ARAnimator: In-situ Character Animation in Mobile AR with User-defined Motion Gestures

HUI YE*, KIN CHUNG KWAN*, WANCHAO SU, and HONGBO FU†, City University of Hong Kong

Fig. 1. Our *ARAnimator* allows users to move an AR-enabled mobile device to directly control and animate a virtual character situated in a real-world environment. Please refer to the accompanying video for the animation results.

Creating animated virtual AR characters closely interacting with real environments is interesting but difficult. Existing systems adopt video see-through approaches to indirectly control a virtual character in mobile AR, making close interaction with real environments not intuitive. In this work we use an AR-enabled mobile device to directly control the position and motion of a virtual character situated in a real environment. We conduct two guessability studies to elicit user-defined motions of a virtual character interacting with real environments, and a set of user-defined motion gestures describing specific character motions. We found that an SVM-based learning approach achieves reasonably high accuracy for gesture classification from the motion data of a mobile device. We present *ARAnimator*, which allows novice and casual animation users to directly represent a virtual character by an AR-enabled mobile phone and control its animation in AR scenes using motion gestures of the device, followed by animation preview and interactive editing through a video see-through interface. Our experimental results show that with *ARAnimator*, users are able to easily create in-situ character animations closely interacting with different real environments.

CCS Concepts: • **Human-centered computing** → *Gestural input*; *Mixed / augmented reality*; **Mobile devices**.

Additional Key Words and Phrases: Mobile Augmented Reality, Interactive System, Character Animation, User Defined Gestures, Gesture Classification

ACM Reference Format:
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

1 INTRODUCTION

Augmented Reality (AR) aims to augment a real world with virtual content, often with proper alignment. With AR technologies, augmenting virtual *static* objects into our real world is relatively easy. However, even with powerful mobile AR platforms recently developed by Apple (i.e., ARKit) and Google (i.e., ARCore), creating animated contents closely interacting with real environments is still challenging. Since virtual objects need to be aligned with real environments at different moments, making the reuse of existing animations difficult. In-situ creation of animation thus becomes more important.

Several tools such as Motion Doodles [Thorne et al. 2004], Spatial Motion Doodles [Garcia et al. 2019] and PuppetPhone [Anderegg et al. 2018] have been designed to allow novice users to create character animations on top of a virtual or real scene. They can be directly applied or extended for in-situ creation of AR animation. However,

*Both authors contributed equally to the paper.
†Corresponding author.

Authors' address: Hui Ye, huiye4-c@my.cityu.edu.hk; Kin Chung Kwan, kckwan@ieee.org; Wanchao Su, wanchaosu2-c@my.cityu.edu.hk; Hongbo Fu, hongbofu@cityu.edu.hk, City University of Hong Kong, School of Creative Media, 18 Tat Hong Avenue Kowloon Tong, Hong Kong.

0730-0301/2020/7-ART83 $15.00
https://doi.org/10.1145/3386569.3392404

these approaches either require precise scene understanding to detect 3D geometry surfaces for character interaction, or highly rely on room-based motion capture (MoCap) hardware with complicated setup (e.g., VR). Thus, it is challenging to use these tools for creating in-situ character animation closely interacting with complex physical environments, such as an outdoor environment with trees.

In this work, we aim to design and develop an intuitive tool for creating in-situ character animation in complex real-environments without the requirement of precise scene understanding or additional MoCap hardware. Our target users are causal animation users, who want to quickly create in-situ character animations without lots of training or hardware setup. Thus, instead of elaborating animation effects, our system design focuses more on the *cost, accessibility, portability*, and *ease-to-learn*.

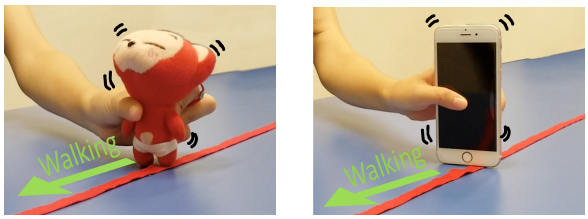
To this end, we explore the use of an AR-enabled mobile phone as a direct controller for in-situ performance-based character animation in mobile AR (Figure 1). Our key idea is to mimic the process of make-believe plays using dolls for storytelling (Figure 2(a)): users control a mobile phone (Figure 2(b)) to move the character on and around real objects, and perform motion gestures simultaneously to retrieve existing animations from a character animation repository. This is an intuitive process even for casual users, since most of us have similar experience of such plays in our childhood.

However, there is a huge gap between character animations with a high degree of freedom (DoF), and the possible gestures on a mobile phone, which is a rigid object, with a low DoF. To address this issue, we first need to obtain a proper list of motion gestures that are expressive and intuitive. This motivates us to conduct two elicitation studies (Section 3). In the first study, users are asked to come up various possible character motions that closely interact with various physical environments. The second study allows users to define motion gestures that describe specific character motions, summarized from the first study. Observing the results of the two studies, we find that 6-DoF motion gestures are commonly used for most of the motion types. We thus design and develop a prototype, *ARAnimator*, which allows casual users to directly represent a virtual character by an AR-enabled mobile phone and control its animation in AR scenes using 6-DoF motion gestures of the device, followed by animation preview and interactive editing through a video see-through interface (Figure 6). We show that an SVM (Support Vector Machine) based learning approach achieves reasonably high accuracy for motion gesture classification. Our quantitative and qualitative evaluation show that *ARAnimator* provides users with an intuitive, easy-to-use and effective way to create in-situ character animations.

2 RELATED WORK

2.1 In-situ Character Animation in AR/VR

The state-of-the-art AR technologies allow users to place animated virtual characters or scenes in real environments. However, most of existing works are highly dependent either on pre-defined physical objects with markers for animation [Bai et al. 2015; Barakonyi and Schmalstieg 2004, 2006] or as interaction environments [Cimen et al. 2018], or on complex setups and devices for creating animation effects [Kim et al. 2014; Osato and Koizumi 2018]. Such requirements



(a) "Doll drama" (b) Our method

Fig. 2. We can intuitively control a mobile phone similar to the way we play with a doll, to tell animated stories.

restrict the flexibility of those techniques for their use in different scenarios. Besides, they support only static effects or limited predefined animation effects in specific locations.

Motion Doodles (MD) [Thorne et al. 2004] is a pioneering animation system for novice users, and mainly uses 2D curves with 2-DoF gestures to define virtual character animations. To extend it to 3D, 3D geometry proxies are needed for projecting 2D curves to 3D trajectories. Lockwood and Singh (LS) [2016] present the first work utilizing 3-DoF motion sensor data of a mobile device to control a character on a virtual plane surface (detected flat surfaces for extending to AR). PuppetPhone (PP) [Anderegg et al. 2018] follows a similar idea and is tailored for in-situ character animation with mobile AR. PP places a virtual character on a detected physical plane in front of the camera with a certain distance (i.e., MotionStick), and controls a character using 3-DoF device motions.

These three methods require accurate scene understanding or even manual creation of 3D planes in an AR world for in-situ creation of AR animations. However, such problems are very challenging on their own for complex real environments. Besides, they all utilize indirect approaches which rely on video see-through interfaces to control a character. Such indirect approaches thus require the camera to be always facing to a character's current position during animation authoring. It thus restricts the phone orientation and results in a lower DoF (i.e., 2 or 3). Their low-DoF design cannot provide free-form character orientation control requiring the character to always be vertical and heading along the trajectory, and they are less expressive to provide various character motions. In contrast, our direct interaction based on 6-DoF motion gestures represents a virtual character and its character motions using a mobile device and its device motions. Section 3.3 gives more detailed discussions on the expressiveness of 6-DoF gestures.

