI. Moreover, the findings regarding the reduced completion time when operating with the VRI that were observed with the objective measurements were also confirmed, since the participants perceived the interaction as faster than with the BNI (4.11 vs 2.91, $p < .001\, d = -1.16$). Besides confirming previous results, this section also showed that the participants expressed more appreciation for the VRI than for the BNI (4.09 vs 3.11, $p < .001\, d = -1.18$), with the former being capable of making them more satisfied (3.77 vs 2.80, $p < .001\, d = -1.09$) than the latter at the end of the experience. Another interesting aspect emerging from the analysis of this section was the ability of the VRI to stimulate more the creativity of the participants compared to the BNI (4.40 vs 2.74, $p < .001\, d = -2.04$), a result that may be related to the improved confidence that the participants felt when operating with that interface. Finally, a limitation of the VRI concerns its reduced flexibility with respect to the BNI (3.46 vs 4.17, $p = .006\, d = 0.82$). This result is not surprising, considering that the proposed interface offers just a subset of the many functionalities available in the underlying animation suite. Nevertheless, considering the scriptability of Blender components, new functionalities could be easily added to the current implementation in order to enhance its flexibility.

Finally, in the last section of the questionnaire, the participants were requested to express their preference for the VRI and the BNI when used for completing each task and overall. Average percentages are reported in Table 2. The majority of the participants preferred the VRI to the BNI, confirming the results observed with the objective and subjective evaluations.

6. Conclusions and future work

In this paper, an immersive facial animation system is proposed for the direct manipulation of blendshapes through a sketch-based interface. The system lets the users sketch strokes into an immersive environment. Strokes are used by the system to de-

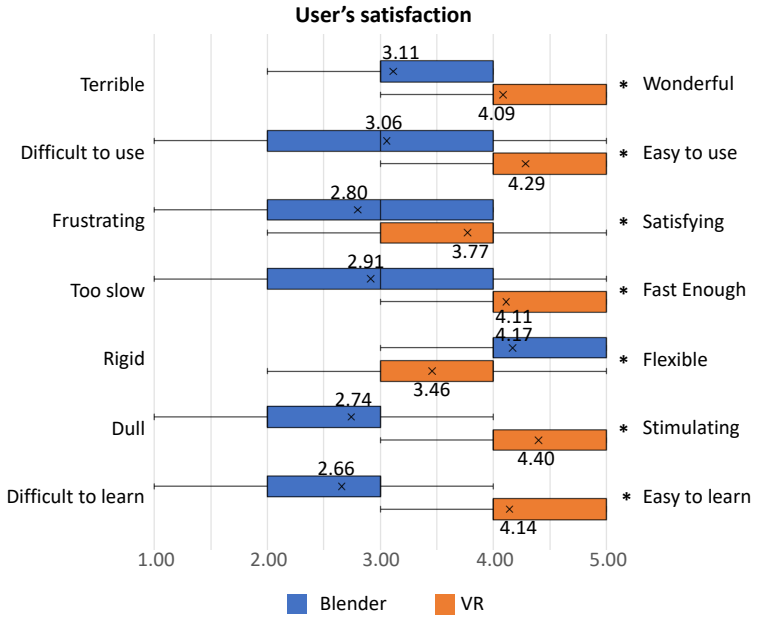


Figure 16. Subjective results based on the criteria defined in Pantuwong (2016) regarding users' satisfaction. Mean values ('x' symbol), minimum and maximum values (whisker width), first and third quartiles (box size), and medians (vertical line) are shown. The '*' symbol indicates statistically significant differences.

Table 2. Preferences expressed by the users for the four tasks and overall (percentages).

Task	BNI	VRI
Task 1	41.18%	**58.82%**
Task 2	17.65%	**82.35%**
Task 3	38.24%	**61.75%**
Task 4	23.53%	**76.47%**
Overall	26.47%	**73.53%**

fine the trajectories to be followed during the transition between facial poses and to control the timing of the animation in the in-between frames. Blendshapes are automatically activated by the system, depending on the drawn strokes. The users can also apply FFD for adding exaggerated characteristics, still making use of the sketch-based interaction paradigm. Finally, 3D handles and controls allows them to fine-tune the created animation directly in the immersive environment.

Experimental results showed the benefits brought by the application of this methodology in terms of completion time, animation accuracy, and users' perceived usability. The validity of results was assessed by means of a post-hoc power analysis. Considering the minimum effect size calculated on statistically significant differences (i.e., Cohen's $d = 0.52$), an actual study power equal to 0.83 was observed. Being larger than 80%, this value was considered as acceptable.

