WristSketcher: Creating 2D Dynamic Sketches in AR With a Sensing Wristband

Enting Ying\textsuperscript{a}, Tianyang Xiong\textsuperscript{a}, Gaoxiang Zhu\textsuperscript{b}, Ming Qiu\textsuperscript{a}, Yipeng Qin\textsuperscript{c}, and Shihui Guo\textsuperscript{a}

\textsuperscript{a}School of Informatics, Xiamen University, Xiamen, China; \textsuperscript{b}Hunan Huanan Optoelectronics (Group) Co, Ltd, Changde, China; \textsuperscript{c}School of Computer Science and Informatics, Cardiff University, Cardiff, UK

\textbf{ABSTRACT}
Restricted by the limited interaction area of native AR glasses, creating sketches is a challenge in it. Existing solutions attempt to use mobile devices (e.g., tablets) or mid-air hand gestures to expand the interactive spaces and as the 2D/3D sketching input interfaces for AR glasses. Between them, mobile devices allow for accurate sketching but are often heavy to carry. Sketching with bare hands is zero-burden but can be inaccurate due to arm instability. In addition, mid-air sketching can easily lead to social misunderstandings and its prolonged use can cause arm fatigue. In this work, we present WristSketcher, a new AR system based on a flexible sensing wristband that enables users to place multiple virtual plane canvases in the real environment and create 2D dynamic sketches based on them, featuring an almost zero-burden authoring model for accurate and comfortable sketch creation in real-world scenarios. Specifically, we streamlined the interaction space from the mid-air to the surface of a lightweight sensing wristband, and implemented AR sketching and associated interaction commands by developing a gesture recognition method based on the sensing pressure points. We designed a set of interactive gestures consisting of Long Press, Tap and Double Tap based on a heuristic study involving 26 participants. These gestures are correspondingly mapped to various command interactions using a combination of multi-touch and hotspots. Moreover, we endow our WristSketcher with the ability of animation creation, allowing it to create dynamic and expressive sketches. Experimental results demonstrate that our WristSketcher (i) recognizes users’ gesture interactions with a high accuracy of 95.9%; (ii) achieves higher sketching accuracy than Freehand sketching; (iii) achieves high user satisfaction in ease of use, usability and functionality; and (iv) shows innovation potentials in art creation, memory aids, and entertainment applications.

\textbf{KEYWORDS}
AR glasses; sketching; flexible touch sensor

1. Introduction
Sketching is a natural language to express users’ thoughts, feelings, and emotions. When used in an Augmented Reality (AR) environment, sketching is provided with a context of surrounding scenes. Compared to traditional creation methods, AR-based creation inspires their creativity and can thus be used as narrative tools for storytelling (Leiva et al., 2020), teaching aids (Kang et al., 2020; Suzuki et al., 2020), 3D design tools (Arora et al., 2018; Kwan & Fu, 2019), etc. Such a wide range of applications show the infinite expressive power of sketching in AR.

Early works on AR sketching are mostly created on smart phones or tablets, which are effective but not immersive (Canadas, 2020, 2022; GitHub, 2018; Kwan & Fu, 2019; Pham, 2017). In contrast, directly sketching with native AR glasses is immersive, but can be extremely troublesome by just using its touch bars, due to the limited interaction area. To address this issue, existing works resort to either gesture recognition and tracking techniques that allow for bare-hand sketching in the air (Dudley et al., 2018; Zhang, Zhu, et al., 2020), or mobile devices such as tablets (Arora et al., 2018; Gasques et al., 2019) as input agents. Between them, bare-hand sketching in 3D space is zero-burden but lacks physical support, which has been shown to be inaccurate (Arora et al., 2017), causing arm fatigue and social misunderstandings (Dobbelstein et al., 2017). In contrast, mobile devices allow for ergonomic and accurate sketching, but are heavy, limiting their application scenarios.

In this paper, we draw ideas from flexible electronics and propose WristSketcher, a novel system (Figure 1) based on a flexible sensing wristband for creating 2D dynamic sketches in AR. Our WristSketcher comprises a lightweight flexible film on a user’s wrist to detect the user’s touch pressure in real time, leading to an almost zero-burden authoring model for accurate and comfortable sketching in real-world scenarios. We also design a set of interactive gestures that is correspondingly mapped to various command interactions using a combination of multi-touch and hotspots based on a heuristic study and implement them with a pressure-based gesture recognition method. In addition, inspired by RealitySketch (Suzuki et al., 2020), we endow our WristSketcher with the ability of animation creation, allowing the creation of dynamic and more expressive sketches. Integrating the above, we implement a dynamic sketching application with diverse functions such as Recognition,
Draw, Erase, and Animation. Finally, we justify the effectiveness of our WristSketcher through (i) its high gesture recognition accuracy; (ii) its superiority in accuracy compared to Freehand sketching, that is one of the modes of mid-air bare-hand interaction, the input is made by tracking the position of the hand in real time; (iii) a positive user evaluation which demonstrates that our WristSketcher provides a pleasant and comfortable creative experience, and (iv) its innovation potentials in art creation, memory aids, and entertainment applications.

In summary, our main contributions include:

1. We present WristSketcher, a novel 2D sketch input agent for AR based on a flexible sensing wristband, which is lightweight and portable, giving users a pleasant and comfortable creative experience.

2. We design a set of interactive gestures that is correspondingly mapped to various command interactions using a combination of multi-touch and hotspots for our WristSketcher based on a heuristic study. They are intuitive, easy-to-remember, accurate, and efficient. We implement them with a pressure-based gesture recognition method.

3. We demonstrate the superiority of our WristSketcher with (i) a comparison with Freehand sketching on sketching accuracy; (ii) a user study, which shows that our WristSketcher can accurately recognize (~95.9% accuracy) and respond to users’ interaction by touching on its surface. In addition, we have received positive feedback from users in terms of ease of use, usability, and functionality of our WristSketcher. These feedback also show the innovation potential of our WristSketcher in art creation, memory aids, and entertainment applications.