Very recently, Garcia et al. [2019] present Spatial Motion Doodle (SMD), which is a 3D extension of Motion Doodles for exploring 3D character animations in VR scenes. They utilize an outside-in VR MoCap system to track a VR controller and map its 3D movement to a virtual character to create a 3D trajectory with free-form 3D orientation control. However, due to the complicated setup of VR hardware, it is inconvenient to apply their system for in-situ animation creation in various environments. More importantly, its limited 3-DoF gestures defined by their authors for animation imply that they are less expressive compared to our 6-DoF user-defined gestures.

These four works are very closely related to ours. We list the major differences in Table 1. In summary, *ARAnimator* has the following

Table 1. The differences between our technique and existing tools that can be potentially extended for in-situ character animations in AR. MD (3D): Motion Doodles with 3D extension [Thorne et al. 2004], SMD: Spatial Motion Doodles [Garcia et al. 2019], LS: Lockwood and Singh [2016], PP: PuppetPhone [Anderegg et al. 2018]. "Y" and "N" represent yes and no, respectively.

Name	MD (3D)	SMD	LS	PP	Ours
Designed for VR/AR	VR	VR	VR	AR	AR
Control type	indirect	direct	indirect	indirect	direct
Orientation control	N	Y	N	N	Y
Work without planes	N	Y	N	N	Y
Gestures	2-DoF	3-DoF	3-DoF	3-DoF	6-DoF
Gesture design	authors	authors	authors	authors	users
Classification	Regexp	Regexp	Manual	FSM	SVM
Devices	PC	HTC Vive	Mobile	Mobile	Mobile

advantages: 1) It does not require any detected plane or surface to work well in complex real environments. 2) Our 6-DoF system design allows free-form character orientation controls and more expressive creation. 3) The intuitive user-defined 6-DoF gestures make our system easy to use for casual users.

2.2 Computer Puppetry and Character Animation

Like performing a puppet play, hardware-based computer puppetry provides users with a specific tangible and/or articulated puppet hardware to manipulate for 3D virtual character animation creation [Glauser et al. 2016; Gupta et al. 2014; Hiroki et al. 2012; Lamberti et al. 2017; Oore et al. 2002; Shiratori et al. 2013; Wang et al. 2018; Yoshizaki et al. 2011]. However, these techniques require specialized devices and hardware (e.g., robot, blocks), or built-in sensors (e.g., electronic and magnetic sensors) and peripheral cameras, thus limiting their accessibility for casual users.

Another kind of puppetry is based on performance-based interfaces, which attempt to map human fingers [Lockwood and Singh 2012], hands [LEite and Orvalho 2017; Liang et al. 2017; Luo et al. 2010], hand shadow [Tsuji et al. 2018], or full body [Dontcheva et al. 2003; Sakashita et al. 2017; Seol et al. 2013] information from motion sensors to the parameters of a virtual character. With these interfaces, users can edit inherent motions of a character, while interactive control of a character's global moving trajectory is not well supported. In contrast, our work uses an AR-enabled mobile device as a puppet for intuitive creation of character animation along a user-defined trajectory and with close interaction with real-world environments.

3D animation tools [Choi et al. 2016; Ciccone et al. 2017, 2019; Eom et al. 2019; Fender et al. 2015; Koyama and Goto 2018] provide interfaces for users to manipulate 3D character animation by widget controls and/or sketch inputs. Similar to performance-based interfaces, these tools do not support direct control of global moving trajectories. Thus, they are not suitable for in-situ animation.

2.3 User-defined Motion Gestures

User-defined motion gestures for mobile interaction were first explored by Ruiz et al. [2011]. Later researchers have conducted guessability studies to elicit user-defined motion gestures for various tasks, such as text processing [Zhu et al. 2017], navigation [Dim and Ren 2014; Shimon et al. 2015], and 3D manipulation [Liang et al. 2012]. However, none of them considers the mapping between 6-DoF user-defined motion gestures and articulated character animations. Our work focuses on eliciting user-defined motion gestures for abundant character motions, and using such gestures together with a moving trajectory to create 3D character animations situated in real environments.

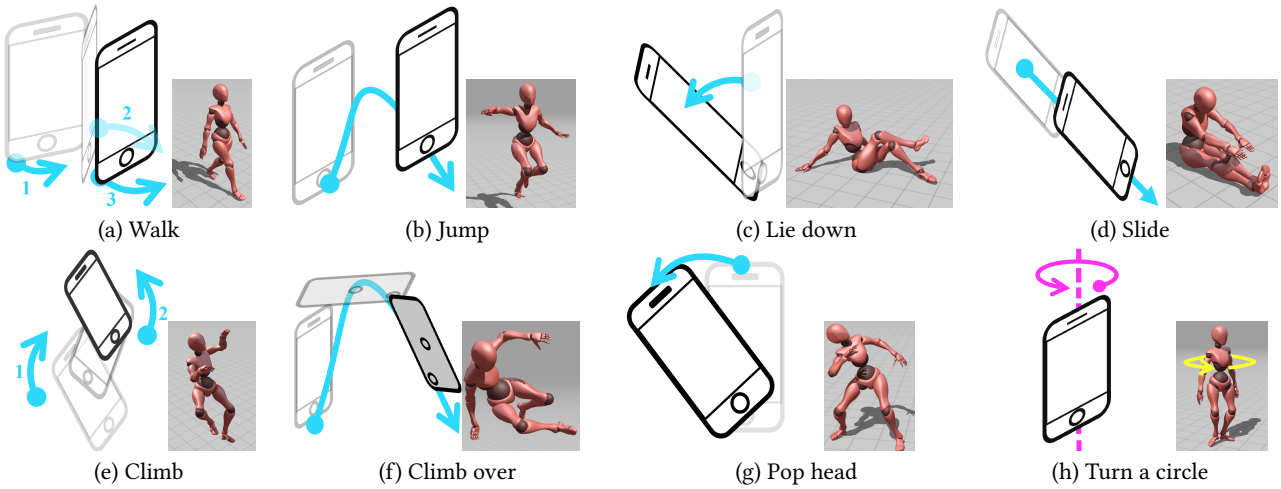
3 USER-DEFINED CHARACTER MOTIONS & GESTURES

Our core idea is to mimic the make-believe doll-play for in-situ AR animation creation. Creating animated scenes with virtual characters usually requires the support of various character motions. However, due to its rigid nature, a doll or mobile device (i.e., a mobile phone in our case) itself is not flexible enough to represent complex and detailed motions (e.g., motion details of dancing). For make-believe, users often perform 6-DoF motion of a doll to depict its complicated character motions. Thus, before designing our system, it is essential to study: 1) "*What* character motions are commonly wanted by novice and casual users to create animated AR scenes?" and 2) "*How* to perform motion gestures corresponding to these character motions with a mobile phone?" To answer these two questions, we conduct two elicitation studies.

3.1 Study I: User-defined Character Motions

In the first study, we aimed to know what character motions are often wanted by novice and casual users for in-situ character animation creation. Since our goal was to explore the possible motions of virtual characters in a human-liked shape, to avoid any bias from the shape of a doll/phone, we selected a human-liked rigid doll to do this study instead of a mobile device (i.e., a cuboid). 5 participants (aged 25-27, 2 females) were invited to do the study. One participant had a little experience in creating 3D animations, while others did not. We first showed them 6 different real scenes of great variety, involving indoor/outdoor, complex/clean, with flat/curvy surface, and tall/short structures. Then each participant was asked to think of any possible character motions interacting with the real-world objects in the given scenes. To come up with as many motions as possible, each participant was given about one day to think. Then the participants came back and performed their designed motions or even stories by holding and moving a given doll (Figure 2(a)) in the respective scenes with a think-aloud strategy. This strategy was important for us to understand the intentions and the designs of the motions from the participants.

