Double Doodles: Sketching Animation in Immersive Environment With 3+6 DOFs Motion Gestures

Ruizhao Chen
Shanghai JiaoTong University
Shanghai, China

Ye Pan*
Shanghai JiaoTong University
Shanghai, China

Zhigang Deng
University of Houston
Houston, Texas, USA

Lili Wang
Beihang University
Beijing, China

Lizhuang Ma
Shanghai JiaoTong University
Shanghai, China

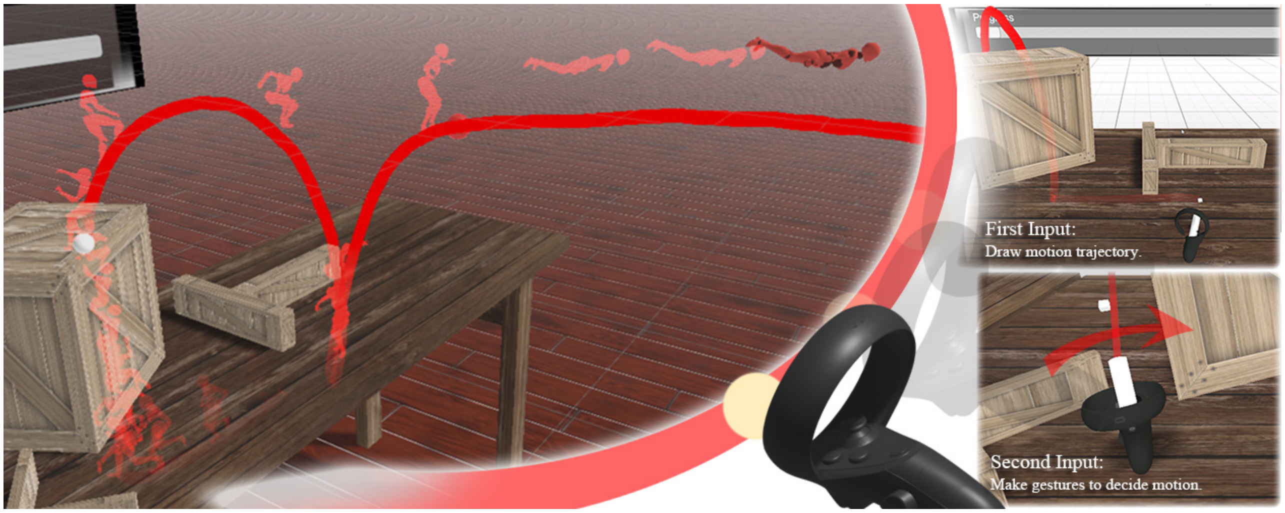


Figure 1: Double Doodles: A novel 3+6 Degrees of Freedom (DOFs) two-staged inputs method, making full use of two asynchronous inputs, for easily drafting expressive character animation in an immersive virtual environment.

ABSTRACT

We present "Double Doodles" to make full use of two sequential inputs of a VR controller with a total of 9 DOFs, 3 DOFs from the first input sequence for the generation of motion paths and 6 DOFs from the second input sequence for motion gestures. When engineering our system, we take ergonomics into consideration and design a set of user-defined motion gestures to describe character motions. We employ a real-time deep learning-based approach for highly accurate motion gesture classification. We then integrate our approach into a prototype system, and it enables users to directly create character animations in VR environments using motion gestures with a VR controller, followed by animation preview and animation interactive editing. Finally, we evaluate the feasibility and effectiveness of our system through a user study, demonstrating the usefulness of our system for visual storytelling dedicated to amateurs, as well as for providing fast drafting tools for artists.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; • **Human-centered computing** → **Virtual reality**.

KEYWORDS

character animation, 3D sketching, computer puppetry, interaction techniques, gesture classification, virtual reality.

ACM Reference Format:
Ruizhao Chen, Ye Pan, Zhigang Deng, Lili Wang, and Lizhuang Ma. 2023. Double Doodles: Sketching Animation in Immersive Environment With 3+6 DOFs Motion Gestures. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3613783

1 INTRODUCTION

Virtual reality (VR) devices enable new interaction paradigms for character animation with immersive, interactive, and imaginative features. They allow artists to navigate and interact with a 3D

[*] Corresponding author. whitneypanye@sjtu.edu.cn

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613783

immersive environment, and animate characters more naturally, compared to 2D interface [20]. Existing methods succinctly inherit the idea of puppet animation, determining the motion and position of the character through gestures of different objects. The available degrees of freedom (DOFs) are limited to 6 or less. However, there is a gap between the number of possible character animations and the number of possible gestures created through a VR controller which is a rigid object with few DOFs.

Several tools have been developed to assist casual users in designing 3D trajectories and character motions, using AR-enabled mobile phones. Examples include PuppetPhone [2], AR-Pose [25], and ARAnimator [27]. However, these workflows, by displaying animation results on 2D screens, cannot provide animators with a fully immersive 3D experience [19, 20]. Existing VR-based methods for creating animated characters in virtual environments, such as Motion Doodles [22] and SMD [7], use the VR controller to record the trajectories of its 6-DOFs over time. These trajectories are then parsed into a sequence of actions that are used to animate articulated characters.

Our objective is to assist users in creating more life-like and intricate character animations, specifically those with more degrees of freedom (DOFs), within immersive virtual environments. Using a smartphone or conventional VR controller as an inextensible rigid body as the input device only allows a maximum of 6 DOFs if a single input is used. It is well known that 6 DOFs are often insufficient to specify 3D character animations with rich details. We propose "Double Doodles", a novel 3+6 DOFs two-stage-inputs approach that employs two asynchronous inputs to efficiently produce natural and expressive character animations in immersive virtual environments. Through semantic decomposition of character animation specification into two steps, this work allows users to specify total 9 DOFs to achieve substantially higher realism and detail-richness of resulting character animations in VR. Addtionally, the core idea of our approach is stepwise planning to reduce the cognitive load, allowing users to plan the motion trajectory of the animation first and then consider how to move along the trajectory. Our main contributions can be summarized as follows:

- Proposal of VR-specific interaction rules to develop an in-situ, 3+6 DOFs, two-stage-inputs method to produce realistic and dynamic character animations interactively in immersive VR environments.
- Introduction of metaphors to expertly process dynamic user inputs and extension of deep learning algorithms to achieve accurate classification of motion gestures.
- Development of a Unity engine prototype system to run on off-the-shelf VR systems, and evaluation of its feasibility and effectiveness through meticulously conducted user studies.