2. Related work

2.1. Sketching input in AR/VR

2.1.1. Bare-hand
Thanks to the advance in deep learning, gesture recognition, and tracking are now well-established technologies and have been widely deployed in real-world applications. For example, instead of introducing external interaction tools, many AR/VR applications in modern mixed reality headphones (Mo et al., 2021) adopt a bare-hand interaction method, which is natural and expressive and can support sketching. This interaction method usually involves mapping different gestures to sketching commands (e.g., Draw, Move). It also includes the calibration of detected 3D coordinates of hands and fingertips, with the option to retain (Amores & Lanier, 2017; Dudley et al., 2018; Jang et al., 2014) or ignore (Zhang, Zhu, et al., 2020) depth information during the calibration process, in order to control the position of the brush. In the latter case, it involves creating a virtual plane in AR where the brush can move. Despite its efficiency and immersive user experience, bare-hand sketching in AR/VR can be inaccurate due to the lack of physical support (Arora et al., 2017; Machuca et al., 2018). Furthermore, it can lead to social misunderstandings when used in the public and its prolonged use can cause arm fatigue.

2.1.2. Pen (in-the-air)
Pens are popular input devices for both AR and VR (Pham & Stuerzlinger, 2019). Some works (Elsayed et al., 2020; Keefe et al., 2001; Wesche & Seidel, 2001) show the high expressiveness of pens in sketching using AR/VR headsets. To facilitate accurate drawing, the sizes, shapes, and weights of these pens are usually carefully designed (Elsayed et al., 2020; Pham & Stuerzlinger, 2019). However, since they are used in-the-air, pens also suffer from social misunderstandings and arm fatigue.

2.1.3. Mobile devices (positioning)
AR Development Kits like ARKit and ARCore have empowered mobile devices (e.g., mobile phones, tablets) with spatial awareness (Oufqir et al., 2020). Based on them, Just-a-Line (GitHub, 2018), ARDraw (Pham, 2017) and AR Draw Augmented Reality 3D (Canadas, 2020) all make use of the global position of a mobile phone for 3D sketching. In this way, the mobile devices are used both for display
and for sketching (positioning), and are thus not immersive. When used for positioning in the air, mobile devices suffer from social misunderstandings and arm fatigue as well.

2.1.4. Mobile device (touching)

To obtain an immersive experience, mobile devices, especially tablets, can be used together with AR/VR headsets where the headsets are used for display and the mobile devices are only used for sketch input. For example, Drey et al. (2020) defined a design space for 2D and 3D sketch interaction metaphors in VR. In addition to sketching directly in the space, users can also place a sketching surface of any size in the virtual environment through a tablet, and use a pen to draw sketches on it; Similarly, Arora et al. (2018) designed a hybrid sketching system that can map continuously created surfaces from an unorganized 3D curve collection into a 2D canvas in a flat panel for sketching. Another example is Gasques et al. (2019), which places the 2D sketches drawn on the tablet in the real environment. These works make use of the touch screens of the mobile devices for sketch input. Thanks to the physical support of the screen, these methods allow for more accurate sketching than above methods. However, these devices are heavy, making it inconvenient to carry around.

2.1.5. Wearable devices

Wearable devices are promising solutions for various real-world problems including AR sketching. For example, Qian et al. (2021) utilize a wearable device (camera) that can be worn on the fingertip to extend the field of view for mobile AR sketching. Jiang et al. (2021) use Manus VR gloves and extend the hand point positions to include fingertips to improve the accuracy of VR sketching. However, their gloves are only used as a physical interaction tool and their system still relies on optical perception and positioning.

In this work, we present WristSketcher to enable input of sketching in headsets. Different from the above input method for sketch creation, this is a novel 2D sketching system for AR based on a flexible sensing wristband. First of all, we capture the scene seen in the AR glasses and convert it into a virtual plane canvas. This canvas serves as the basis for creating a sketch. Second, we provide users with a portable interactive surface. This surface-based creation method will support users in creating more accurate AR sketches. In addition, compared to using other agents as input, our WristSketcher as a wearable device can be worn on the surface of a body, which provides convenience for going out to a certain extent. For example, users do not need to hold it in their hands when not creating, and the lightweight device feature can also give users a comfortable experience.

2.2. Dynamic sketching

Compared to static ones, dynamic sketches are those with animated effects and are thus more expressive. How to add such animations efficiently has been an important topic in human-computer interaction (HCI). Recently, Suzuki et al. (2020) define the two modes of dynamic sketching, Separated and Embedded, according to their design spaces, i.e., whether the sketches are embedded in real-world scenes. With this definition, (Kazi, Chevalier, Grossman, & Fitzmaurice, 2014; Kazi, Chevalier, Grossman, Zhao, et al., 2014; Su et al., 2018; Willett et al., 2018) follow the Separated mode and isolate dynamic sketching from the real world. (Leiva et al., 2020; Liu et al., 2020) proposed a dynamic sketching method for videos, by adding sketches to an input video and controlling their motion according to keyframes. Note that their methods still adopt the Separated mode as they rely on keyframes of pre-recorded videos rather than embedding sketches into real-world scenes in real-time. DoodleLens (Wolf, 2019) follows the Embedded mode, by converting sketched animation frames drawn on paper into digital content, placing them in real-world scenes, and animating them by quickly switching frames. RealitySketch (Suzuki et al., 2020) goes a step further in the Embedded mode, allowing drawn lines and defined prefab to interact with real-world events in real-time.

In this work, we follow the Embedded mode and endow our WristSketcher with the ability to create dynamic sketches with AR glasses. In addition, we carefully design its associated interaction commands to enable the creation of custom animations with rich options in a user-friendly way.

2.3. Interaction with flexible wearable devices

Flexible wearable devices typically consist of textiles, sensing units, and the data transmission and processing systems. Like clothing, such devices can be worn on various parts of a human body, such as the head (Hansen et al., 2006), hands (Glauser et al., 2019), wrists (Fang et al., 2023; Knibbe et al., 2021), thighs (Dobbelstein et al., 2017), feet (Zhang, Chen, et al., 2020), leading to various applications in motion analysis (Mokhlespour Esfahani & Nussbaum, 2018; Shi et al., 2019), interactive control (Olwal et al., 2018), healthcare (Leong et al., 2016), etc.