In this study we observed in total 43 types of character motions from all the participants. However, the motions observed in this study do not necessarily mean that they are commonly wanted, as some motions might come from the inspiration of single individual users. Thus we filtered out 14 unusual motions (i.e., appeared only once during the study). Figure 5 lists the remaining 29 motions. Among all of them, the most frequently used motions are "jumping" (frequency: 36), "climbing" (frequency: 17), and "walking" (frequency: 15) among 172 motions (including those with the same types) in total. Most participants claimed that with these three motions, they could design many interesting short animations around various real-world objects.



(a) Walk (b) Jump (c) Lie down (d) Slide

(e) Climb (f) Climb over (g) Pop head (h) Turn a circle

Fig. 3. Examples of the motion gestures collected from Study II, and their corresponding motions.

3.2 Study II: User-defined Motion Gestures

Since we want to explore an AR-enabled mobile device as a direct controller, it is essential to figure out the mapping between articulated character motions and a mobile device. The mapping can be achieved by a motion-gesture based solution. However, it is challenging, since the required number of motion gestures is not small and manually designing a moderately large set of intuitive gestures is not an easy task. To get a set of both expressive and intuitive gestures, we adopt a gesture elicitation approach for user-defined gestures, instead of defining them by ourselves, as done in the existing works [Garcia et al. 2019; Lockwood and Singh 2016; Thorne et al. 2004]. Thus, in the second study, we explore how users define motion gestures of the mobile device (mobile phone in our case) corresponding to the 29 character motions from Study I. We invited 12 participants (aged 24-28, 6 females). All of them were daily smartphone users and right-handed. Two participants had relevant experience of creating 3D character animation. We asked each participant to design a motion gesture that can best represent each of the character motions. They used a mobile phone to imitate an animated character. Each participant was given an iPhone 6S, with the screen turned off so that no button or touchscreen input was allowed. The participants were required to perform each designed gesture in 5 seconds, and repeated 3 times for consistency for each character motion.

To help each participant design unique gestures (i.e., different motion gestures for different character motions), we asked them to go through the list of motions involved and think about different gestures for different motions before performing. We also categorized the 29 motions into 5 groups (locomotion motion, object-dependent motion, in-place motion, mid-air motion, and rotational motion) according to their similarities. This categorization helped each participant memorize and distinguish their designed gestures in a short period, thus avoid designing similar gestures in the same group. The grouped motion list is in the supplemental materials. Before starting each group, each participant first watched the animated models corresponding to all the motions in this group, and then designed all the motion gestures for the corresponding character motions in the same group. The participants were allowed to revisit the animated models and refine their designed gestures any time. Once they were satisfied with their design, they could proceed to the next group of character motions. The whole designing process was video-taped for further observation and analysis. After finishing all the tasks, the participants were asked to rate their own gestures in term of goodness, ease-of-planning, and ease-of-performing on seven-point Likert scales (i.e., 1 = strongly disagree to 7 = strongly agree).

We collected a total number of 1,044 motion gestures. We manually observed their performances (through the recorded videos) and categorized gestures for each motion using open coding. The gesture type with the largest number for each motion was chosen as the representative gesture for each motion (See Figure 3 for examples). Please refer to our accompanying video for the final gestures of 29 motions.

We found that no participant designed similar gestures for motions from different groups. Some participants designed similar gestures for motions in the same groups (e.g., walk and run, lie down and sit down), since such gestures might differ only on motion parameters (e.g., speed, bending angle). To evaluate the degree of consensus among user-defined gestures, we calculated the agreement score A using the equation of Wobbrock et al. [2005]:

$$A_t = \sum_{P_i} \left(\frac{|P_i|}{|P_t|} \right)^2, \tag{1}$$

where t is one of the character motions being studied, P_t is the set of all the collected gestures for t, and P_i is a subset of identical gestures from P_t. The range for A is [0,1].

Figure 5 illustrates the agreement rate of gestures designed by the participants for all the character motions. For 16 motions (e.g., walk, climb over) in our list, all the participants designed and performed identical gestures for the same motions. Most of the motions (79.31%) had very high agreement scores (over 70%). We got the results of rating on goodness (M: 5.20, SD: 0.71), ease-of-planning (M: 5.63, SD: 0.51), and ease-of-performing (M: 5.55, SD: 0.60). The high scores evidence that mapping motion gestures with character motions was easy, natural, and intuitive. Please refer to our supplementary materials for the detailed rating of each gesture.

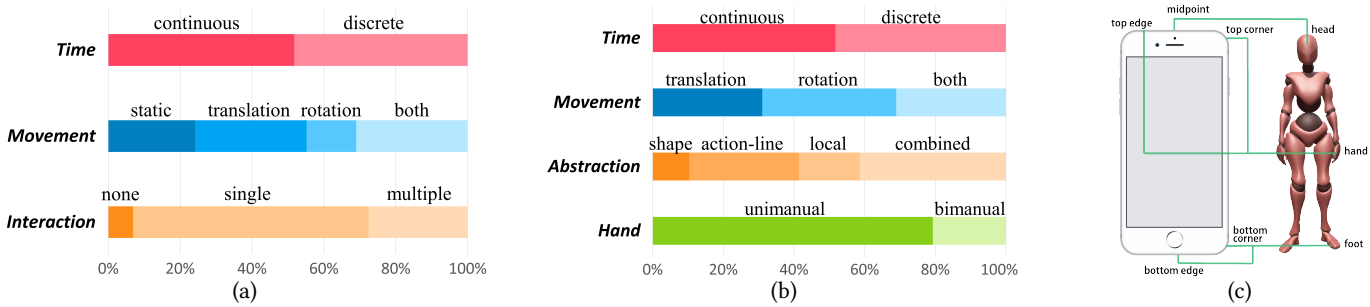


Fig. 4. (a) Percentages of motions in each taxonomy category. (b) Percentages of motion gestures in each taxonomy category. (c) Illustration of mapping between a mobile phone and a virtual character.

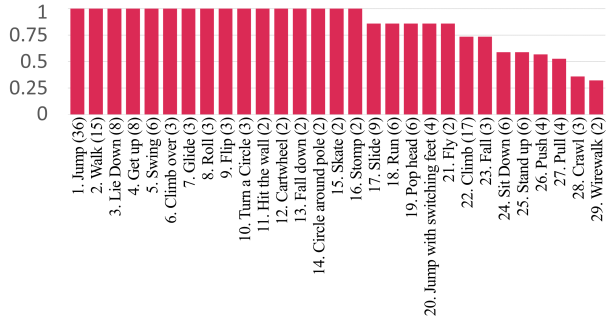


Fig. 5. Agreement rates of user-defined motion gestures for character motions in Study II. The numbers in the bracket next to the motion names are their frequency values.

This was confirmed by the feedback of the participants. All the participants said that it was very easy to think about a corresponding motion gesture for most motions. They agreed that a virtual character could be mapped to a mobile phone well. They believed the goal of our project was nice and promising. Next we will discuss some findings from two studies.

3.3 Discussions on the Findings

Taxonomy of Motions. We categorize the collected character motions along three separate dimensions: time, movement, and interaction, as summarized in Table 2. The time dimension describes the temporal features of motions. The movement dimension characterizes the global movement of a character in the world space. Finally the interaction dimension captures how a character interacts with a physical environment. It differs in *none*, *single*, and *multiple* in terms of the number of interacting surfaces (typically planar surfaces). We consider two interacting surfaces are different surfaces if they are disconnected (e.g., jump from one to another) or there are obvious changes between the normal of the surfaces (e.g., climb up from floor to wall).

Figure 4(a) illustrates the breakdown of all the motions using the above taxonomy. For in-situ animation, we observed that:

- Motions involving translation (62.1%) were commonly needed for virtual characters to travel in the world.
- Most (65.5%) of the motions interacted with a single surface. However, motions that interacted with multiple surfaces (27.6%) were still needed for characters to move between different surfaces. Thus, systems [Anderegg et al. 2018; Lockwood and Singh 2016] that work only on a single detected floor are insufficient for in-situ animations.