2 RELATED WORK

2.1 Character Animation in AR/VR

By placing and controlling an avatar object of a real 3D character model, computer puppetry [21] techniques allow users to define character animations with intuitive user interfaces. Combined with AR/VR technologies, users can directly see the generated animation from different views in real-time [8]. The aforementioned traits make for a more user-friendly and interactive approach to produce character animation in AR/VR.

Presently, professional animation software implementing AR/VR puppetry functions is still developing better interaction techniques, mainly due to the intricate 2D interface, which requires a higher degree of expertise to manipulate, or complex configuration steps before producing any usable results [13, 26]. Thus, interests in developing simple and intuitive AR/VR user interfaces for animation creation have grown significantly [8, 24].

Motion Doodles (MD) is a design that guides action synthesis using 2D curves. Garcia et al. [7] expanded on this with their Spatial Motion Doodles (SMD) system for VR platforms, which incorporates 3-DOFs and expands the concepts into 3D space. However, the SMD method uses the VR controller as a metaphor for the character, which means the controller's movements directly impact the character. Thus, users must hold the controller like a puppet, rather than in their hand, as originally intended.

Several other works along this line have been completed in AR environment. Recently, the ARAnimator [27] was introduced to generate character animation on mobile AR devices in casual settings. This method increases the DOFs of the controller to six, which allows users to create more possible motions for their characters.

These methods described above are trying to solve the same issue known as the "DOF gap", which refers to the difference in DOFs between a 3D character and the control device. The device is typically composed of multiple rigid bodies with no more than 6 DOFs. It is challenging to simultaneously convey or specify two critical elements (gestures and paths) of a 3D character using a 6-DOFs controller only.

In this work we aim to increase the number of DOFs available on existing VR controllers for character animation generation. Our core idea is to decompose motions into motion gestures and motion paths, and then record them separately with two-staged inputs. We summarize the main feature differences between our work and some previous related systems in Table 1.

Table 1: Our work VS. previous systems

Name	MD	SMD	PP	ARA	**Ours**
Environment	VR	VR	AR	AR	**VR**
DOF	2	3	3	6	**3+6**
Gesture design	experts	experts	experts	users	**experts/users**
Classification	Regexp	Regexp	FSM	SVM	**NN**
Devices	PC	HTC Vive	Mobile	Mobile	**Oculus Rift S**

* MD: Motion Doodles[22], SMD: Spatial Motion Doodles[7], PP: PuppetPhone[2], ARA: ARanimator[27], Regexp: Bayesian classification, FSM: Fixed state machine, SVM: Support vector machine, NN: Neural network.

2.2 VR Ergonomics in Computer Puppetry

VR technology provide thrilling and new ways for entertainment and consumption of information. It offers exciting possibilities such as practical, time-saving animation production, a rich narrative device for conventional film and dynamic storytelling medium. Virtual work environments can liberate users from physical world constraints, potentially altering how they perform professional activities [10]. The immersive feature of virtual reality also increases users' learning abilities [16, 17] to help them.

Due to the limitation of the human physiological structure and the device's gripping manner, it is practically impossible to develop movements that accurately mirror the character's dynamics. Liang et al. [15] and Ye et al. [27] conducted studies on suitable gestures for manipulating 3D characters. These works aim to maintain a coherent mapping of motion gestures with character animation. While it is required to evaluate the feasibility of actions in real-world applications, it is also necessary to take into account the ease with which actions can be recognized. Thus, our work incorporates these two facets.

It is noteworthy that those interaction concepts designed for AR-enabled mobile phones based methods [2, 27] are difficult to directly translate or extend to VR-based methods, since their original metaphor is inextricably linked to a mobile phone with a flat rectangular shape and a camera. For example, it is feasible to imagine the two bottom corners of the mobile phone as the character's feet [27]. However, this may not be valid for a cylindrical VR controller. Standard VR controllers (e.g., HTC VIVE, Oculus, Valve Index) have cylindrical bodies. Comparing to ARAnimator[27], the VR input devices are difficult to be gripped on both edges like a mobile phone (a flat rectangular shape), thus the two bottom corners cannot be metaphorized as the character's feet. To utilize the original ergonomic design of VR itself, we want to keep a doodling and painting experience in our interaction design [22].

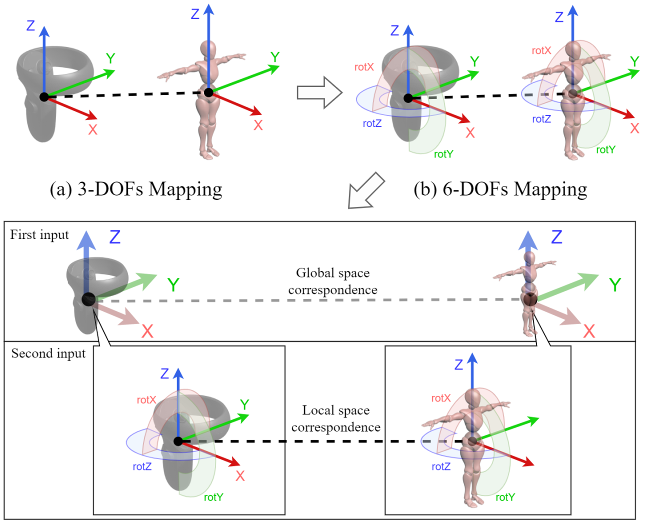
3 SYSTEM DESIGN AND DEVELOPMENT

3.1 VR Interaction Design: Two-staged Inputs

Various efforts have been attempted to tackle the well-known "DOF gap" issue. The early 2-DOFs MD method [22] and the subsequent 3-DOFs SMD approach [7] have an insufficient number of DOFs, limiting the number of actions that can be performed. Although the ARanimator approach [27] employs a 6-DOFs puppetry metaphor, which significantly enhances its expressive capabilities, it still suffers from trail distortions and inaccuracies in motion recognition. Distortions happen when the generated animations do not follow the input trail precisely.