In previous works, flexible wearable devices complete the input of instructions by providing diverse input semantics. For example, Dobbelstein et al. (2017) and Heller et al. (Heller et al., 2014) implement fabric tactile input by embedding capacitive multi-touch sensing devices into fabrics and using the front pocket of the pants as a touch area. Dobbelstein et al. (2015) propose a touch belt for Google Glass input, whose large input surface allows users to interact in a subtle and unobtrusive way without raising their arms. GestureSleeve (Schneegass & Voit, 2016) is a wristband variant that supports the recognition of tap and swipe gestures. zPatch (Strohmeier et al., 2018) is an eTextile patch that enables hover, touch and pressure input via resistive and capacitive sensing, and is used to control music players, text input and game input. Handwriting Velcro (Fang et al., 2023) presents a text input method designed for AR glasses, utilizing a wristband for inputting text. However, it differs from this work in that it primarily emphasizes the accuracy of text input and recognition, rather than involving complex
interactive tasks. Despite these works, the use of wearable devices in AR sketching and their associated interaction commands is still an under-explored topic.

In this work, we explore the application scenario of dynamic sketch creation in augmented reality (AR) using a sensing wristband.

3. System overview
3.1. Prototype setup

3.1.1. WristSketcher
As illustrated in Figure 2(a), the proposed WristSketcher is composed of a flexible pressure film and a signal acquisition module (“Module” for short). The former is a sensing point array sensor made of polyester film with comprehensive mechanical properties, highly conductive materials, and nano-scale varistor materials, with a size of $8.36 \times 8.36 \text{ cm}^2$ and a thickness of $0.2 \text{ mm}$ [9] (Figure 2(b)). It is divided into a bottom flexible circuit pressure-sensitive layer and a top flexible circuit pressure-sensitive layer, which are pasted by double-sided tape to isolate the upper and lower sensing areas. When the sensing area is pressurized, the two pressure-sensitive layers are in contact with each other, and the output resistance of the channel varies with the position. The film has a total of 1936 sensing points, each with a size of $1.6 \times 1.6 \text{ cm}^2$ and arranged with a gap of $0.3 \text{ mm}$. The sensing threshold is $20 \text{ g}$, and it can respond within $10 \mu\text{s}$.

During the experiment, in order to avoid data interruption, we selected 40 sensing points in the horizontal direction and 40 in the vertical direction. The velcro is located on the back of the film (Figure 2(b)), and can help us attach the film to the surface of the retractable straps tied to the human body, so as to carry it with us. The signal acquisition module is a data collector with a serial interface and USB interface (Figure 2(b)), which can collect pressure information at a frame rate of 60 fps. In our current implementation, we use a USB interface to connect the collector to a computer equipped with a serial data processing program.

3.1.2. AR glasses
We use Rokid Glass 2 (Rokid Inc, 2020) (Figure 2(a)), an optically transparent head-mounted AR glasses launched by Rokid Corporation Ltd in 2020, to develop our system. This device has a 1280 $\times$ 720 resolution screen, is based on Android system, and supports USB-C connection. Due to the lack of a Simultaneous Localization and Mapping (SLAM) function in the current device, positioning sketch elements in real-world scenes is challenging. To address this issue, we use a method based on image recognition and tracking, which identifies useful contents using (relevant patches cropped from) the original images captured by the camera. Specifically, real-time detection and matching of feature points in the actual image against the template image’s feature point data in the database enable the successful integration and display of corresponding elements in the current scene.

3.1.3. Data processing unit
In this work, we use a USB interface to connect the collector to the computer (Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz 2.59 GHz, RAM 8.00 GB) equipped with a serial data processing and a recognition algorithm program. We then use socket communication to transmit the user’s interaction information to the AR glasses.

3.2. Sketching user interface
The software of our WristSketcher, implemented in Unity, can be divided into three modules: (i) AR recording and rendering, (ii) sketching interaction, (iii) animation creation.

3.2.1. Recording and rendering
The recording and rendering page is shown in Figure 3. Users can perform captured operations on the current page, and see the dynamic sketches combined with the real scene after the creation is completed. Specifically, Figure 3(a) shows the process of the user recording the current scene. Users can perform the action of tapping the Record in the
lower left corner through a simple gesture command to display a floating window. This window presents the scene captured from the user’s current perspective. To minimize interference from other features, users have the ability to interactively control the corner movement and frame the recognizable feature area. Once the user confirms the selection, the current scene is processed accordingly and identified. At the same time, a virtual canvas will be generated and placed in the current scene. Figure 3(b) shows the scene where the user’s dynamic sketch is displayed after completing the creation. When the identified feature points in the user’s perspective match the previously captured content, the dynamic sketch created will be rendered. This technology based on feature point recognition and rendering is implemented based on Vuforia.

3.2.2. Sketching interaction

The activation of the sketching interaction interface (Figure 4(a)) occurs after the user triggers the end of the capture operation. In this interface, the menu is hidden by default, except for Current Tool and Save and Back, to reduce the occlusion caused by the menu bar and thus provide a wider visual interaction area. The main menu (Figure 4(b)) shows the basic sketch creation tools: Move, Draw, Erase, Create a New Asset, Animation, and Delete. Users activate the menu and select menu items through gestures. Draw and Animation have corresponding sub-menus (Figures 4(c,d)). In its selected state, the user can call out the sub-menu and perform the current functional subtask by interacting with the sub-menu, as described below. Besides, Undo can be executed by a specific gesture so we deliberately remove it from the interface for a more concise view.

3.2.2.1. Functions of draw’s submenu. Draw focuses on providing brush functions. In the Draw, users can adjust brush thickness, color, and perform mirroring mode.