Table 2. Taxonomy of character motions for in-situ animation.

Time	*Continuous*	*Loop continuously by period without time interruption between periods.*
	Discrete	*Occur only once for each time, and wait in time interval for the next occurrence.*
Movement	*Static*	*Stay in the position.*
	Translation	*With translation only.*
	Rotation	*With rotation only.*
	Both	*Motion with both translation and rotation.*
Interaction	*None*	*Do not interact with any surface.*
	Single	*Interact with a single surface.*
	Multiple	*Interact with more than one surface.*

Table 3. Taxonomy of motion gestures for character animation.

Time	*Continuous*	*Repeat in a time interval.*
	Discrete	*Perform only once.*
Movement	*Translation*	*Change the 3D position of a device.*
	Rotation	*Change the 3D orientation of a device.*
	Both	*Translate and rotate a device (6-DoF).*
Abstraction	*Shape*	*Follow the shape of a desired moving trajectory.*
	Action-line	*Imitate the orientation of a character's action line.*
	Local	*Imitate the movement of individual local parts of a character.*
	Combined	*Involve two or more of the above types.*
Hand	*Unimanual*	*Use one hand to perform.*
	Bimanual	*Use two hands to perform.*

Taxonomy of Motion Gestures. As summarized in Table 3, we categorize the collected motion gestures according to four dimensions: time, movement, abstraction, and hand. Time classifies motions gestures according to the temporal information by *continuous* and *discrete*. Movement describes how the translation and rotation of the phone contribute to the gestures. Abstraction describes how users abstract the motions into the corresponding gestures. More specifically, *shape* refers to the gestures that are performed simply following the shape of a desired moving trajectory (e.g., jump). This is similar to the 2-DoF gestures used in Motion Doodles [Thorne et al. 2004]. *Action-line* refers to the gestures following the rough orientation change of the action line of a character (e.g., pop head). *Local* refers to the gestures imitating the local movement of individual parts of a character with different parts of the device (e.g., walk). *Combined* refers to the gestures that use a combination of abstraction types to represent the corresponding motions. Finally, we have *unimanual* and *bimanual* motion gestures for the hand category.

Figure 4(b) illustrates the breakdown of all the motion gestures using above taxonomy of motion gestures. We observed that:

- A continuous motion was often represented by a continuous gesture, and vice versa.
- All of the static motions were represented by rotational gestures, and 68.9% of the motions required rotational information.
- It was not very common (10%) to purely abstract the shape of a moving trajectory into a gesture. 41.3% of the gestures required a combination of abstraction types.
- The participants tended to use a roughly consistent mapping between local parts of a virtual character and parts of a mobile phone, as illustrated in Figure 4(c). For example, feet were often represented by the bottom corners and edge of the mobile phone.
- Most (79.31%) of the gestures were performed by using one hand. Bimanual gestures were found with a lower mean score (5.21) of ease-of-performing than unimanual gestures (5.64).

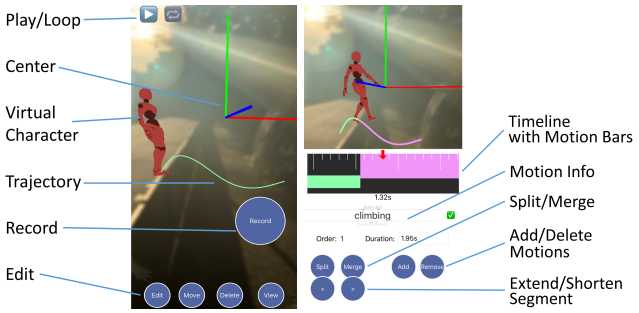
Expressiveness of 6-DoF Gestures. Through Study II, we found that if only 3-DoF gestures were allowed to use, they could only represent around 37.9% of the motion gestures designed by the users. For example, using the shape of a moving trajectory alone to represent a motion gesture would not distinguish between "jump" (Figure 3(b)) and "climb over" (Figure 3(f)) or between "lie down" (Figure 3(c)) and "pop head" (Figure 3(g)). This verifies the importance of using 6-DoF gestures for our problem. Compared with the 2-DoF or 3-DoF gestures used in other VR/AR animation systems (Table 1), our gesture set contains not only 3-DoF gestures but also 6-DoF gestures, allowing us to design an intuitive and expressive system more easily.

4 ARANIMATOR SYSTEM

Based on the collected motions and corresponding motion gestures, we have designed and developed a prototype of in-situ animation creation, named *ARAnimator*, using mobile AR. For demonstration purposes, our current implementation only supports 15 out of 29 motions in our list due to the limited set of corresponding animation clips we can find online[1].

Our *ARAnimator* allows users to take a single mobile phone as a 6-DoF controller and use it to represent a single virtual character, which will be animated in-situ using motion gestures in real environments. To accomplish this, we employ a modern mobile AR platform, Apple ARKit in our implementation. The simple idea is to map the 3D pose of a motion-tracked mobile phone directly to the pose of the virtual character. This allows users to animate the virtual character in situ by simply moving the mobile phone along a desired 3D path, and to perform continuous motion gestures simultaneously (similar to puppetry). Our system then automatically analyzes the gestures and classifies the motion of each segment on the trajectory. The animation results are displayed on the mobile phone screen, allowing users to quickly preview the animation in situ.

However, when the mobile phone interacts with physical environments closely, ARKit sometimes loses tracking due to camera occlusion and/or textureless surfaces in the view, leading to a problematic path of the animated character. In addition, due to the inconsistency of user performance, *ARAnimator* sometimes fails to provide correct motion gesture recognition results. In order to resolve incorrect gesture recognition results, we provide an editing mode for users to interactively refine the animation results. Figure 6 shows the main user interface of our *ARAnimator* prototype.



(a) Preview mode (b) Editing mode

Fig. 6. The user interface of our *ARAnimator*.

4.1 Authoring

To create 3D character animation, a user first presses the "Record" button (Figure 6(a)) to start. Then, the user can start to move the mobile device and meanwhile performs specific motion gestures representing desired motions. To reduce the chance of occlusion of camera, we recommend the user to hold the mobile device with the back camera facing backward (with respect to the moving direction) (Figure 2(b)). To stop recording, the user simply presses the "Record" button again. The collected pose data will then be sent to a gesture classifier for motion gesture classification. Afterwards, a predefined virtual character and its moving trajectory are displayed on the screen for preview and editing. The playback of the synthesized animation is controlled by the "Play" button. In our prototype, we allow users to repeat the above process to create multiple animated characters to generate various animated AR scenes.

4.2 Editing

After an animated character is authored, the user can edit it by taping on its trajectory for selection, and press the "Edit" button to enter the editing mode (Figure 6(b)). It then shows the timeline with motion bars for the selected character. The animated scene and the timeline are synchronized. Unlike traditional animation software for supporting multi-track timelines, our system uses a single-track timeline each time for a selected character, due to the limited mobile screen space. The user can perform the traditional pan and pinch gestures to control the translation and zoom level of the timeline, respectively. Motion bars with different colors are displayed in the timeline. Each motion bar, with an associated motion type (represented by the color of the bars) and duration (represented by the length of the bars), corresponds to one animation segment at a certain time. The user is able to edit the selected motion bar (i.e., pointing by red arrow) by using 4 different core functions (see below), as illustrated in Figure 7. These functions include:

Motion switch. Choose a new desired motion type to replace the one in current motion bars using picker.

