Scene Re-Experience: A New Design to Enhance Remote Communication

YANG XU
44191638-8

Master of Engineering

Supervisor: Prof. Jiro Tanaka
Graduate School of Information, Production and Systems
Waseda University

September 2021

Abstract

After graduation, because people are away from home to work, to further their education, or because of the recent COVID-19 situation, people cannot stay with their families and friends. People need to use some remote communication applications to maintain contact with family and friends and gain a sense of company. However, the existing mainstream social media applications, such as WeChat and LINE, only provide communication functions such as text communication, picture communication, and real-time video call. The content is limited to plane and lacks immersion and realism. The existing remote communication experience needs to be improved, and new remote communication methods need to be developed.

Our system attempts to use augmented reality technology to enhance the experience of remote communication. Augmented reality technology can provide virtual content on the basis of the real world, and improve the immersion of the system by combining the real environment and virtual content.

Also, when people communicate with family and friends remotely, they always talk about recent events. It is difficult for family and friends to understand the event in detail only through verbal description or video. Our system attempts to help users better understand the event by converting 2D video into 3D animation and allowing users and their families and friends to re-experience the 3D animation in a virtual environment.

In all, our system includes the following three main innovative designs: re-experiencing recent events through 3D animation, real-time communication through voice and 3D painting, and real-time simulation of user actions through user avatar to improve the realism and immersion of the system.

Keywords: AR remote communication, 2D-to-3D converting, User avatar, 3D painting

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Jiro TANAKA, for the continuously support on my study and research. He is the man who leads me into HCI academic research field and he helps me find a research direction to provide people with a better remote communication experience under the difficult situation of COVID-19. During 2-year study in Waseda University, he uses his knowledge, patience, motivation and enthusiasm guides me a lot. In the past two years, his advice has been my beacon in the sea of confusion, guiding me forward. After my intermediate presentation failed, he helped me find the problem and gave effective suggestions. I am deeply grateful to him for his help in my two years of study and life in Japan. It is my great honor to have the opportunity to do research in IPLAB and to be a student of Professor Jiro TANAKA.

Next, I would like to thank all members from IPLAB. Thank you for your tireless efforts to help me sort out my research ideas and put forward constructive suggestions for improvement. Thank you for giving advice and encouraging me to move on. You are not only my best partner in my research, but also my best friend in my life. Thanks for your companionship and support.

Then, I want to thank my friends who have accompanied me to play games, have dinners, and travel during the past two years. Due to COVID-19 and I am in a foreign country, I am very grateful to these friends who accompanied me a lot to reduce my feelings of restlessness and loneliness.

Finally, I want to give my gratitude to my parents. You give me financial support and warm mental support throughout my master study life. I would also like to thank my motherland and the staff of the embassy for the anti-epidemic supplies and care they delivered during these 2 years.

Contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Introduction

After graduation, because people are away from home to work, to further their education, or because of the recent COVID-19 situation, people cannot stay with their families and friends. Fortunately, due to the development of social media applications, it's easier to maintain relationships among friends and families over distances[1]. However, due to the lack of physical contact, time difference, and high dependence on the Internet, people often experience negative emotions in long-distance relationships[2].

On the other hand, people always talk about recent events when chatting remotely with their families and friends. However, due to the limitations of the narrative[3][4], it is difficult for the audience to understand the full picture of the event and gain a sense of immersion only through verbal description.

As a result, we notice that there is a need to enhance the remote communication experience and the experience of talking about recent events. In order to achieve these goals, we need to solve the following two problems:

1. How to enhance the remote communication experience?

2. How to make people feel more immersed when talking about recent events?

To solve the first problem, we investigated and found that augmented reality technology will increase the immersion of the game[5], thereby enhancing the gaming experience. Inspired by this, we designed and used augmented reality technology to increase the immersion of remote users in real-time communication, thereby improving the remote communication experience.

For the second problem, we designed a new communication method, "scene re-experience". "Scene re-experience" means that, when talking about recent event, users can view the 3D animation of the main characters of the event, so as to obtain more detailed information[6] and at the same time obtain a deeper sense of immersion.