- Thickness. The control state of the thickness, displayed by a bar. Users can control it with a simple interaction on the device.
- Color Picker. The color picker (Figure 4(e)) is called up by a shortcut command. When the color picker is displayed, the user can move the pin (Figure 4(e)) to select a color. This color is reserved for the next stroke and works as the indicated color of the brush. Note that the color picker is hidden when sketching, allowing for a larger canvas.
- Mirror Brush. This function is used to assist in creating a symmetrical shape. When this property is checked, strokes drawn by the user on the canvas will be mirrored. For example, the windmill leaves in Figure 4(d) are drawn by using this function.
3.3. Functions of animation’s submenu

The animation component panel (Figure 4(d)), which enables users to create five types of animations. By selecting an animation item, the corresponding type of animation will be bound to the currently edited layer. Note that users can not only add, but also overlay animation styles to create more engaging and dynamic sketches. Details of the five types of animations are described below.

3.2.3. Animation creation

To create engaging and dynamic sketches, we draw ideas from BABA Is You (Oy, 2020), Kitty (Kazi, Chevalier, Grossman, & Fitzmaurice, 2014), Draco (Kazi, Chevalier, Grossman, Zhao, et al., 2014), PoseTween (Liu et al., 2020), and introduce five types of animations (Doodle, Frame, Emit, Rotate, and Move) into our WristSketcher.

3.2.3.1. Doodle. Doodle is a constantly dithering image effect. Users can make their sketches jitter (Figure 5) by simply binding Doodle to their input static sketches. Whereas achieving such an effect in a traditional animation pipeline may require a large number of hand-drawn frames, our Doodle requires only a static sketch, resulting in time savings.

3.2.3.2. Frame. Frame is a frame animation plug-in (Figure 6) to achieve the fast-page turning effect. Such frame-based animations are commonly used in anime storyboard production and other fields. When using Frame, users can create multiple frames, and animate them by fast switching. In addition, users can customize the frame rate.
3.2.3.3. Emit. Emit uses the currently created asset as a material and continuously emit it. It has three functions: spray area setting, motion trajectory setting, and gravity simulation. The process of creating an Emit animation is shown in Figure 7(a). The user first creates a line (the blue one) to set an ejection area, from which the object will be ejected with an initial velocity, and then defines the movement trajectory of the object by creating a straight line (the red one). For a straight line to guide the motion of an object, we further introduce “force.” Similar to those in mechanics, the straight line in Figure 7(b) can be viewed as a resultant force from the the horizontal and vertical forces. The final effect is shown in Figure 7(c). The raindrops follow the trajectory and spray area initially defined by the user to simulate the effect of raindrops being blown by the wind.

3.2.3.4. Rotate. Rotate generates rotation effects by continuously modifying the rotation properties of an object (Figure 8). By default, we set the rotation point to be the center of mass of an object. The user can also customize the positions of rotation point by moving the rotation point mark. The angle and time of rotation are controlled by two sliding bars, respectively.

3.2.3.5. Move. Binding Move to an asset allows it to move along a predetermined trajectory (Figure 9). In Move, users can set the trajectory and movement time of the object.

4. Study 1: Heuristic study

In interactive applications, it’s common to use menu system or corresponding shortcut gestures associated with specific functions for input. Bailly et al. (2016) have characterized the design space of menus. The standard procedure for command input in menu systems involves menu activation, item selection, item activation, and command execution (menu release). However, for our system, due to the separation of interaction and display, there is a lack of visual correspondence between the AR display and the gestures on the wristband. Therefore, employing one-to-one shortcut gestures to activate complex and unrelated menu items is not user-friendly. As a result, we have adopted menus as the primary input solution, complemented by quick commands. Furthermore, given the complexity and uniqueness of our system’s menus. Even within this menu-based interaction paradigm, we recognize the importance of an intuitive set of interaction designs.

In previous interaction prototypes, author-designed interaction methods were typically employed. However, these designs may be influenced by various factors, including technical implementation constraints and interaction psychology models Wickens et al. (2021), among others. Consequently, they may not fully represent user behaviors and real-world needs. Existing research suggests that user-customized designs are not only easier to learn and remember compared to predefined standard designs but also contribute to enhancing user operational efficiency, resulting in quicker task execution. Therefore, we conducted the following heuristic study to obtain a set of easily learnable and intuitive gestures for sketch input and interaction commands, helping users adapt more seamlessly to the new interaction modality and improve their efficiency in executing functions within AR glasses.
4.1. Participants and apparatus

We invited 26 participants (16 males, 10 females), aged 18–25 years ($\bar{x} = 22.73$, $SD = 1.85$) to participate in this experiment. One of them was left-handed, and 18 participants had a background in computer science. All participants had prior experience using 2D touch interfaces in their daily lives. For the experiment, we utilized a flexible touch-pad device (Figure 2(a)). To encourage participants’ creativity, we disabled pressure sensitivity, allowing them to simulate interactive tasks on WristSketcher without immediate feedback. All participants voluntarily signed an informed consent form before the experiment, and upon completion of the experiment, they were paid for their participation.

4.2. Task and procedure

We asked participants to design interactions for a multi-level menu system as well as specific gestures for particular functions. We allowed them to make revisions after answering all the questions to ensure that there were no conflicts between their interaction designs. Additionally, we requested participants to provide their rationale for the designs. The design activities were conducted independently among the participants without communication. We posed the following seven questions to the participants:

- Q1: Please design a gesture command to activate the main menu.
- Q2: Please design a gesture command to activate the secondary menu.
- Q3: Please design a gesture command to activate the tertiary menu.
- Q4: Please design a gesture command to navigate menu items.
- Q5: Please design a gesture command to release the menu.
- Q6: Please design a gesture command as a confirmation after the operation.
- Q7: Please design a gesture command as an undo operation.

Q1–Q5 involve the multi-level menu system, while Q6 and Q7 mainly pertain to discrete operations outside the menu system, such as confirming adjustments to brush size and undoing actions during the sketching process.

4.3. Data collection and analysis

Figure 10 shows the touch gestures designed by the participants for the seven questions above. In addition to the gestures commonly used in 2D touch devices, including Tap, Long Press, and Slide, the participants also designed some custom gestures, such as Five-Finger Pinch and Five-Finger Open. Considering that among the seven questions, there is similarity and continuity within Q1–Q5, we have chosen to analyze them by categorizing them as follows: (1) Menu activation (Q1–Q3); (2) Menu item operation (Q4–Q5); (3) Confirmation (Q6); and (4) Undo (Q7). And conduct interactive design based on the “minimum” principle.