[1] www.mixamo.com

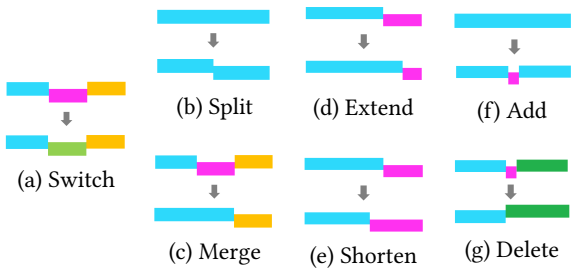
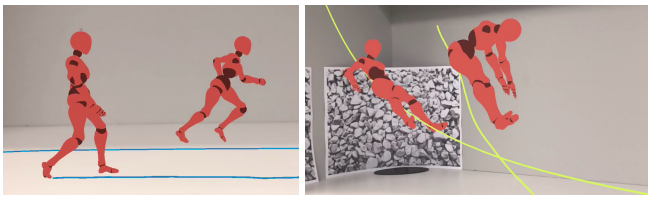


Fig. 7. Illustration for various operations supported in the editing mode. Here different colors represent different motion types.



(a) Walk/Run (b) Slide

Fig. 8. Our system automatically chooses different animation variations for the same motions according to the motion parameters (e.g., speed) of the trajectories.

Merge & split. To merge two animation segments into one, or split the currently selected segment into two.

Extend & Shorten. To change the duration of a selected animation segment.

Add & Delete. To add a new or delete the current motion type in between the animation segments. It is useful for adding character motion without gesture supported.

Beside the motion bars, our system also provides a simple interface to refine the trajectory. Users can move the endpoint of a selected animation segment along the three Cartesian coordinate axes by six buttons (two directions for each axis), or rotate it along the Y-axis (gravity direction) with a slider. The whole AR world coordinate system can also be moved or rotated if no character is selected. The axis and the center of the AR world are displayed in our interface.

4.3 Motion Synthesis

The animated virtual character(s) will be displayed on the mobile screen in the AR world directly. For each motion type supported by our system, we have prepared a pre-defined 3D character animation. When the user plays the recorded animation, our system directly plays the corresponding animation clips with respect to the current motion types using the native rendering API provided by iOS, and then translates and/or rotates each animated character along its trajectory. To make the animation fit the trajectory well, we loop the clips for continuous motions, and normalize the time for discrete motions. To get a smooth animation effect, we blend two different animation clips by explicitly setting the fade in/out duration, provided by the rendering API, to 0.3 seconds for all animations. Since the trajectory can be affected by the movement of performed motion gestures, we perform a mean filter on the trajectory's positions, orientation, and timestamp to reduce the local displacements caused by the performed motion gestures.

Besides simple playback of character motions, our system also supports automatic control of their motion parameters (e.g., speed of walk, height of jump). This enables our system to automatically choose proper animation variations (Figure 8) according to the motion information associated with the trajectory. For instance, if a user performs the "walk" gesture but moves the device quickly, our system automatically plays the "run" animation instead of "walk".

Finally, thanks to the portability of mobile AR, users can freely move the device to preview the created animations from different viewpoints. It enables users to easily create "movie-like" camera movement and record interesting animated scenes (Figure 11).

5 GESTURE CLASSIFICATION

The core contribution of our system is to utilize 6-DoF user-defined motion gestures as our direct input method. To this end, we employ an SVM (Support Vector Machine) based learning approach to recognize the supported gestures in our system.

5.1 Data Collection

We use the IMU data provided by ARKit API associated with motion gestures. In particular, the API provides the position data $p_t = (x, y, z)$ and the orientation data in a quaternion form $q_t = (x, y, z, w)$ at any time t. A gesture data is a continuous motion sequence consisting of both the position and orientation data (i.e., $[p_{t_0}, ..., p_{t_1}]$ and $[q_{t_0}, ..., q_{t_1}]$, where t_0 and t_1 represent the starting and ending moments of the gesture respectively).

To train a general SVM model for predicting motion gestures of all the users, we recruited five participants who were different from the users participated in our final usability study, collecting their data for training. We asked each participant to perform each type of motion gestures for five times, and manually select the parts corresponding to the gestures as the training data.

5.2 SVM-based Gesture Classification

To input the collected data into SVM for training, we first need to extract valid and reasonable features, which is challenging. Fortunately, the findings in Section 3.3 provide very useful hints.

The first important hint is from the *movement* taxonomy. It is observed that motion gestures involve the changes of position or orientation, or both position and orientation. It motivated us to use the position change and the global orientation change as our features. We sample at the starting points, middle points, and ending points of the gesture data. Then we calculate the displacement along the vertical and horizontal axes of these points, and the global angular change of device heading. The next hint is from the *abstraction* taxonomy. Users tend to imitate the orientation of action-line or local movement using a rotational gesture. Thus, we incorporate the device's orientation into our feature to distinguish those gestures. In particular, we decompose a unit vector along Y-axis (i.e., the roll axis) of the local device space into three global axes, e.g., formed by the direction of the gravity G, the device's horizontal heading

Table 4. The 24-dimensional features used for training our SVM.

# Dim.	Features
2	Vertical displacement in the first half and second half of a trajectory.
1	Horizontal displacement.
3	Decomposed y-direction along G at the first, middle, last point.
3	Decomposed y-direction along H at the first, middle, last point.
3	Decomposed y-direction along H' at the first, middle, last point.
2	Global roll change during the first half and second half of trajectory.
2	Global pitch change during the first half and second half of trajectory.
2	Global yaw change during the first half and second half of trajectory.
3	Sine wave amplitude of the angular movement along X/Y/Z-axis.
3	Sine wave offset of the angular movement along X/Y/Z-axis.

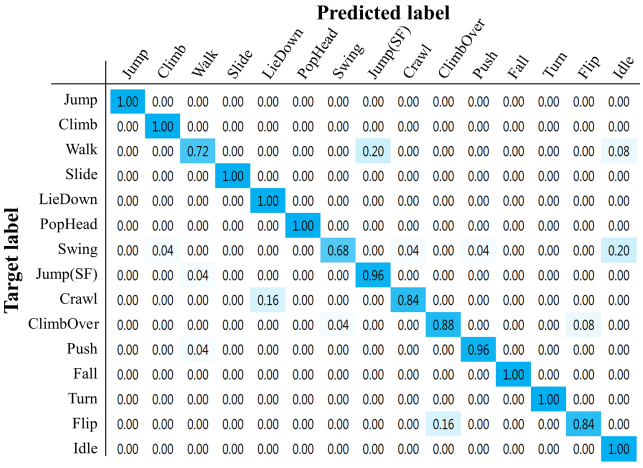


Fig. 9. The normalized confusion matrix of prediction results (X-axis) and the ground truth labels (Y-axis). The values of each row sum up to 1.0. "Jump (SF)" and "Turn" refer to "Jump with switching feet" and "Turn a circle", respectively.

direction H, and the other horizontal axis H' that is perpendicular to both G and H. The coefficients of the decomposed vector along these axes formed three features.

Finally, to explicitly handle continuous gestures, we individually fit sine waves to the global angular movement of the local X-, Y-, and Z-axis of the device. The amplitude and the offset values of the sine waves are used as our additional features. The same gestures might be performed differently each time, for robustness, the frequency and the period of the fitted sine waves are ignored. Using the above features, we form a 24-dimensional feature vector and train an SVM using these features. For easy reference we give a complete list of the features in Table 4.