Inspired by the above works, we specify the object's global XYZ position displacement information in the first input while specifying its local motion dynamics of the character during the second input. Figure 2 illustrates the decomposition of DOFs control in our method. Our method decouples global displacement information from local motion dynamics, enabling the expression of posture changes in all six DOFs of the character during movement. Using two consecutive inputs of a VR controller, this method provides a fixed-gripping approach and simplifies movement logic.

3.1.1 The First Input. The first input is used to determine the virtual character's 3D trajectory in a VR environment. The user holds the controller like a brush and draws trajectories in the air. A red curve will be used to indicate the plotted trajectory. Its positions are sampled at the "brush tip" (refer to Figure 3). Specifically, the spatial location information is determined using a white cube attached to the head of the VR controller held by the user (illustrated in Figure 3). We collect input data with 3 DOFs and a sampling rate of 60Hz and then use these information to create a motion trajectory in 3D space.

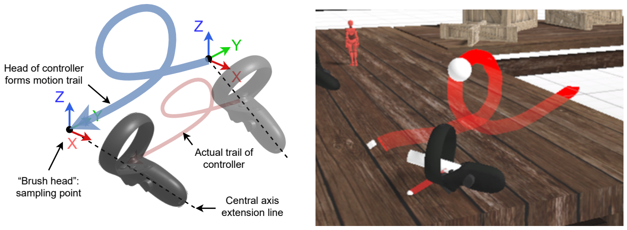


(a) 3-DOFs Mapping

(b) 6-DOFs Mapping

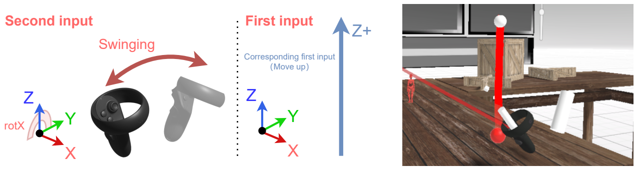
(c) the 3+6 DOFs mapping of our method via double inputs

Figure 2: The decomposition of the DOF control. We use the XYZ coordinates in the first input to create 3D trajectory. The second input, which contains XYZ coordinates and rotation information, will then determine the motion gesture.


Head of controller forms motion trail
Z
Y
X
"Brush head": sampling point
Actual trail of controller
Central axis extension line

(a) illustration of the trajectory **(b) corresponding VR scene**

Figure 3: Illustration of the first input. The VR controller is regarded as a long-handled brush. Users can draw a 3D trajectory for the character in the mid-air.



(a) the first and second inputs **(b) corresponding VR scene**

Figure 4: Illustration of the second input ("climbing" motion as an example). The character's motion type is controlled using all 6 DOFs, including XYZ coordinates and rotation information.

3.1.2 The Second Input. The second input serves to inform the generation of local motion dynamics of the 3D character. This input is processed at a rate that is reduced in comparison to the first input (e.g., 0.5 times normal speed), although the sample rate can be adjusted. The reduction in rate is intended to allow the animator sufficient time to design the animation effectively. This visualization interface allows the user to employ pre-designed gestures, described in Section 3.2, to specify local motion dynamics of the character (as shown in Figure 4). After the completion of sampling, the second input will be realigned, ensuring one-to-one temporal correspondence with the first input data and the generation of 6-DOFs data, which directs pose synthesis.

3.2 Gesture Design

According to the summary of the actions required for various 3D character animation productions in [27], 29 frequent actions can be roughly classified into five groups, depending on the situational information required. This study also identified those types of character motions that are commonly desired by novice or casual users when creating character animations, by identifying commonly used motions in user-created character animations.

Additionally, researchers have conducted research on ergonomics using hand-held devices. Dong et al. [5] classify frequently used motion gestures for hand-held devices. Based on the analysis conducted by Benzina et al. [4], for 3D positionable input devices with 2D touch inputs (e.g., buttons on the controllers), 4 DOFs manipulation metaphors centered on the roll and pitch rotation axes perform the best in terms of input efficiency and accuracy. Given that virtual reality controllers share these features, and in light of the preceding discovery, the action design in our system should be as near to this metaphor as possible.

Based on previous studies, we select 13 most commonly used actions to design our input gestures, described in Table 2. We also show some example gestures and their corresponding actions in Figure 5. The first input specifies the trajectory of an action type, and the second input specifies the character's local motion dynamics. For instance, when the character's state is required to be "lie down," the VR controller will be horizontal. For some actions such as "climb," "crawl," and "roll," we choose their most representative characteristic as the indicator of their second input. Using the "climb" motion as an example, if we use real-world experience as the reference, the most representative characteristic of the climbing movement is that two arms travel up and down sequentially, creating a repeated "left-right-left" pattern throughout the movement. Based on this observed pattern, the gesture (in the second input) for the climb motion needs to swing from side to side to imitate the movement pattern.

3.3 Gesture Recognition

Accurate and efficient gesture classification is essential for any successful gesture-based interaction system. Spatial Motion Doodles (SMD) [7] and ARAnimator [27] rely on manually-designed feature extraction techniques that aim to match specific scenarios. The incorporation of prior knowledge is advantageous in cases where the conditions of interaction are well-defined. However, tailor-made categories resulting from such an approach may suffer from over-specialization and lack the flexibility necessary for accommodating the full range of possible inputs. Furthermore, creating new categories is a non-trivial task that is contingent upon the developers' extensive experience. In situations where unintentional inputs are frequent, such as those encountered during natural interactions involving human users, manually-designed classification schemes can pose significant challenges. Thus, we opt to utilize neural networks in our gesture recognition approach.