4.3.1. Menu activation

Figure 11(a) presents a summary of the selected gesture types for these types of questions. Users exhibited a strong preference for using Long Press (38%) for this operation. This choice is primarily attributed to two advantages over Tap (32%): (i) it reduces the impact of accidental touches, and (ii) it avoids confusion between Tap and drawing small dots. Therefore, we have opted for Long Press as the interaction method for menu activation. Furthermore, to achieve finer-grained interaction gestures for activating different menus, we compared Long Press across three questions and found that users proposed various approaches, including multi-touch interactions, hot-spots, or a combination of both. For instance, P2 associated the four corners of the pressure pad with different menus, activated by One-Finger Long Press in the corresponding area, while P16 distinguished between second-level and third-level menus using One- and Two-Finger Long Press. In our interaction design, we ultimately adopted a combination of these approaches. This decision was based on the benefits of forming hot-spots corresponding to menu positions, aiding user memorization, and the intuitiveness of multi-touch interactions for menus located in the same position. Specifically, we set menu activation for the main menu as One-Finger Left Long Press, second-level menu activation as Two-Finger Left Long Press, and third-level menu activation as One-Finger Right Long Press.

4.3.2. Menu item operation

Menu item operation consists of two parts: one is the selection of menu items after activating the menu, and the other is confirming the menu item and releasing the menu. As shown in Figure 11(b), when the menu is activated, the majority of participants prefer to use the Long Press (42%) to switch menu items. Observing a similar pattern from the design of the menu activation, we believe that one major reason for this situation is the participants’ inherited habits and preferences for menu activation. Interestingly, based on their feedback, some participants who chose other options in Q1–Q3 switched to Long Press because they felt that automatic menu item switching reduced the interaction burden. Regarding releasing the menu, participants tend to have a preference for “None,” meaning they prefer issuing a release command when a persistent state change (e.g., Long Press). This allows them to remember fewer gestures. In our final design, we adopted this approach by associating menu item switching with the activation of the corresponding menu (Hold On) and using the “release” action Cancel Long Press as a signal to confirm closing the menu.

4.3.3. Confirmation

The confirmation operation takes place during the attribute modification phase and the end of sketching.
As illustrated in Figure 11(c), participants exhibited similar preferences for this confirmation interaction gesture: One-Finger Double Tap, which is an intuitive confirmation gesture. Therefore, we adopted this design as the universal confirmation interaction for all actions within the system.

4.3.2. Undo

Undo serves as a complement to Erase and is frequently employed during the sketching process. Therefore, we considered establishing it as a global interaction gesture for users to use while creating sketches. Figure 11(d) illustrates the design preferences of participants for the Undo operation, with participants showing a preference for using the Tap to execute undo actions. Taking inspiration from the digital illustration software Procreate (Ltd, 2022), in our final design, we employ the Two-Finger Tap gesture for the Undo function.

Finally, we have established a set of interaction gestures consisting of Long Press, Tap and Double Tap. These gestures are correspondingly mapped to various command

---

**Figure 10.** The touch gestures designed by the 26 participants for the 7 questions in Section 4.2. The point in the rectangle indicates the position of the first touch.
interactions using a combination of multi-touch and hot-spots. The final design of the correspondence between our gestures and interaction commands is presented in Table 1.

## 5. Implementation techniques

### 5.1. Touch point recognition

Figure 12 illustrates the gesture recognition technique used in our WristSketcher.

#### 5.1.1. Data acquisition and preprocessing

Our system reads the sensor data of each frame from the serial port of our WristSketcher in real-time, and converts the hexadecimal data into a $40 \times 40$ matrix. Since the raw data can be noisy, we also apply a threshold followed by a two-dimensional median filter (George et al., 2018) to mitigate the impact of noise.

#### 5.1.2. Touch point and no. of touching fingers

We employ the one-scan connected-component analysis (CCA) algorithm (He et al., 2017), which uses a randomly-selected pressed position as a seed, and then traverses its eight neighborhoods, identifying other points pressed by the same finger if the neighbour has a pressure value greater than half of it. We repeat this process, treating each identified neighbour as a new seed until there are no new seeds, and marking all identified points as having been pressed by the same finger. Note that we discard all identified points if their number is less than five to reduce the impact of noise. For all points pressed by a single finger, we define its touch point as the one with the highest pressure value. Then, to reliably identify the number of touching fingers, we define it

### 5.2. Touch point and no. of touching fingers

![Figure 12](image-url) Illustration of touch point recognition. “Fr”: Frame, “Po”: Point. $n = 3$ in our experiments.
as the number of touching fingers that appears with the highest frequency in consecutive frames. We further calibrate the touch point as the average of those in these highest-frequency frames. The final recognition results are transmitted to the AR glasses through Socket communication. We conducted a test to investigate whether our touch point detection method would greatly affect the performance. The test results show that our gesture recognition runs in real-time at a rate of approximately 31 FPS.

5.2. Gesture recognition

The gestures used by our WristSketcher for interaction commands can be summarized into three categories: Tap, Double Tap, and Long Press. These three types of gestures can be distinguished according to their interval, duration, and type of leading gesture. Specifically, we recognize a gesture as (i) Tap if its time interval is less than 0.15s and there is not a second tap in 0.5s; (ii) Double Tap if there are two valid taps within 0.5s; (iii) Long Press if the touch time is longer than 1s.

6. Study 2: User evaluation

To investigate the feasibility and usability of our WristSketcher, we conducted a user evaluation experiment on the accuracy of gesture recognition accuracy and the experience of animation creation. Among them, for the evaluation of gesture recognition accuracy, we compared it with Leap Motion. We also compared our WristSketcher with Freehand sketching, one of the modes of mid-air bare-hand interaction, its input is made by tracking the position of the hand in real time, to demonstrate its effectiveness as a new input mode for AR glasses.