5.3 Prediction

For prediction, we apply the same data processing procedure to the recorded raw data, except for manual segmentation of individual parts for different gestures. For an interactive system, we prefer to have a fully automatic method for gesture classification. To this end, we automatically trim a continuous motion sequence by using a method similar to Spatial Motion Doodles [Garcia et al. 2019]: we segment the sequence depending on whether the sum of Euclidean velocity and angular velocity (i.e., changes in quaternion space) is

Fig. 10. *ARAnimator* was used in various scenes for in-situ animation creation.

below a certain threshold (3cm/s in our current implementation). We use this way to automatically segment the motion sequence into parts for further prediction. It works well when there is a sufficient delay between the performance of different motions gestures. However, this segmentation method might also return false positive results. For instance, the velocity at the highest point of a jump (cyan curve in Figure 3(b)) could be close to zero, possibly leading to a cut at this point. To address this issue, before training, we search for any possible false positive cases, and generate a new gesture for each possible segment (e.g., decompose a jump action to jump_up and jump_down). The detected partial motions are combined as the original motions in a post-processing step after prediction.

6 RESULTS

We have implemented our *ARAnimator* prototype as a mobile app based on Apple ARKit. For the results below, we use iPhone devices. The gesture classification is implemented with Python on a server for easy implementation. A wireless network with HTTP request is used to connect our mobile app with the server.

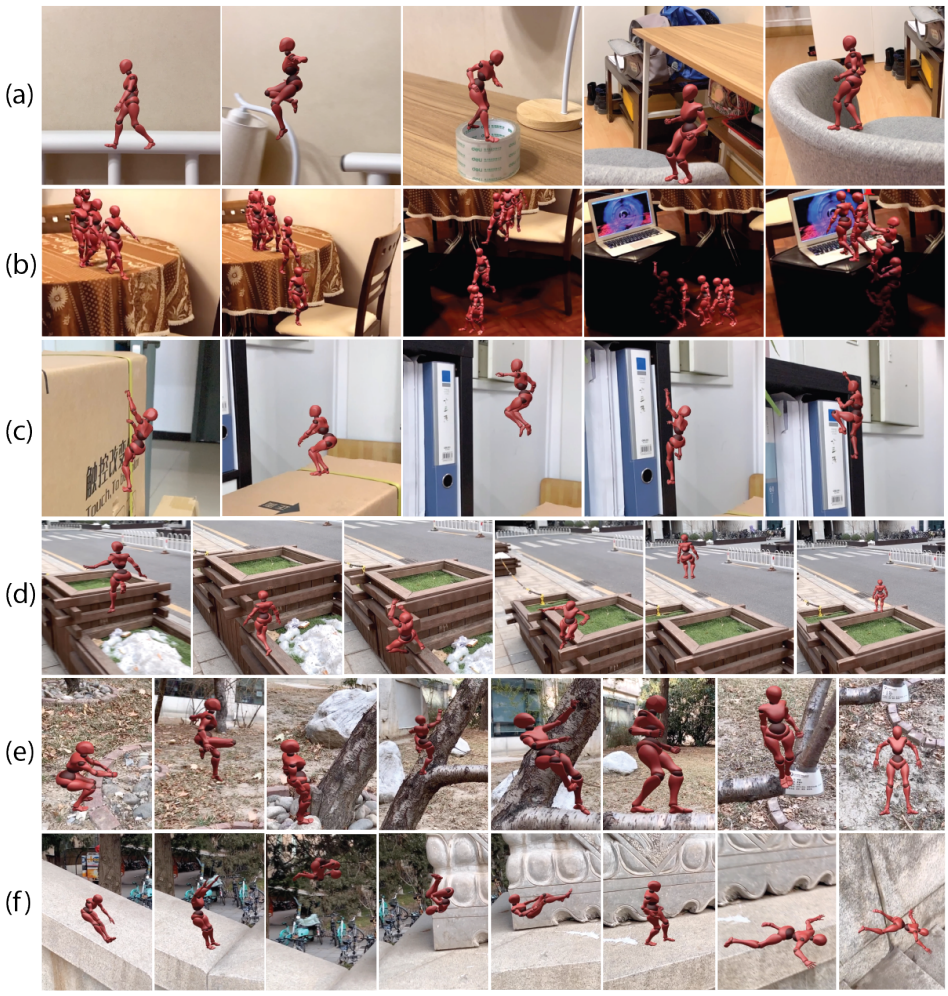
6.1 Quantitative Evaluation on Gesture Classification

We conducted a pilot study to evaluate the accuracy of our SVM-based gesture classifier. We performed leave-one-out cross validation to evaluate our classifier. In particular, we selected the data from one user for testing, and used the data from four remaining users for training our SVM. Then, we performed the same testing for every user and calculated all the accuracy. The total classification accuracy of the trained SVM ranged from 89.33% to 96.00% (M: 94.13%, SD: 2.76%) among different users.

In Figure 9 we visualize the normalized confusion matrix. From the diagonal values of the confusion matrix we can see that most of the gestures were correctly classified. For the cases of confusing gestures, "Swing" is the one of the most erroneous cases as reflected in Figure 9: our SVM tended to classify it as "Idle" (20%).

6.2 Usability Study

To evaluate the usability and effectiveness of *ARAnimator*, we ran a small-scale usability study by inviting four university students to create free-form in-situ 3D character animations using our system. All of them were non-professional animation creators (i.e., belong to our target user group), but had interests in creating animations using simple approaches. At the beginning of the study, each of them was given an iPhone XS and a 10-minute tutorial on our system, including how to perform individual gestures. They designed short



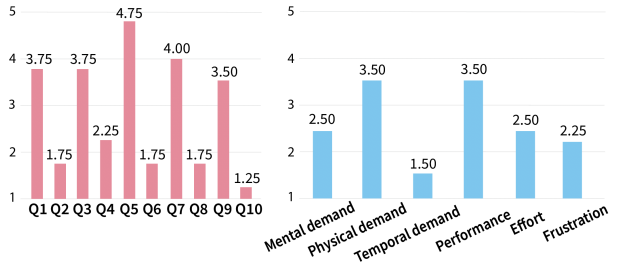
Motion	Accuracy	Time	Operation
walk jump circle jump jump sf	100%	51s	trajectory edit
walk jump jump walk climb	80%	120s	motion switch motion split trajectory edit
climb jump climb jump	100%	32s	/
jump walk climb circle jump	80%	40s	motion switch
jump climb jump sf slide	75%	61s	motion switch trajectory edit
slide forward flip walk fall	100%	32s	/

Fig. 11. A gallery of in-situ 3D character animation results produced by the four test users of our system. Each row (from left to right) presents a sequence of images from each animation. The statistics of the results can be found in the right table, where "jump sf" refers to "jump with switching feet", and "time" refers to the total time to create an animation including the authoring and editing time. Please refer to the accompanying video for the animation effects.

stories of a single or multiple characters containing multiple motions interacting with physical objects in various indoor and outdoor scenes (Figure 10). Then they utilized *ARAnimator* to create desired animated AR scenes. To reduce the tracking errors of AR in some scenes, papers with rich features were inserted into some of the surrounding environments. Figure 11 shows sample frames of the created representative animation results. At the end of the study, each participant was asked to fill in a questionnaire of five-point System Usability Scale (SUS, 1 = strongly disagree to 5 = strongly agree) and NASA Task Load Index (NASA-TLX, 1 = very low to 5 = very high) to evaluate the usability and perceived workload of *ARAnimator*, respectively.

The preliminary analysis on the process of animation creation demonstrates *ARAnimator* is a very useful, efficient and expressive tool. Four participants created in total 13 animations, each of which contains 2-8 motion types and 1-6 characters. These animations cover animated characters interacting with various objects (e.g., desks, doors, stairs, trees) in both indoor and outdoor scenarios. The mean classification accuracy over all the animations is 81.77% (SD: 21.20%), meaning that users could get correct results after performing motion gestures in most cases without further motion editing. The average creation time is 69.21s including 33.57s for authoring and 35.64s for previewing and editing (4.93s for trajectory editing and 30.71s for motion editing). On average 0.85 motion switching, 0.36 motion splitting and 0.07 trajectory editing operations were used per animation. It shows that users could quickly create and edit the animations with a few operations to get desired results.