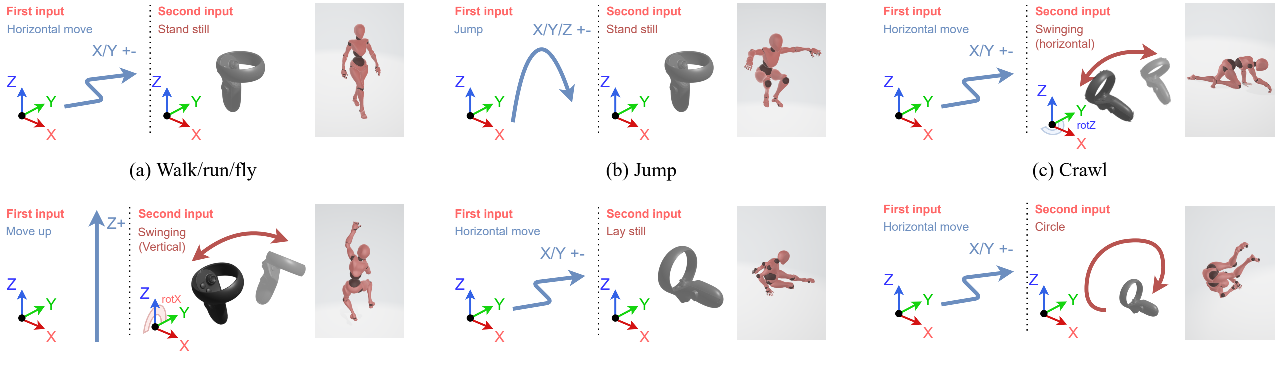
Table 2: Pre-defined action categories in this work

Name	Trajectory input	controller gesture input	Env. information
Idle	No movement	Stand still	On the ground
Walk/run/fly	Horizontal move	Stand still	On the ground/in air
Jump	Jump	Stand still	No walls/in air
Climb	Vertical move	Swing left and right vertically	Near a wall
Crawl	Horizontal move	Swing left and right horizontally	On the ground
Jump over	Jump(slight)	Hurdle	Short walls
Slide	Horizontal move	Lay still	On the ground
Push	Horizontal move	Lean forward	Moveable objects
Pull	Horizontal move	Lean backward	Moveable objects
Lie down	No movement	Lay still	On the ground
Fall	Horizontal down	Stand still	In air
Swing	No movement	Swing back and forth	In air
Roll/flip	Horizontal move	Swing in a circle	On the ground

3.3.1 Data Collection, Pre-processing, and Feature Selection. To train our deep learning model for gesture recognition, we collected 16 types of gesture motions (refer to Table 2) of 15 participants (ages from 20 to 23; 8 males and 7 females; all of them have no professional knowledge on animation creating but with basic knowledge of VR devices) as the training data. All participants were recruited on campus. We had each participants do all the gestures twice to compose a dataset containing 30 samples for each gesture, while each sample was at least 15 seconds in length. We use data augmentation by rotating our captured data around the z-axis 45 degrees, seven times in total. This removes any orientation issue and expands our training data.

The original input data includes two trajectories (of the VR controller), corresponding to the first input and the second input respectively. Then, using the well-known discrete differential calculus, we compute the speed and acceleration from each 1-D signal sequence of positional input. Specifically, instead of using Euler angles or Quaternions to represent 3D rotations, we choose the 6D continuous representation proposed by Zhou et al. [28] due to its compatibility with neural networks. The final input features contain 18 dimensions, which include the velocity and acceleration of the first input, the velocity and acceleration of the second input, as well as the rotation information of the second input.

3.3.2 Deep Learning based Gesture Recognition. The schematic view of our deep learning based gesture recognition model is illustrated in Figure 7. It uses the DeepConvLSTM structure [18] as the basic structure. Additionally, we include Batch Normalization (BN) layers preceding each convolution layer to improve performance [12]. The information that needs to be provided at runtime includes action data (n:1, h:1, w:24, c:18), h_0 and c_0 (n:1, h:1, w:24, c:1). Here h_0 and c_0 are initial values of the LSTM hidden layer parameters, and h_0 and c_0 are filled with 0 constants. We choose to use the



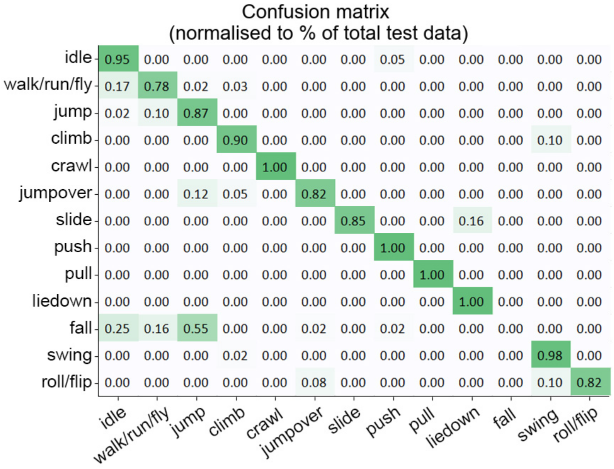
(a) Walk/run/fly (b) Jump (c) Crawl

(d) Climb (e) Slide (f) Roll

Figure 5: Examples for pre-designed gestures in our system. The first input specifies the trajectory of an action type. The second input is to specify the character’s local motion dynamics, following the ergonomics of the VR controller. Since the second input is used to specify the character’s local movement, some actions may have similar second inputs.

classic cross-entropy loss function for multi-classifications in our network.

The input window size when dividing the pre-processed data is set to 24. The sliding stride of the window is 9, Each slide generates input data with a size of (18, 24), along with its associated action label. With a time interval of 0.05 second between samples, a window size of 24 can capture 1.2 seconds of motion. Motions shorter than 1.2 seconds will be ignored by our system. The sequence length *seq_len* is determined experimentally. If it is too short, the sampled sequence will fall short of the minimum period required for self-repeating motions such as walking. Increasing the window size above a certain limit results in decreased precision in recognizing actions and may result in the inclusion of excessive information, causing overfitting of the network.



	idle	walk/run/fly	jump	climb	crawl	jumpover	slide	push	pull	liedown	fall	swing	roll/flip
idle	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00
walk/run/fly	0.17	0.78	0.02	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jump	0.02	0.10	0.87	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
climb	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00
crawl	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
jumpover	0.00	0.00	0.12	0.05	0.00	0.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00
slide	0.00	0.00	0.00	0.00	0.00	0.00	0.85	0.00	0.00	0.16	0.00	0.00	0.00
push	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
pull	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
liedown	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
fall	0.25	0.16	0.55	0.00	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00
swing	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.00
roll/flip	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.10	0.82

Figure 6: The normalized confusion matrix of the test results. Actual prediction results are on the X-axis and the ground truth labels are on the Y-axis.