Future works will be devoted to address the limitations of the system that emerged during the experimental evaluation, taking into account improvements suggested by the participants involved in the user study. In particular, the participants lamented the lack of graphics controls (similar to those provided in the Blender F-curve Editor) to manipulate the time interpolation, as the yellow sphere handle moving along the stroke was considered as too limited. The participants also suggested to let the users define

more control points on the drawn stroke for manipulating the trajectories. They also recommended to introduce a mechanism for making it clearer which blendshape will be activated; they suggested, e.g., to show possible target positions when approaching the vertices of the mesh and not only when the user is drawing, or to use the so-called "ghost metaphor" for providing a (transparent) preview of the deformations applied by the blendshapes.

Besides solving the above limitations, further efforts are needed to improve the algorithm that selects the blendshape to be activated. At present, a stroke can control only the weight of one blendshape at a time. In the future, the methodology proposed in Lewis and Anjyo (2010) could be integrated into the proposed system, in order to make it possible to combine the weights of more blendshapes with a single stroke. Moreover, the algorithm could be extended to make it compare not only the ending point of the stroke with the target position of the reference vertex, but rather the whole path. In this way, animators might better select blendshapes that contain exaggerated versions of some micro-expressions far from the maximum level of deformation. The proposed methodology based on sketches drawn into an immersive environment could be also applied to different facial animation techniques, e.g., to physically based method to control the movements of the muscles involved in the simulation, or to manipulate the whole character pose.

Finally, experiments should be repeated by using more complex 3D models (with more blendshapes and more sophisticated rigs) and involving users with higher levels of expertise, e.g., professional animators, with the aim to evaluate the effectiveness of the interface in a wider range of scenarios.

Acknowledgement(s)

This work was mainly developed at the VR@POLITO, the Virtual Reality lab of Politecnico di Torino. The authors want to thank the Logitech Design Lab for providing them with a VR Ink Pilot Edition stylus.

Disclosure statement

The authors declares that they has no relevant or material financial interests that relate to the research described in this paper.

Funding

The research was supported by PON "Ricerca e Innovazione" 2014-2020 – DM 1062/2021 funds.

References

Aaron, B., Philip, K., & James, M. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, *4*(3), 114–123.

Anjyo, K., Todo, H., & Lewis, J. P. (2012). A practical approach to direct manipulation blendshapes. *Journal of Graphics Tools*, *16*(3), 160–176.

Bae, S.-H., Balakrishnan, R., & Singh, K. (2008). ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In *Proc. annual acm symposium on user interface software and technology* (pp. 151–160).

Barbieri, S., Garau, N., Hu, W., Xiao, Z., & Yang, X. (2016). Enhancing character posing by a sketch-based interaction. In *Proc. acm siggraph* (pp. 1–2).

Barr, A. H. (1984, jan). Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph.*, *18*(3), 21–30.

Beeler, T., Hahn, F., Bradley, D., Bickel, B., Beardsley, P., Gotsman, C., ... Gross, M. (2011). High-quality passive facial performance capture using anchor frames. In *Proc. acm siggraph* (pp. 1–10).

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Bernier, E., Chellali, R., Thouvenin, I. M., & Blom, K. (2012). The ICEA plug-in for virtual reality, immersive creation and edition of animation. In *Workshop on software engineering and architectures for realtime interactive systems* (pp. 36–42).

Berson, E., Soladie, C., Barrielle, V., & Stoiber, N. (2019). A robust interactive facial animation editing system. In *Proc. motion, interaction and games* (pp. 1–10).

Blanz, V., Basso, C., Poggio, T., & Vetter, T. (2003). Reanimating faces in images and video. *Computer Graphics Forum*, *22*(3), 641–650.

Brooke, J. (1996). SUS - A quick and dirty usability scale. *Usability Evaluation in Industry*, *189*(194), 4–7.

Cannavò, A., Calandra, D., Kehoe, A., & Lamberti, F. (2020). Evaluating consumer interaction interfaces for 3D sketching in virtual reality. In *Proc. eai international conference: Artsit, interactivity & game creation* (pp. 291–306).

Cannavò, A., Zhang, C., Wang, W., & Lamberti, F. (2020). Posing 3D characters in virtual reality through in-the-air sketches. In *Proc. international conference on computer animation and social agents* (pp. 51–61).

Cannavò, A., Demartini, C., Morra, L., & Lamberti, F. (2019). Immersive virtual reality-based interfaces for character animation. *IEEE Access*, *7*, 125463-125480.

Cetinaslan, O., & Orvalho, V. (2018). Direct manipulation of blendshapes using a sketch-based interface. In *Proc. international acm conference on 3d web technology* (pp. 1–10).

Cetinaslan, O., & Orvalho, V. (2020a). Sketching manipulators for localized blendshape editing. *Graphical Models*, *108*, 101059.