We also designed and implemented some functions to improve the user experience, such as: designing communication methods for users other than voice communication, using personalized user avatar to improve user ownership and presence[7], and letting user avatar simulate user actions in real time to improve the realism of the system.

To sum up, in this study, we designed and implemented a new remote communication method, "scene re-experience", to enhance the remote communication experience.

1.2 Organization of Thesis

The rest of the thesis is organized as follows: Chapter 2 will introduce the background of the thesis and also analyze how we figure out our standards and requirements. Chapter 3 will briefly talk about the research goal and also the approaches. Chapter 4 will introduce the related work. Chapter 5 will conclude the system design part, where the design concept and ideas will be illustrated and algorithm will also be told. Chapter 6 will be the system implementation part where the detailed environment and implementation will be talked. Chapter 6 will introduce the related work. Chapter 7 will be the conclusion and future work part, where we will conclude the previous content and discuss the future expectation.

Chapter 2

Background

2.1 Current Remote Communication Method

Due to separation or the impact of COVID-19, people lack the opportunity to stay with their families and friends and communicate face-to-face. Existing social media has its limitations[2] . The contents of some existing social media applications[8][9] are limited to the plane and lack spatial context. Some people researchers tried to combine the mixed reality technology and social media to increase user immersion and improve user communication experience[10]. In their system, users have their own avatars, and they can talk and interact with friends far away in the same virtual environment. Some researchers have also designed a virtual reality lecture system where users can use their avatars to appear in a virtual classroom to listen to the presentation of the presenter[11]. Some other researchers worked on the MR meeting system. They designed a mixed reality conference system. The user's avatar appears in a virtual conference room to participate in the conference, and at the same time, gestures can be used to perform some operations[12].

2.2 Criteria of Better Remote Communication Method

When we talk about the drawbacks of the existing remote communication methods, we always talk about the limitations of the communication content, the limitations of the verbal description and the lack of realism. A better method of remote communication needs to include the advantages of existing social media and have more and better functions, such as:

1. More non-planar contents, such as: 3D models, etc.
2. More immersive, such as being able to see the whole body instead of just the face.
3. More real, such as being able to see the other person's actions in real time.
4. More ways to communicate, not just voice.

In summary, current remote communication methods are not sufficient to meet users' requirements. A better remote communication method with high immersion, high realism and more spatial context is the development direction.

Chapter 3

Research Goal and Approach

3.1 Goal

The fundamental goal of this research is to provide a better remote communication method and improve the remote communication experience.

As a remote communication system, we must consider generating the advantages of the existing real-time chatting system, while improving the shortcomings, such as: lack of realism, lack of immersion, lack of interaction, and current contents are limited to planes.

After analyzing some existing real-time chatting systems and virtual meeting systems, some criteria for improving the remote communication experience are summarized. Therefore, in this research, we propose a new remote communication method, "Scene Re-experience", and focus on the following parts for improvement:

1. Use augmented reality technology to improve immersion.
2. Use user avatars to improve realism.
3. Provide spatial context through 3D animation.
4. Provide more ways to interact with the environment and the other user.

3.2 Approach

First, we use head-mounted mixed reality devices to allow users to enter the virtual environment and view virtual content.

Then, users can chat with their families and friends in real-time by voice in the virtual environment, or use virtual brushes to draw paintings to express their ideas.

Next, when talking about recent events, users can choose to share the 3D animation of the main characters in the event and watch the animation together with their families or friends, so as to get more spatial context and the feeling of re-experiencing the scene.

Last, in order to improve the realism of using experience, we use the avatar of the user's friend to represent the user's friend, and let the avatar simulate the actions of the user's friend in real time, so as to provide the user with the feeling of face-to-face communication with the friend.

When using this system, users will feel more immersive and realistic, thereby improving the experience of remote communication.