The model was trained for a total of 200 epochs with a batch size of 24. Test set inference was performed every 10 epochs of the training process, and the resulting accuracy was recorded. The model with the highest accuracy rate on the test set was selected. During training, we obtained a minimum loss function value of 1.879, an accuracy rate of 81.2%, and an optimal F1 score of 0.785.

Figure 6 shows the confusion matrix on the test data. As shown in this figure, our approach is capable of accurately classifying most of the actions studied in this work, with the exception of the “fall” action. A sizable portion of the “fall” motion data is classified as “idle/move/jump.” Further research needs to be done on improving the accuracy of recognizing the “fall” action.

3.3.3 *Real-time Prediction Workflow.*

We implemented our pre-trained model for real-time gesture prediction in *Unity 2019.4.20f1c1*, with the aid of the *Unity Barracuda 1.0.4* plugin.

We present a straightforward example to explicate our prediction process in greater detail. During runtime, we employ the identical moving window utilized for model training, consolidating the input features to predict gestures from the trained neural network model. Each prediction result represents the outcome of prediction within a given moving window. Sequences of actions and their corresponding prediction results are displayed in Figure 8. We illustrate the scenario where the prediction yields a “1-1-1-1-1-” result, indicating that prediction results extracted from five motion segments are under consideration. We then determine the final prediction by invoking a majority voting scheme among all the prediction nodes. As a specific example, our final prediction result is “1-136”, suggesting that the action should be one of walking, running, or flying, having 136 frames. Notably, environment information also plays a vital role in the prediction process. For instance, we might assume that the corresponding environmental context is “grounded”. Based on this assumption, we can infer that the performed action is consistent with a “walk/run” state. By subsequently examining the character’s velocity, we can distinguish between a “walk” or “run” action dependent on whether this velocity surpasses a user-defined threshold.

In our prototype, we use a humanoid character with the height of around 0.2 unity units. We detect the presence of an object right below using a 0.15-unit-long ray casting to provide ground

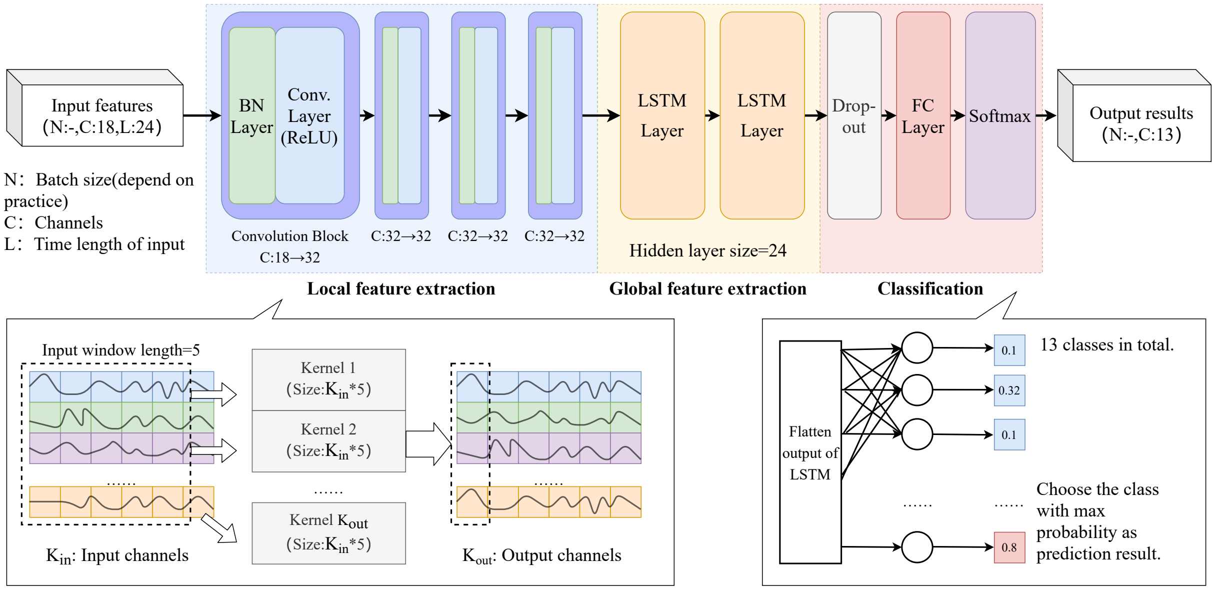
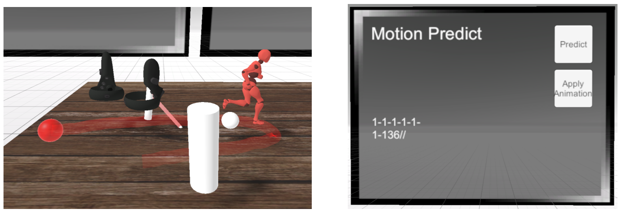


Figure 7: The schematic view of the deep learning based gesture recognition model. The model is customized to meet our requirement for gesture recognition. The Unity Barracuda plugin is used to deploy a pre-trained model for real-time gesture recognition.



1-1-1-1-1-
1-136//

(a) motion sample **(b) corresponding prediction sequence**

Figure 8: An example of motion prediction at runtime.

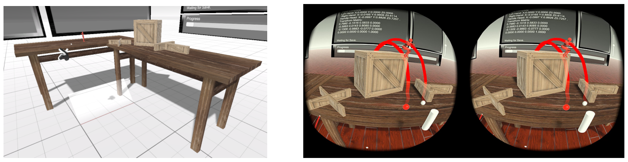
information. Wall existence is detected within a cylinder of the radius of 0.01 units and the height of 0.2 units. This information is used to select the appropriate motion type and adapt the animation.