Cetinaslan, O., & Orvalho, V. (2020b). Stabilized blendshape editing using localized Jacobian transpose descent. *Graphical Models*, *112*, 101091.

Cetinaslan, O., Orvalho, V., & Lewis, J. (2015). Sketch-based controllers for blendshape facial animation. In *Proc. eurographics* (pp. 25–28).

Chang, E., & Jenkins, O. C. (2006). Sketching articulation and pose for facial animation. In *Proc. acm siggraph/eurographics symposium on computer animation* (pp. 145–161).

Choi, B., i Ribera, R. B., Lewis, J. P., Seol, Y., Hong, S., Eom, H., ... Noh, J. (2016). SketchiMo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics*, *35*(4), 1–12.

Ciccone, L., Öztireli, C., & Sumner, R. W. (2019). Tangent-space optimization for interactive animation control. *ACM Transactions on Graphics*, *38*(4), 1–10.

Flueckiger, B. (2011). Computer-generated characters in Avatar and Benjamin Button. *Digitalitat und Kino*, *1*.

Gipson, J., Brown, L., Robbins, E., Gomez, J., Anderson, M., Velasquez, J., ... Cooper, D. (2018). VR story production on Disney animation's "cycles". In *Proc. acm siggraph* (pp. 1–2).

Gouvatsos, A., Xiao, Z., Marsden, N., & Zhang, J. J. (2017). Posing 3D models from drawings. *Computers in Entertainment*, *15*(2), 1–14.

Guay, M., Cani, M.-P., & Ronfard, R. (2013). The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics*, *32*(6), 1–8.

Gunnarsson, Ö., & Maddock, S. C. (2010). Sketch-based posing of 3D faces for facial animation.

In *Proc. theory and practice of computer graphics* (pp. 223–230).

Hahn, F., Mutzel, F., Coros, S., Thomaszewski, B., Nitti, M., Gross, M., & Sumner, R. W. (2015). Sketch abstractions for character posing. In *Proc. acm siggraph/eurographics symposium on computer animation* (pp. 185–191).

Henrikson, R., Araujo, B., Chevalier, F., Singh, K., & Balakrishnan, R. (2016). Multi-device storyboards for cinematic narratives in VR. In *Proc. annual symposium on user interface software and technology* (pp. 787–796).

Igarashi, T., Matsuoka, S., & Tanaka, H. (2006). Teddy: A sketching interface for 3D freeform design. In *Proc. acm siggraph* (pp. 11–es).

Jackson, B., & Keefe, D. F. (2016). Lift-off: Using reference imagery and freehand sketching to create 3D models in VR. *IEEE Transactions on Visualization and Computer Graphics*, *22*(4), 1442–1451.

Jiang, Y., Zhang, C., Fu, H., Cannavò, A., Lamberti, F., Lau, H. Y., & Wang, W. (2021). Handpainter - 3D sketching in VR with hand-based physical proxy. In *Proc. chi conference on human factors in computing systems* (pp. 1–13).

Karras, T., Aila, T., Laine, S., Herva, A., & Lehtinen, J. (2017). Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Transactions on Graphics*, *36*(4), 1–12.

Kim, B. (2019). Virtual reality for 3D modeling. *Beyond Reality: Augmented, Virtual, and Mixed Reality in the Library*, *1*(1), 31–46.

Kim, J., & Singh, K. (2021). Optimizing ui layouts for deformable face-rig manipulation. *ACM Transactions on Graphics*, *40*(4), 1–12.

Lamberti, F., Cannavo, A., & Montuschi, P. (2020). Is immersive virtual reality the ultimate interface for 3D animators? *Computer*, *53*(4), 36–45.

Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F. H., & Deng, Z. (2014). Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, *1*(8), 2.

Lewis, J. P., & Anjyo, K.-i. (2010). Direct manipulation blendshapes. *IEEE Computer Graphics and Applications*, *30*(4), 42–50.

Li, C., Pan, H., Liu, Y., Tong, X., Sheffer, A., & Wang, W. (2018). Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Transactions on Graphics*, *37*(6), 1–12.

Liu, N., Zhang, D., Ru, X., Zhao, H., Wang, X., & Wu, Z. (2021). Caricature expression extrapolation based on Kendall shape space theory. *IEEE Computer Graphics and Applications*, *41*(3), 71–84.

Liu, X., Xia, S., Fan, Y., & Wang, Z. (2011). Exploring non-linear relationship of blendshape facial animation. *Computer Graphics Forum*, *30*(6), 1655–1666.

Mao, C., Qin, S. F., & Wright, D. (2007). Sketch-based virtual human modelling and animation. In *Proc. international symposium on smart graphics* (pp. 220–223).