6.1. Participants

We recruited 10 participants (7 males, 3 females) aged 21–27 years ($\bar{x} = 23.1, SD = 1.79$) for this experiment. All participants had no experience or relevant expertise in AR glasses.

6.2. Task and procedures

6.2.1. Gesture recognition accuracy

In this task, we compared WristSketcher and Leap Motion. For fingers recognition and interactive gestures with Leap Motion, we utilized two services it offers: Leap Motion Control Service and TouchFree Service, obtaining relevant information based on these services. Before the task started, we first demonstrated twelve different types of gestures to participants: One-Finger Left Tap, One-Finger Right Tap, One-Finger Top Tap, One-Finger Bottom Tap, One-Finger Left Long Press, One-Finger Right Long Press, One-Finger Top Long Press, One-Finger Bottom Long Press, One-Finger Double Tap, Two-Finger Tap, Two-Finger Left Long Press, and Two-Finger Right Press. This encompassed how these gestures are recognized and triggered in both WristSketcher and Leap Motion. After explaining the task, users were given 5 min for practice. Afterwards, participants were asked to perform each type of gesture 10 times. The order of gesture types is randomized.

6.2.2. Pre-defined and freeform creation

Before the start of this task, we provided participants with a formal introduction to our system. This included an overview of the system’s features, the mapping of the 12 interactive gestures and interactive functionalities conducted in Task 1, and a comprehensive workflow for creating 2D dynamic sketches within a pair of AR glasses. Following the introduction, participants were given 20 minutes to practice using the AR glasses and our WristSketcher. This practice time aimed to mitigate the impact of insufficient familiarity. Due to time constraints, we only asked them to complete two pre-defined animation creation tasks: creating a twinkling star and a raining effect. These tasks are the most representative and time-consuming ones in our system. However, there was no specified time limit for each task. Throughout the creative process, the facilitator recorded observations but did not intervene. After these pre-defined tasks, the participants were encouraged to use the tools freely to create. Finally, after completing all tasks, we invited the participants to rate our system on a 5-point Likert scale in terms of ease of use, usability, and functionality.

6.2.3. Comparison with freehand sketching

We followed the common practice in sketch interface evaluation (Dudley et al., 2018; Elsayed et al., 2020; Wiese et al., 2010; Zhang, Zhu, et al., 2020) and asked the participants to complete the drawing of three 2D basic templates (Rectangle, Triangle, and Circle) in AR glasses in two input modes:

- **WristSketcher.** The participants wore the WristSketcher on their arms (Figure 13(a)) and input with gestures on its surface.
- **Freehand.** The participants used bare hands to draw continuously in the air (Figure 13(b)). It should be emphasized that we ignore the depth information and project the finger position information on the flat canvas.

During the experiment, the participants were asked to draw according to all the three templates that were presented three times each in a random order. The drawing time and results were recorded and used in the comparison of the following two metrics:

- **Completion Time.** This is the time spent between the first and last points of a drawing.
- **Drawing Error.** This is defined as the average minimum distance between the drawing and its corresponding template:

$$DE = \frac{\sum_{i=1}^{N} ||d_i - t_i||}{N}, d_i \in D, t_i \in T,$$  \hspace{1cm} (1)

Where $N$ is the size of the sample set $D$ of the drawing and that of the sample set $T$ of $D$’s corresponding template, $t_i$ is the nearest point to $d_i$ in $T$. 

6.3. Results

The test results show that our gesture recognition runs in real-time at a rate of approximately 31 FPS.
6.3. Results and discussions

6.3.1. Gesture recognition accuracy

We finally collected 2400 pieces of data (10 participants × 120 repetitions × 2 types) for the task, and defined the gesture recognition accuracy as the proportion of correctly identified gestures out of the total number of gestures. The confusion matrix for the gesture recognition experiment is showed in Figure 14. Figure 14(a) represents the gesture recognition accuracy for WristSketcher, while Figure 14(b) illustrates the accuracy for Leap Motion. The experimental results demonstrate that our WristSketcher achieved an average recognition accuracy of 95.9% across all gestures, surpassing the 91.4% average recognition accuracy achieved using Leap Motion. This suggests that our designed interaction method and implementation are capable of accurately responding to user interactions, minimizing erroneous functional triggers. However, compared to the zero-recognition error rate for the upper right corner showed in Figure 14(b), we identified an issue with inaccurate region recognition in our device. This is primarily attributed to the lack of clear region boundaries in our interaction area and the displacement of pressure points caused by the gap between the touchpad and the forearm. In contrast, Leap Motion can clearly define regions using the cursor displayed in the
interface. Additionally, the recognition accuracy for Two-Finger Tap and One-Finger Double Tap was relatively low. We attribute this low accuracy to (i) the different pressure levels applied by the two fingers during Two-Finger Tap, occasionally resulting in insufficient touch pressure by some participants, causing the sensor to misinterpret the touch. As a result, participants’ Two-Finger operation is recognized as a One-Finger operation; (ii) One-Finger Double Tap recognition is based on the time interval between two consecutive clicks. Some participants were affected by the gap and occasionally failed to click again promptly, leading the system to misinterpret a One-Finger Double Tap as two One-Finger Tap. Despite the challenges posed by the gap and its impact on the recognition accuracy of these two gestures, the gesture recognition system still achieved a relatively high level of accuracy. In future work, we will address this issue.

6.3.2. Pre-defined and freeform creation
Our participants were able to complete the creation of both pre-defined dynamic sketches without assistance. From the time recorded, the participants spent an average of 22.42 min (Min = 16.93 min, Max = 27.21 min) on this task, with an average of 3.77 min (Min = 2.94 min, Max = 4.74 min) to create a twinkling star, and with an average of 3.83 min (Min = 3.14 min, Max = 5.01 min) to create a

Figure 15. A gallery of sketches created by our WristSketcher. The sketching times for each example are as follows: (a) 25.19 min; (b) 5.32 min; (c) 4.03 min; (d) 3.12 min; (e) 4.51 min; (f) 5.36 min; (g) 18.06 min; (h) 20.35 min. (c), (d), and (h) were created outdoors, while the rest were created indoors.
raining effect. This shows that with a short practice session, the users could quickly learn to use WristSketcher to create dynamic sketches.