In general, all the participants appreciated *ARAnimator*. P1 said that "*the system allows for creating fascinating animations easily*". P4 found that "*it's really cool to animate characters in this way*". The motion gesture based interaction in *ARAnimator* was also appreciated by all the participants. In fact, during the tutorial session,



(a) SUS (b) NASA-TLX

Fig. 12. Mean scores of SUS and NASA-TLX in a 5-point scale. Please refer to the appendix for the detailed SUS questions. For the questions with the odd numbers, the higher the SUS scores, the better; for the rest of the questions, the lower the SUS scores, the better.

we showed each motion gesture to the participants only once, and found that they could understand and memorize the gestures easily and quickly. P2 mentioned that "*it does not need to remember the gestures intentionally. Once I hold the phone I can naturally perform it without thinking too much*". P3 also said "*it is very intuitive, just like playing with a doll*".

The participants also reported that the editing function was user friendly. P1 commented that "*although initial (motion classification) results are sometimes wrong, I can fix them quickly via the provided editing interface*". The participants were overall satisfied with their results and appreciated the usefulness of our system. P1 mentioned that "*the animation results are nice. It is simple but interesting".* P1 and P4 both thought "*the character fits the objects well*". With some experience in animation creation before, P3 said that "*it may be very useful to do quick prototyping for animation designers*".

For the subjective measurements, we found the overall SUS score for all 10 questions was 80 on average (SD: 9.13), out of a scale of 100, indicating the good usability of our system. In Figure 12(a), we show the mean scores for all the individual SUS questions. From the feedbacks of the participants for Q2, Q3, Q4, Q7, Q8 and Q10, we can conclude our system was easy to use and intuitive, at least for this group of participants. From Q5 and Q6, the well-designed functions and interface of our system were also recognized by the participants. From the NASA-TLX results shown in Figure 12(b), we found that the mental demand and effort required by our system were moderate. It implies that our system does not require users to pay a lot of concentration and efforts when using our system. For temporal demand, the participants also believed the pace of our animation creation task was appropriate, since our system provided them with freeform time controlling. The participants were also satisfied with their animation results reflected by the high performance score, especially when they watched the real-time results from different views. Meanwhile, the physical load and frustration scores were not very low, possibly due to the requirement of body movement during animation creation, and occasional drifting caused by unstable tracking of ARKit.

7 CONCLUSION AND DISCUSSION

In this paper, we have presented a set of character motions and a corresponding set of motion gestures derived from two guessability studies. Based on the findings, we developed a prototype of *ARAnimator* with mobile AR to easily create 3D character animations closely interacting with real environments. The usability of our system has been demonstrated by our pilot study. Below we will discuss the limitations and other issues of *ARAnimator*.

7.1 Limitations

First, unlike the indirect animation tools [Anderegg et al. 2018; Thorne et al. 2004], our direct approach is more intuitive but applicable to small-size environments only (i.e., reachable by arms). In the future, it will be interesting to integrate our direct interface together with indirect interfaces (e.g., relative drawing [Kwan and Fu 2019]) for a more complete system applicable to both small-size and large-scale scenarios.

Second, our *ARAnimator* highly depends on the quality of motion tracking of ARKit, which is scene dependent. To suppress the tracking errors, we recommend to perform in-situ animation in the environments with rich visual features, or insert objects with rich features to surrounding environments (see the inserted paper in Figure 8(b)).

Third, in our system, users need to move a mobile device and perform motion gestures simultaneously. This leads to some degree of distortion on the desired trajectories of a character. Although we applied the mean filter to suppress it, the distortion cannot be removed completely and the remaining distortion might affect the visual quality of animated scenes. In the future, we will investigate more advanced curve smoothing methods to address this issue.

Fourth, our current usability test is limited to a small group of participants (i.e., 4 subjects). We are interested in conducting a larger-scale user study to more thoroughly evaluate our system. Besides, we noticed that the participant with some previous experience in animation creation proposed our system for quick prototyping, which is a valuable and potential insight. However, whether our proposed tool is beneficial to professional animators needs more investigation in the future.

Finally, character animation in mobile AR has been largely unexplored. There are still many challenging issues to be resolved. One important direction might be to improve the matching between characters and environments to address artifacts exhibited in our animation results like foot sliding. Using advanced rendering engines or techniques (e.g., prepare additional transition clips, searching for the optimal transition points) can be a possible solution for addressing unrealistic transitions between animation clips. We could also enhance our system by adding interactions to real objects (e.g., move or destroy them by the animated character), though this is challenging due to the requirement for real-time object extraction and manipulation in 3D AR space.

7.2 Discussion

GUI vs. Motion Gestures. It is possible to make the editing interface in Section 4.2 play a more important role and at the same time

reduce the role of our interface based on motion gestures. This potentially results in a two-stage approach: first sketch a 3D moving trajectory for a character using mobile AR (but without any motion gesture) and then purely employ a GUI-based interface similar to our editing interface for manually selecting the animations of the character. However, we would argue that this alternative solution and ours have their own pros and cons. GUIs are easier to use for first-time users, while gestures are quicker to perform at the cost of a steep learning curve and additional requirement for gesture recognition. In an ideal case with 100% gesture classification accuracy, our solution leads to a largely one-stage approach, making the editing step optional. Given the unique advantages of the two types of interfaces, we tend to believe they can and should co-exist in a unified system.

Scalability. Our system currently only supports a subset of the collected motions from the elicitation study (Section 3). Since the mapping from gestures to commonly used motions is intuitive, as reflected by the high agreement rates of our user-defined motion gestures, we believe that end users may introduce their own customized gestures to extend our system. However, newly added gestures might cause ambiguities in recognition. It thus may cause difficulties for incorporating a very large set of gestures in our system.

Classification. Although our *ARAnimator* currently only supports 15 motions, our preliminary test still found around 80% classification accuracy with all the 29 motions collected from the elicitation study. Even so, it is possible to improve the accuracy by exploring either new features for SVM or advanced deep learning based approaches. Unlike traditional gesture recognition works for invoking commands [Yoon et al. 2001; Zhou et al. 2009], which have rather fixed performing speeds and durations, our recognition task needs to handle gestures with varying speeds and durations depending on animation contents. This poses a special gesture recognition problem. Currently, we solve this challenge by simple segmentation before classification. Jointly addressing these two tasks might produce better classification results.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments and the user study participants for their time. This work was supported by a gift from Adobe and grants from the Research Grants Council of the HKSAR, China (No. CityU 11212119, 11204120), City University of Hong Kong (No. SRG 7005176), the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Project No. VRLAB2018C11), and CityU Shenzhen Research Institute.

REFERENCES

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*. ACM, 1–6.

Zhen Bai, Alan F Blackwell, and George Coulouris. 2015. Exploring expressive augmented reality: The FingAR puppet system for social pretend play. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1035–1044.

István Barakonyi and Dieter Schmalstieg. 2004. AR Puppet: Animated Agents in Augmented Reality. (2004), 35–42.

Istvan Barakonyi and Dieter Schmalstieg. 2006. Ubiquitous animated agents for augmented reality. In *The Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality*. 145–154.

Byungkuk Choi, Roger Blanco i Ribera, J. P. Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. 2016. SketchiMo: Sketch-Based Motion Editing for Articulated Characters. In *ACM Transactions on Graphics (TOG)*, Vol. 35. ACM, Article 146, 12 pages.

Loïc Ciccone, Martin Guay, Maurizio Nitti, and Robert W. Sumner. 2017. Authoring Motion Cycles. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. Association for Computing Machinery, New York, NY, USA, Article 8, 9 pages.