Miranda, J. C., Alvarez, X., Orvalho, J., Gutierrez, D., Sousa, A. A., & Orvalho, V. (2012). Sketch express: A sketching interface for facial animation. *Computers & Graphics*, *36*(6), 585–595.

Müller, M. (2007). Dynamic time warping. *Information Retrieval for Music and Motion*, 69–84.

Niehorster, D. C., Li, L., & Lappe, M. (2017). The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, *8*(3), 2041669517708205.

Osawa, N., & Asai, K. (2003). An immersive path editor for keyframe animation using hand direct manipulation and 3D gearbox widgets. In *Proc. international conference on information visualization* (pp. 524–529).

Pantuwong, N. (2016). A tangible interface for 3D character animation using augmented reality technology. In *Proc. international conference on information technology and electrical engineering* (pp. 1–6).

Paravati, G., Lamberti, F., Gatteschi, V., Demartini, C., & Montuschi, P. (2016). Point cloud-based automatic assessment of 3D computer animation courseworks. *IEEE Transactions*

on Learning Technologies, *10*(4), 532–543.
Parke, F. I. (1974). *A parametric model for human faces.* The University of Utah.
Sagar, M. (2006). Facial performance capture and expressive translation for King Kong. In *Proc. acm siggraph* (pp. 26–es).
Serra, J., Cetinaslan, O., Ravikumar, S., Orvalho, V., & Cosker, D. (2018). Easy generation of facial animation using motion graphs. *Computer Graphics Forum*, *37*(1), 97–111.
Sifakis, E., Neverov, I., & Fedkiw, R. (2005). Automatic determination of facial muscle activations from sparse motion capture marker data. In *Proc. acm siggraph* (pp. 417–425).
Singer, G. (2003). The two towers: Face to face with Gollum. *Animation World Network*, *1*(2).
Thomas, F., Johnston, O., & Thomas, F. (1995). *The illusion of life: Disney animation.* Hyperion New York.
Thorne, M., Burke, D., & van de Panne, M. (2004). Motion doodles: An interface for sketching character motion. *ACM Transactions on Graphics*, *23*(3), 424–431.
Vogel, D., Lubos, P., & Steinicke, F. (2018). AnimationVR-interactive controller-based animating in virtual reality. In *Proc. workshop on animation in virtual and augmented environments* (pp. 1–6).
Zhang, L., Snavely, N., Curless, B., & Seitz, S. M. (2004). Spacetime faces: high resolution capture for modeling and animation. In *Acm siggraph 2004 papers* (pp. 548–558).
Zou, Q., Bai, H., Gao, L., Fowler, A., & Billinghurst, M. (2022). Asymmetric interfaces with stylus and gesture for VR sketching. In *Proc. ieee conference on virtual reality and 3d user interfaces abstracts and workshops* (pp. 968–969).

Appendix A. Glossary

A glossary of terms is provided in Table A1.

Table A1. Glossary of terms.

Term	Meaning
Armature (Rig)	An "armature" or "rig" is a kinematic chain used in the context of computer animation to deform the mesh of virtual characters. The chain is composed by individual elements named "bones", which can be articulated by the animator. The process of creating the chain is called "rigging", whereas the process of articulating it is called "posing".
Bezier curve	A Bézier curve is a parametric curve used in computer graphics and computer animation. It is defined by a set of control points (three for a cubic curve). The first and last control points are the endpoints of the curve, whereas the other ones interpolate it.
Blendshape	A blendshape is a parameterized deformation of a mesh that is often used in facial animation. Deformation can be expressed, e.g., in a range from 0 to 1, with 0 representing the "basis" mesh (no deformation), 1 the maximum deformation. Multiple blendshapes can be blended together by linearly combining their weights in order to obtain the intended deformation.
Bone	In computer animation, a "bone" is part of a hierarchical set of interconnected elements that composes an armature. Each bone is associated with a group of mesh vertices through a process named "skinning". Transformations applied to the bones are used to compute a weighted deformation of the associated vertices.
Direct-Manipulation Blendshapes (DMB)	It is a facial animation method introduced in Lewis and Anjyo (2010) that allows animators to control the weights of blendshapes by means of an interface based on few degrees of freedom.
Free-form deformation (FFD)	It is a geometric technique originally defined in Barr (1984) that can be used to model deformations of rigid objects as a set of hierarchical transformations.
Mesh	In computer graphics and computer animation, it is a collection of vertices, edges, and faces that can be used to represent the surface of an object.
Proportional editing	In computer graphics, proportional editing is a technique used to transform selected elements (e.g., vertices) while also affecting the nearby unselected elements. The term proportional indicates that the closer an unselected element is, the more it will be affected and vice-versa. Various types of functions can be used to control the deformation: constant, linear, Gaussian, etc.