3.4 Character Animation Synthesis

We utilized the state machine system integrated within the Unity engine to compile pre-established animation clips into a runtime animation that adheres to the end-user's specification. The assortment of animation clips was procured from Mixamo [1].Once the corresponding action sequence has been determined, our system activates the Unity APIs to commence the playback of their associated animations. To ensure the character's proper alignment with the ground, we conducted fine-tuning as a supplementary post-processing step. Additionally, we adjusted the animation playback rate based on the action's speed.

Our approach limits the fade-in and fade-out times to 0.25 seconds to assure a natural transition between two different action animations. The state machine's states permit transitions among each other, contributing to the transition animation's fluidity.

3.5 Double Doodles: Environment Deployment



(a) the experimental VR environment in our system **(b) 3D virtual environment seen by the user**

Figure 9: Snapshots of the Double Doodles system prototype.

We developed a prototype system named "Double Doodles" based on the Unity engine. This system uses Oculus Rift S as the reference device during development. The test VR environment is shown in Figure 9.

At runtime, the user wears a VR head-mounted display headset and holds a VR controller in each hand. The buttons on the VR controllers can be used to trigger action recording or a saving function. The X and Y keys on the left controller can control the first and second action recording. If the user presses and holds the X key, the system will continue to record the first input, capturing the right controller's position and rotation. Releasing the X key indicates the end of the first input. The user can then record the second action in a similar manner. After all recordings are complete, the data can be saved by pushing the B button on the right controller.

Once produced, we can display the animation in a VR environment with user-defined trajectory and motion dynamics.

The Double Doodles system was built and run within the Unity environment, which was deployed on a PC with AMD Ryzen 3800X CPU, Nvidia GTX 1080 GPU, 16GB RAM, and Windows 10 OS. We chose Oculus Rift S as our VR device. Data collection and network training were also accomplished on the same PC.

The system performance indicators we measured include projection time (processing a 10-second input takes 2.49 seconds on average), startup speed (on average 8.76 seconds), and data saving time (on average 0.46 seconds).

4 USER STUDY

4.1 Usability Study

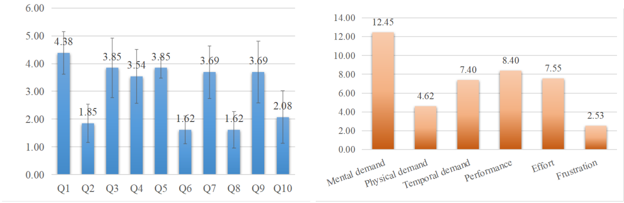
4.1.1 Study Introduction. The objective of this study is to investigate the usability of the "Double Doodles" software from a novice user's point of view. The specific objective is to assess whether the application meets users' needs and evaluates their task performance within the system. This evaluation considers four aspects: (1) user-friendliness, (2) functional completeness, (3) visual effects, and (4) response time. Following the study method of ARAnimator[27], our study also comprises both a system usability test and a task load assessment.

4.1.2 Participants. We recruited 13 participants on campus (ages between 19 and 22; 8 males and 5 females). All of them do not have prior experience or background in animation creation, which well fits the purpose of our study.

4.1.3 Materials. Each participant is asked to utilize our system for creating an action sequence within a VR environment as outlined in section 3.5. The VR environment's configuration is demonstrated in 9a, comprising two tables placed with boxes of various sizes and positions. The largest box is reserved for climbing and jumping animations, whereas smaller ones are designated for use as obstacles. Each participant will receive instructions on how to use the system and perform supported gestures before crafting their animations. The participants will be instructed to create an animation sequence with Idle, Walk, Climb, and Jump. The proposed time frame for task completion per participant is 24 minutes. Participant's primary objective is to produce engaging and authentic character animations that are well-suited to the 3D VR environment.

4.1.4 Procedure. The experimenter helps participants to wear the VR headset and provides instructions before the experiment. After the participants confirm that the VR software functions properly and that the VR headset has been adjusted correctly, the experiment will proceed. During the experiment, participants cannot seek guidance from the experimenter.

We asked participants to fill in two questionnaires after completing the experiment, following the guide in [23]. The first questionnaire is the System Usability Scale (SUS) [3] (as shown in appendix). The second questionnaire is the NASA Task Load Index (NASA-TLX) [11], which is used to evaluate the subjective workload of the participants when using our system. Each participant was also interviewed by the experimenter to obtain feedback on their experience.



(a) the average original SUS scores

(b) NASA-TLX scores

Figure 10: Results of the post-study questionnaires. (a) The mean scores and standard deviations of the original SUS scores. The higher the score for odd-numbered questions, the better; and even-numbered questions are opposite. (b) Averaged workload scores for each index.

4.1.5 Study Results.

SUS Scores. The average original scores of the SUS questionnaire are shown in Figure 10a. Our system scored 71.92 according to the standard SUS score calculation process, which outperforms roughly 73% of similar systems, as reported by the research study conducted by Sauro et al. [14]. Combining the fourth and tenth questions of the SUS questionnaire forms a Learnability indicator, while the other questions together form a Usability indicator [14]. Figure 10a illustrates that the participants gave significantly higher scores on the fourth question, suggesting that participants experienced some difficulty learning the pre-designed gestures, as opposed to relying on their intuition. From the usability perspective, the participants scored our system around 4 on positive remarks (odd-numbered questions) and around 1.8 on negative remarks (even-numbered questions). These results show that our system can achieve a relatively high level of usability, even for novice users.

NASA-TLX Scores. The obtained average NASA-TLX scores are shown in Figure 10b. By summing up the scores, we obtained a total workload score of 42.94. According to the statistics of NASA-TLX scores in different tasks by Grier [9], this score is about the 35th percentile, which means that the workload of our system is approximately lighter than 65% similar systems. As shown in Figure 10b, the participants gave high scores for the mental demand aspect. During our post-study interviews with participants, we found that the second input and motion segmentation processes need considerable brain-power for them.