Loïc Ciccone, Cengiz Öztireli, and Robert W. Sumner. 2019. Tangent-Space Optimization for Interactive Animation Control. In *ACM Transactions on Graphics (TOG)*, Vol. 38. ACM, New York, NY, USA, Article 101, 10 pages.

Gokcen Cimen, Ye Yuan, Robert W Sumner, Stelian Coros, and Martin Guay. 2018. Interacting with Intelligent Characters in AR. *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)* 2017/2 (2018), 24–29.

Nem Khan Dim and Xiangshi Ren. 2014. Designing motion gesture interfaces in mobile phones for blind people. *Journal of Computer Science and technology* 29, 5 (2014), 812–824.

Mira Dontcheva, Gary Yngve, and Zoran Popović. 2003. Layered acting for character animation. In *ACM Transactions on Graphics (TOG)*, Vol. 22. 409–416.

Haegwang Eom, Byungkuk Choi, Kyungmin Cho, Sunjin Jung, Seokpyo Hong, and Junyong Noh. 2019. Synthesizing Character Animation with Smoothly Decomposed Motion Layers. *Computer Graphics Forum* 39, 1 (2019), 595–606.

Andreas Fender, Jörg Müller, and David Lindlbauer. 2015. Creature teacher: A performance-based animation system for creating cyclic movements. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*. ACM, 113–122.

Maxime Garcia, Rémi 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*. ACM, Article 10, 10 pages.

Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. 2016. Rig animation with a tangible and modular input device. In *ACM Transactions on Graphics (TOG)*, Vol. 35. ACM, Article 144, 11 pages.

Saikat Gupta, Sujin Jang, and Karthik Ramani. 2014. Puppetx: A framework for gestural interactions with user constructed playthings. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. ACM, 73–80.

Narukawa Hiroki, Natapon Pantuwong, and Masanori Sugimoto. 2012. A puppet interface for the development of an intuitive computer animation system. In *Proceedings of the 21st International Conference on Pattern Recognition*. IEEE, 3136–3139.

Hanyuool Kim, Issei Takahashi, Hiroki Yamamoto, Satoshi Maekawa, and Takeshi Naemura. 2014. Mario: Mid-air augmented reality interaction with objects. *Entertainment Computing* 5, 4 (2014), 233–241.

Yuki Koyama and Masataka Goto. 2018. OptiMo: Optimization-Guided Motion Editing for Keyframe Character Animation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Article 161, 12 pages.

Kin Chung Kwan and Hongbo Fu. 2019. Mobi3DSketch: 3D Sketching in Mobile AR. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Article 176, 11 pages.

Fabrizio Lamberti, Gianluca Paravati, Valentina Gatteschi, Alberto Cannavo, and Paolo Montuschi. 2017. Virtual character animation based on affordable motion capture and reconfigurable tangible interfaces. *IEEE transactions on visualization and computer graphics* 24, 5 (2017), 1742–1755.

Luis LEite and Veronica Orvalho. 2017. Mani-Pull-Action: Hand-based Digital Puppetry. In *Proceedings of the ACM on Human-Computer Interaction*, Vol. 1. ACM, Article 2, 16 pages.

Hui Liang, Jian Chang, Ismail K Kazmi, Jian J Zhang, and Peifeng Jiao. 2017. Hand gesture-based interactive puppetry system to assist storytelling for children. *The Visual Computer* 33, 4 (2017), 517–531.

Hai-Ning Liang, Cary Williams, Myron Semegen, Wolfgang Stuerzlinger, and Pourang Irani. 2012. User-defined surface+ motion gestures for 3d manipulation of objects at a distance through a mobile device. In *Proceedings of the 10th Asia Pacific Conference on Computer human interaction*. ACM, 299–308.

Noah Lockwood and Karan Singh. 2012. Finger walking: Motion editing with contact-based hand performance. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 43–52.

Noah Lockwood and Karan Singh. 2016. Gestural Motion Editing Using Mobile Devices. In *Proceedings of the 9th International Conference on Motion in Games*. ACM, 25–30.

Zhiqiang Luo, I-Ming Chen, Song Huat Yeo, Chih-Chung Lin, and Tsai-Yen Li. 2010. Building hand motion-based character animation: The case of puppetry. In *2010 International Conference on Cyberworlds*. IEEE, 46–52.

Sageev Oore, Demetri Terzopoulos, and Geoffrey Hinton. 2002. A desktop input device and interface for interactive 3d character animation. In *Graphics Interface*, Vol. 2. 133–140.

Yui Osato and Naoya Koizumi. 2018. Charrot: Pseudo-haptics with Mid-air CG Character and Small Robot. In *Proceedings of the Virtual Reality International Conference-Laval Virtual*. ACM, Article 18, 5 pages.

Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 197–206.

Mose Sakashita, Tatsuya Minagawa, Amy Koike, Ippei Suzuki, Keisuke Kawahara, and Yoichi Ochiai. 2017. You as a Puppet: Evaluation of Telepresence User Interface for Puppetry. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 217–228.

Yeongho Seol, Carol O'Sullivan, and Jehee Lee. 2013. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 213–221.

Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring user-defined back-of-device gestures for mobile devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 227–232.

Takaaki Shiratori, Moshe Mahler, Warren Trezevant, and Jessica K Hodgins. 2013. Expressing animated performances through puppeteering. In *2013 IEEE Symposium on 3D User Interfaces*. IEEE, 59–66.

Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion doodles: an interface for sketching character motion. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 424–431.

Amato Tsuji, Keita Ushida, and Qiu Chen. 2018. Real Time Animation of 3D Models with Finger Plays and Hand Shadow. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, 441–444.

Meng Wang, Kehua Lei, Zhichun Li, Haipeng Mi, and Yingqing Xu. 2018. TwistBlocks: Pluggable and Twistable Modular TUI for Armature Interaction in 3D Design. In *Proceedings of the 12th International Conference on Tangible, Embedded, and Embodied Interaction*. ACM, 19–26.

Jacob O Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A Myers. 2005. Maximizing the guessability of symbolic input. In *CHI'05 extended abstracts on Human Factors in Computing Systems*. ACM, 1869–1872.

Ho-Sub Yoon, Jung Soh, Younglae J Bae, and Hyun Seung Yang. 2001. Hand gesture recognition using combined features of location, angle and velocity. *Pattern recognition* 34, 7 (2001), 1491–1501.

Wataru Yoshizaki, Yuta Sugiura, Albert C Chiou, Sunao Hashimoto, Masahiko Inami, Takeo Igarashi, Yoshiaki Akazawa, Katsuaki Kawachi, Satoshi Kagami, and Masaaki Mochimaru. 2011. An actuated physical puppet as an input device for controlling a digital manikin. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 637–646.

Shengli Zhou, Qing Shan, Fei Fei, Wen J Li, Chung Ping Kwong, Patrick CK Wu, Bojun Meng, Christina KH Chan, and Jay YJ Liou. 2009. Gesture recognition for interactive controllers using MEMS motion sensors. In *2009 4th IEEE international conference on nano/micro engineered and molecular systems*. IEEE, 935–940.

Kening Zhu, Xiaojuan Ma, Haoyuan Chen, and Miaoyin Liang. 2017. Tripartite Effects: Exploring Users' Mental Model of Mobile Gestures under the Influence of Operation, Handheld Posture, and Interaction Space. *International Journal of Human–Computer Interaction* 33, 6 (2017), 443–459.

A DETAILED SUS QUESTIONS

For completeness, below we give a full list of SUS questions:

1. I think I would like to use this tool frequently.
2. I found the tool unnecessarily complex.
3. I thought the tool was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this tool were well integrated.
6. I thought there was too much inconsistency in this tool.
7. I would imagine that most people would learn to use this tool very quickly.
8. I found the tool very cumbersome to use.
9. I felt very confident using the tool.
10. I needed to learn a lot of things before I could get going with this tool.