Figure 15 shows the freeform sketches drawn by the participants, demonstrating the innovation potentials of our WristSketcher in art creation, memory aids, and entertainment applications. For example, Figure 15(a) is a dynamic pattern drawn on clothes. For the intention of this design, the participant mentioned: “My clothes are very simple and plain, and I want to design some patterns for my clothes that I like. Animate it to make it cute and fun.” Figure 15(c) is used as a memory aid for garbage types. The participant who drew it said, “The local garbage classification is implemented, but sometimes I cannot tell which trash can I should throw it into. By doing this, when I will throw the garbage in the future, just look at the patterns I have drawn. It is a lot easier.”

At the end of the experiment, most of the participants gave positive feedback on the ease of use, usability, and functionality of our WristSketcher. From the 5-point Likert scale (1 = strongly disagree 5 = strongly agree), the participants’ average score for the overall rating was 4.01. Specifically, the average score for ease of use was 4.10 (Min = 3.5, SD = 0.47), the average score for usability was 4.03 (Min = 3.5, SD = 0.34), and the mean score for functionality was 3.90 (Min = 3.4, SD = 0.30). In the supporting comments, the vast majority of the participants felt that the gesture design was straightforward and that it facilitated rapid interaction tasks. For example, P2 said, “I commonly use the gestures of the system, and I can quickly remember all the interaction gestures.” P5 said, “Gestures are linked to the interface, and I can think of what actions I need to perform by positioning user interface elements.” In terms of usability, P4 provided the lowest rating primarily due to experiencing arm fatigue during prolonged creative sessions while wearing the WristSketcher. This discomfort was a result of maintaining the upper arm at an elevated position for prolonged periods. We also sought feedback from other participants who reported that they did not perceive any significant weight or persistent fatigue during creative sessions. We observed that these users typically positioned their upper arms almost vertically and relied on a comfortable forearm position for their creative work. In summary, the majority of participants expressed acceptance and willingness to use the WristSketcher for dynamic sketching in AR glasses as: (i) our WristSketcher is lightweight and comfortable to wear; (ii) it offers a novel interactive experience for users; (iii) it can respond accurately and quickly to user interactions. For the functionality, although some participants pointed out areas for improvement, such as: P6: “It would be nice if you could add an interactive response with me.” and P8: “The waiting time is too uncomfortable after I miss an option, I have to wait for a round to reselect.”, most participants agreed that, overall, our WristSketcher provided an interesting creative support for dynamic sketching using AR glasses.

### 6.3.3. Comparison with Freehand sketching

Tables 2 and 3 show the qualitative and quantitative comparisons between our WristSketcher and Freehand respectively. The significance of quantitative results was analyzed by one-way ANOVA. It can be observed that: (i) for completion time, the difference between WristSketcher and Freehand is significant ($F=13.39, p<0.05$) that sketching with our WristSketcher consumes slightly more time due to its smaller interaction area; (ii) for drawing error, the difference between the two input methods is more significant ($F=31.45, p<0.001$) that our WristSketcher allows for more accurate sketching. We ascribe this to the observation that although the user can trace standard elements steadily at the beginning, their hands become unstable over time. Such a trade-off shows that our WristSketcher can be used as a new input mode for sketching with AR glasses.

### Table 3. Quantitative comparison between our WristSketcher and Freehand.

| Condition          | Completion Time (s) | Drawing Error (pixel) |
|--------------------|---------------------|-----------------------|
| WristSketcher      | 13.38 ± 1.77        | 5.14 ± 2.15           |
| Freehand           | 10.31 ± 1.98        | 7.16 ± 2.65           |

The images show the overlay sketches by all users.
7. Conclusion and future work

We have presented WristSketcher, a new input agent that allows users to create 2D dynamic sketches in AR glasses. Specifically, our WristSketcher makes up for the lack of precision caused by mid-air bare-hand sketching, as well as the portability and comfort that users need to consider when using mobile devices, by diverting AR sketch input to a lightweight and flexible sensing wristband. To implement such an input mode, we designed a set of interaction gestures through a heuristic study and its corresponding gesture recognition method. We also extend our WristSketcher with the ability of animation creation, allowing it to create dynamic sketches embedded in real-world scenes. Experimental results show that our WristSketcher (i) achieves a high gesture recognition accuracy of 95.9%; (ii) achieves higher accuracy compared to Freehand sketching; (iii) achieves high user satisfaction in ease of use, usability and functionality; and (iv) shows the innovation potential of our WristSketcher in art creation, memory aids, and entertainment applications.

7.1. Limitations and future work

Although our WristSketcher allows users to create dynamic sketches in real-world scenes, the animations are not responsive to user interactions. We hope to extend our WristSketcher and incorporate multimodal interaction technologies for responsive animations in our future work. In addition, due to the hardware of AR glasses, we cannot obtain depth information and cannot use the SLAM function. While the Vuforia-based rendering technique enables the positioning of sketches in the real world, it is based on feature point matching in a view and does not allow for multiple views of the sketch drawn. So the sketches currently created by our WristSketcher are essentially 2D images. We also hope to extend our WristSketcher for 3D dynamic sketching in future work.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Amores, J., & Lanier, J. (2017). Holoart: Painting with holograms in mixed reality. In Proceedings of the 2017 CHI conference extended abstracts on human factors in computing systems (pp. 421–424). Association for Computing Machinery. https://doi.org/10.1145/3027063.3050435

Arora, R., Habib Kazi, R., Grossman, T., Fitzmaurice, G., & Singh, K. (2018). Symbiosketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In Proceedings of the 2018 CHI conference on human factors in computing systems (pp. 1–15). Association for Computing Machinery. https://doi.org/10.1145/3173574.3173759

Arora, R., Kazi, R. H., Anderson, F., Grossman, T., Singh, K., & Fitzmaurice, G. (2017). Experimental evaluation of sketching on surfaces in VR. In Proceedings of the 2017 CHI conference on human factors in computing systems (pp. 5643–5654). Association for Computing Machinery. https://doi.org/10.1145/3025453.3025474