According to our post-study interviews with the participants, they can understand the system's core concept, interfaces, and usages within ten minutes. They also believed that the final animations produced by our technology accurately depict their design goal. Several character animations created by the participants in our study are shown in the appendix.

4.2 Effectiveness Study

4.2.1 Study Introduction. The purpose of this user study is to evaluate the effectiveness of the proposed "Double Doodles" system. We compare our system with existing methods, such as, ARanimator [27] and Spatial Motion Doodles [7], and explore whether the 3+6 DOFs controlling of the system is better than the 6 DOFs controlling in terms of efficiency and effectiveness. System will be assessed based on four aspects identified as important by experts: (1) animation quality, (2) efficiency, and (3) customization features.

4.2.2 Participants. The study involved 30 participants (ages between 19 and 28; 15 males and 15 females) who have basic familiarity with VR technology and have experience with character animation. Participants are recruited through social media and online forums associated with animation, game design, and VR technology.

4.2.3 Materials. Participants use the "Double Doodles" system to create character animations in a VR environment using motion gestures. The system configuration is similar to that described in subsection 3.5, but with three different control methods. Experimental group 1 utilizes the ARAnimator animation input method, where the VR controller substitutes for the mobile phone's original position. Experimental group 2 uses an input interaction form where participants can input partial movement simultaneously with both hands. The control group uses the original 3+6 DOF control method.

4.2.4 Procedure. The participants will be randomly assigned to one of three groups. Each group will receive a brief tutorial on how to use the assigned system to create character animations using motion gestures. They will be allowed to practice to become familiar with the system. After the tutorial, the participants are asked create a 30-second character animation. The concept for the animation will be the same for both groups, and the participants will be given 10 minutes to complete the task.

After completing the first animation task, the participants will take a break and then create another 30-second animation using the other controlling mechanism. Each participant in the experiment will create the same animation three times, with the three groups rotating to ensure an equal number of participants using the three different system configurations in various orders.

The participants will be asked to fill out a UMUX questionnaire[6] to rate the effectiveness, efficiency, and satisfaction of each controlling mechanism after the experiment.

4.2.5 Study Results. By utilizing UMUX scoring standards[6], experimental group 1 obtained a score of 51.33, experimental group 2 obtained 54.31, and the control group achieved 61.25 points. The error caused by the order of using the system has been eliminated by controlling the number of participants with different usage orders.

An ANOVA significance test was conducted on the outcomes of three groups of interaction experiments. The results demonstrated significant main effects of the interaction methods (F (2, 58) = 11.98, $p < 0.001$). Furthermore, Tukey's Honestly Significant Difference (HSD) test revealed that all differences between the interaction methods were statistically significant ($p < 0.001$).

The results indicate that the 3+6 DoFs controlling group outperformed the 6 DoFs controlling group in terms of efficiency and effectiveness when generating character animations using motion gestures. 3+6 DoFs controlling group participants created animations more swiftly and accurately, resulting in a higher satisfaction level with the system.

Nevertheless, for the 3+6 DoFs design, performing simultaneous operations is still unrealistic for users. In terms of usability, the score of experimental group 2 is significantly lower than the score achieved by the control group under the two-step input scenario.

5 DISCUSSION

We found that it took approximately one minute for the participants to correctly generate an animation sequence involving three actions. We demonstrate that the Double Doodles system in terms of ease of use and the capability to match user needs. The primary learning cost comes from the new input mode and its associated relationship with actions and animations. However, in general, users can quickly learn the basic use of our system, demonstrating that users, including novices, can learn and well grasp the concept of two-staged inputs in a reasonably short time.

In this paper, we have not studied how the user's expertise affects the quality of their animations. It requires considerable work to transform this prototype method into a system with more operations, which users may perform and design scenarios that can be extended to target users' requirements. Additionally, we are yet to identify a suitable way to conduct comparative evaluations with closely related animation systems because they are designed for different immersive scenes and input devices. It is challenging to determine a compelling approach to evaluate them.

In the future, we plan to add more actions to our system, leveraging the powerful flexibility of our 3+6 DOFs-based control method for VR gesture design. In order to incorporate new actions, it is necessary to have a larger user base to perform these actions and gather the corresponding motion data. This data is then used to train classification neural networks for action segmentation and recognition. Once trained, these actions can be seamlessly integrated into our system. Also, the manner in which certain actions are expressed also could be improved to increase their identifiability. Moreover, we plan to extend the use of our method to animate non-human characters in VR as part of our future research.

6 CONCLUSION

To address the fundamental DOF gap that is well recognized in existing approaches for interactive character animation generation in VR environment, we offer an in-situ, 3+6 DOFs, two-stage-inputs method to generate character animations in an immersive VR environment interactively. Also, we introduce a set of basic gestures that can be performed with VR controllers as metaphors for character animations. Finally, through a carefully-designed user study we demonstrate that our approach can be effectively used for VR-based interactive character animation generation in practice.

Our system utilizes the advantages of VR/AR environments, providing users with an intuitive, user-friendly, and efficient method for creating character animations. It can be applied to various domains, including virtual reality filmmaking, stage rehearsal, and the future of storytelling. The combination of stereoscopic 3D perception, room-scale space, and 3+6 DoF input devices offers a natural interface for performing 3D character animation.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (NSFC, NO. 62102255), NSFC (No. 61972157), and Zhigang Deng was supported in part by US NSF IIS 2005430.

REFERENCES

[1] Adobe. 2022. Mixamo. https://www.mixamo.com.

[2] Raphael Anderegg, Loïc Ciccone, and Robert W. Sumner. 2018. PuppetPhone: Puppeteering Virtual Characters Using a Smartphone. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (Limassol, Cyprus) *(MIG '18)*. Association for Computing Machinery, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/3274247.3274511

[3] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *Int. J. Hum. Comput. Interact.* 24, 6 (2008), 574–594. https://doi.org/10.1080/10447310802205776

[4] Amal Benzina, Marcus Toennis, Gudrun Klinker, and Mohamed Ashry. 2011. Phone-Based Motion Control in VR: Analysis of Degrees of Freedom. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI EA '11)*. Association for Computing Machinery, New York, NY, USA, 1519–1524. https://doi.org/10.1145/1979742.1979801

[5] Ze Dong, Thammathip Piumsomboon, Jingjing Zhang, Adrian Clark, Huidong Bai, and Rob Lindeman. 2020. A Comparison of Surface and Motion User-Defined Gestures for Mobile Augmented Reality. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3334480.3382883

[6] Kraig Finstad. 2010. The Usability Metric for User Experience. *Interacting with Computers* 22 (09 2010), 323–327. https://doi.org/10.1016/j.intcom.2010.04.004

[7] Maxime Garcia, Remi Ronfard, and Marie-Paule Cani. 2019. Spatial Motion Doodles: Sketching Animation in VR Using Hand Gestures and Laban Motion Analysis. In *Motion, Interaction and Games* (Newcastle upon Tyne, United Kingdom) *(MIG '19)*. Association for Computing Machinery, New York, NY, USA, Article 10, 10 pages. https://doi.org/10.1145/3359566.3360061

[8] Google. 2022. Tilt Brush. https://www.tiltbrush.com/.

[9] Rebecca A. Grier. 2015. How High is High? A Meta-Analysis of NASA-TLX Global Workload Scores. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 59, 1 (2015), 1727–1731. https://doi.org/10.1177/1541931215591373

[10] Jens Grubert, Eyal Ofek, Michel Pahud, and Per Ola Kristensson. 2018. The Office of the Future: Virtual, Portable, and Global. *IEEE Comput. Graph. Appl.* 38, 6 (Nov. 2018), 125–133. https://doi.org/10.1109/MCG.2018.2875609

[11] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (2006), 904–908. https://doi.org/10.1177/154193120605000909

[12] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 448–456. http://proceedings.mlr.press/v37/ioffe15.html

[13] Fabrizio Lamberti, Gianluca Paravati, Valentina Gatteschi, Alberto Cannavò, and Paolo Montuschi. 2018. Virtual Character Animation Based on Affordable Motion Capture and Reconfigurable Tangible Interfaces. *IEEE Transactions on Visualization and Computer Graphics* 24, 5 (2018), 1742–1755. https://doi.org/10.1109/TVCG.2017.2690433

[14] James R. Lewis and Jeff Sauro. 2009. The Factor Structure of the System Usability Scale. In *Human Centered Design, First International Conference, HCD 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009, Proceedings (Lecture Notes in Computer Science, Vol. 5619)*, Masaaki Kurosu (Ed.). Springer, 94–103. https://doi.org/10.1007/978-3-642-02806-9_12

[15] Hui Liang, Jian Chang, Ismail Khalid Kazmi, Jian J. Zhang, and Peifeng Jiao. 2017. Hand gesture-based interactive puppetry system to assist storytelling for children. *Vis. Comput.* 33, 4 (2017), 517–531. https://doi.org/10.1007/s00371-016-1272-6

[16] Aline Menin, Rafael Torchelsen, and Luciana Nedel. 2018. An Analysis of VR Technology Used in Immersive Simulations with a Serious Game Perspective. *IEEE Comput. Graph. Appl.* 38, 2 (March 2018), 57–73. https://doi.org/10.1109/MCG.2018.021951633

[17] Juliana Montes and Pablo Figueroa. 2019. VR Salsa: Learning to Dance in Virtual Reality. In *Proceedings of the IX Latin American Conference on Human Computer Interaction* (Panama City, Panama) *(CLIHC '19)*. Association for Computing Machinery, New York, NY, USA, Article 2, 4 pages. https://doi.org/10.1145/3358961.3358969

[18] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016), 115. https://doi.org/10.3390/s16010115

[19] Ye Pan and Kenny Mitchell. 2020. Group-Based Expert Walkthroughs: How Immersive Technologies Can Facilitate the Collaborative Authoring of Character Animation. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 188–195. https://doi.org/10.1109/VRW50115.2020.00041

[20] Ye Pan and Kenny Mitchell. 2020. PoseMMR: A Collaborative Mixed Reality Authoring Tool for Character Animation. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 758–759. https://doi.org/10.1109/VRW50115.2020.00230

[21] D.J. Sturman. 1998. Computer puppetry. *IEEE Computer Graphics and Applications* 18, 1 (1998), 38–45. https://doi.org/10.1109/38.637269

[22] Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion Doodles: An Interface for Sketching Character Motion. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 424–431. https://doi.org/10.1145/1015706.1015740

[23] Thomas Tullis and William Albert. 2013. *Measuring the User Experience, Second Edition: Collecting, Analyzing, and Presenting Usability Metrics* (2nd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[24] Tvori. 2022. Tvori. https://tvori.co/.

[25] Andreia Valente, Augusto Esteves, and Daniel Lopes. 2021. From A-Pose to AR-Pose: Animating Characters in Mobile AR. In *ACM SIGGRAPH 2021 Appy Hour* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3450415.3464401

[26] Daniel Vogel, Paul Lubos, and Frank Steinicke. 2018. AnimationVR - Interactive Controller-based Animating in Virtual Reality. In *2018 IEEE 1st Workshop on Animation in Virtual and Augmented Environments (ANIVAE)*. 1–6. https://doi.org/10.1109/ANIVAE.2018.8587268

[27] Hui Ye, Kin Chung Kwan, Wanchao Su, and Hongbo Fu. 2020. *ARAnimator*: In-Situ Character Animation in Mobile AR with User-Defined Motion Gestures. *ACM Trans. Graph.* 39, 4, Article 83 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392404

[28] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the Continuity of Rotation Representations in Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5738–5746. https://doi.org/10.1109/CVPR.2019.00589

A SUS QUESTIONNAIRE

This is the SUS Questionnaire used in our study.

Table 3: The SUS post-study Questionnaire

No.	Question
1	I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

B UMUX QUESTIONNAIRE

This is the UMUX Questionnaire used in our study.

Table 4: The UMUX Questionnaire

No.	Question
1	This prototype capabilities meet my requirements.
2	Using this prototype is a frustrating experience.
3	This prototype is easy to use.
4	I have to spend too much time correcting things with this prototype.