Bailly, G., Lecolinet, E., & Nigay, L. (2016). Visual menu techniques. ACM Computing Surveys, 49(4), 1–41. https://doi.org/10.1145/3002171

Dobbelstein, D., Hock, P., & Rukzio, E. (2015). Belt: An unobtrusive touch input device for head-worn displays. In Proceedings of the 33rd annual ACM conference on human factors in computing systems (pp. 2135–2138). Association for Computing Machinery. https://doi.org/10.1145/2702123.2702450

Dobbelstein, D., Winkler, C., Haas, G., & Rukzio, E. (2017). Pocketthumb: A wearable dual-sided touch interface for cursor-based control of smart-eyewear. The Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(2). https://doi.org/10.1145/3090055

Drey, T., Gugenheimer, J., Karlbauer, J., Milo, M., & Rukzio, E. (2020). Vrsketchin: Exploring the design space of pen and tablet interaction for 3D sketching in virtual reality. In Proceedings of the 2020 chi conference on human factors in computing systems (pp. 1–14). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376628

Dudley, J. J., Schuff, H., & Kristensson, P. O. (2018). Bare-handed 3D drawing in augmented reality. In Proceedings of the 2018 designing interactive systems conference (pp. 241–252). Association for Computing Machinery. https://doi.org/10.1145/3196709.3196737

Elsayed, H., Barrera Machuca, M. D., Schaarschmidt, C., Marky, K., Müller, F., Riemann, J., Matvienko, A., Schmitz, M., Weigel, M., & Mühlhäuser, M. (2020). Vrsketchpen: Unconstrained haptic assistance for sketching in virtual 3D environments. In 26th ACM symposium on virtual reality software and technology. Association for Computing Machinery. https://doi.org/10.1145/3385956.3418953

Fang, F., Zhang, H., Zhan, L., Guo, S., Zhang, M., Lin, J., Qin, Y., & Fu, H. (2023). Handwriting velcro: Endowing AR glasses with personalized and posture-adaptive text input using flexible touch sensor. The Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 6(4). https://doi.org/10.1145/3569461

Gasques, D., Johnson, J. G., Sharkey, T., & Weibel, N. (2019). What you sketch is what you get: Quick and easy augmented reality prototyping with pintar. In Extended abstracts of the 2019 CHI conference on human factors in computing systems (pp. 1–6). Association for Computing Machinery. https://doi.org/10.1145/3290607.3312847

George, G., Oommen, R. M., Shelly, S., Phillipose, S. S., & Varghese, A. M. (2018). A survey on various median filtering techniques for Removal of impulse noise from digital image. In 2018 conference on emerging devices and smart systems (ICEDSS) (pp. 235–238). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/ICEDSS.2018.8544273

GitHub. (2018). Just a line. https://justaline.withgoogle.com/

Glauser, O., Wu, S., Panuzzo, D., Hilliges, O., & Sorkine-Hornung, O. (2019). Interactive hand pose estimation using a stretch-sensing soft glove. ACM Trans. Graph., 38(4). https://doi.org/10.1145/3306346.3322957

Hansen, T. R., Eriksson, E., & Lykke-Olesen, A. (2006). Use your head: Exploring face tracking for mobile interaction. In Chi ’06 extended abstracts on human factors in computing systems (pp. 845–850). Association for Computing Machinery. https://doi.org/10.1145/1125451.1125617

He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., & Chao, Y. (2017). The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition, 70(C), 25–43. https://doi.org/10.1016/j.patcog.2017.04.018

Heller, F., Ivanov, S., Wacharamanottham, C., & Borchers, J. (2014). Fabritouch: Exploring flexible touch input on textiles. In Proceedings of the 2014 ACM international symposium on wearable computers (pp. 59–62). Association for Computing Machinery. https://doi.org/10.1145/2634317.2634345

Jang, S.-A., Kim, H.-I., Woo, W., & Wakefield, G. (2014). Airsculpt: A finger interaction glove. In Proceedings of the 2014 conference on human factors in computing systems (pp. 235–244). Association for Computing Machinery. https://doi.org/10.1145/2556288.2557115

Jiang, Y., Zhang, C., Fu, H., Cannavò, A., Lambertí, F., Lau, H. Y. K., & Wang, W. (2021). Handpainter - 3D sketching in VR with hand-
Zhang, Z., Zhu, H., & Zhang, Q. (2020). Arsketch: Sketch-based user interface for augmented reality glasses. In Proceedings of the 28th ACM International Conference on Multimedia (pp. 825–833). Association for Computing Machinery. https://doi.org/10.1145/3394171.3413633

About the Authors

Enting Ying is currently working toward the master’s degree with the School of Informatics, Xiamen University, Xiamen, China and supervised by professor Shihui Guo and associate professor Ming Qiu. His research interests include human–computer interaction and virtual reality.

Tianyang Xiong is currently working toward the master’s degree with the School of Informatics, Xiamen University, Xiamen, China and supervised by professor Shihui Guo and associate professor Ming Qiu. His research interests include human–computer interaction and virtual reality.

Gaoxiang Zhu received the bachelor’s degree in mechanical design, manufacturing and automation from Hunan University of Arts and Science, Hunan, China, in 2011. He is currently working in Hunan Huanan Optoelectronics (Group) Co., Ltd., engaged in scientific research project demonstration. He is interested in 3D modeling.

Ming Qiu received the PhD degrees from Zhejiang University, Hangzhou, China, in 2006. He is currently an assistant professor at the School of Informatics, Xiamen University. He is interested in intelligent information processing, Semantic Web, and Ontology.

Yipeng Qin received the BSc in Electronic Engineering from Shanghai Jiao Tong University, China and the PhD in Computer Science from the National Centre for Computer Animation, Bournemouth University, UK. He is currently an Associate Professor at the School of Computer Science and Informatics, Cardiff University, UK.

Shihui Guo received the BS degree from Peking University, Beijing, China, and the PhD degree in computer animation from the National Centre for Computer Animation, Bournemouth University, Poole, U.K. He is currently an associate professor with the School of Informatics, Xiamen University. His research interests include virtual reality.