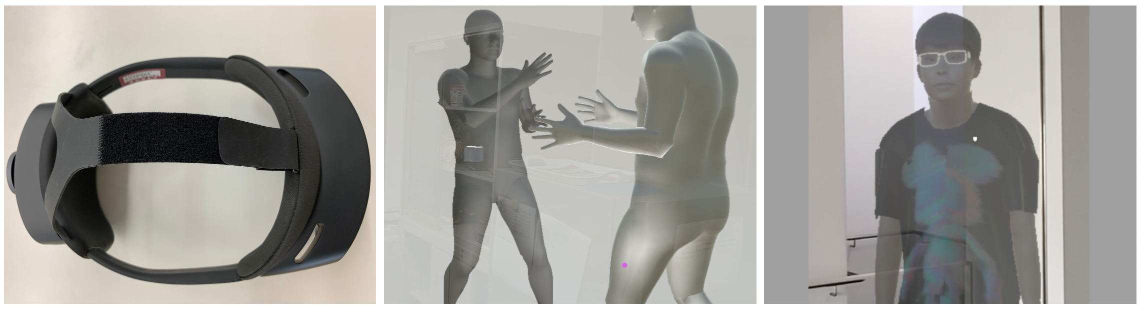

(a) MR HMD Device (b) 3D Animation (c) User Avatar

Fig. 3.1 Three parts for the new method

3.3 Novelty

The novelty of this research mainly reflects in the two aspects:

1. We provided an environment where users can immerse themselves in communication with their remote families and friends in their own homes.
2. We designed a new communication method, "Scene Re-experience", so that users can re-experience and talk about recent events immersively.

Chapter 4

Related Work

In this section, we introduce related work for remote chatting system, 2D video to 3D animation converting, making avatars more realistic and interaction methods of remote communication.

4.1 Remote Chatting System

Due to the living in different places, or the impact of COVID-19, some people lack the opportunity to communicate face-to-face with their families and friends. At this time, remote chatting systems are quite important. Some companies have released real-time audio and video chat systems[8][9], and some studies have tried to improve the users' sense of immersion and realism through augmented reality or virtual reality technology in remote communicating[13][14].

We focus on the research which is related to our goal and divide into the following two parts.

4.1.1 Common Remote Chatting System

Tencent Company has developed a social media application, WeChat[8]. It supports users to use text and images for remote communication, and also provides real-time audio and video chatting functions. LINE Corporation also developed a similar application, Line[9].

Line is a social media application that integrates many functions, such as: texts, video call, stickers, emojis, open chat group and timeline.

4.1.2 AR/VR Remote Chatting System

Boletsis and Costas proved that virtual reality technology will improve the immersive feeling of users[15]. On the other hand, Toni Alatalo, Timo Koskela, Matti Pouke, Paula Alavesa and Timo Ojala introduced a method of constructing virtual cities and virtual environments on the web[16] and Thomas Hilfert and Markus König introduced a method for constructing a low-cost virtual reality environment for engineering and construction[17]. These studies can help us develop the system.

Research on AR interfaces for remote conferencing or chatting has largely been in the area of sharing remote workspaces[18]. A collaborator can look from someone else's viewpoint at the real world and place virtual annotations there to help them with the task at hand[19]. Cha Zhang, Qin Cai, Philip A. Chou, and Zhengyou Zhang designed and developed an augmented reality immersive teleconference system[20]. They use the Kinect camera, IR camera and color cameras to take the user's photos from different angles, and design an algorithm to render these photos into a 3D video stream to simulate the user's avatar.

As for VR part, Du Ruofei, Li David and Varshney Amitabh developed a virtual world social media based on real geotagged information on both website and HMD device[21][10]. Users can use avatars to move in the virtual world and talk with other users. The geotagged information in the virtual world corresponds to the real world one-to-one, and billboards, famous buildings, etc. can be viewed in the virtual world.

4.2 2D Video To 3D Animation Converting

Junyuan Xie, Ross Girshick and Ali Farhadi proposed Deep3D, a fully automatic 2D-to-3D conversion algorithm that takes 2D images or video frames as input and output 3D stereo image pairs[22]. The stereo images can be viewed with 3D glasses or head-mounted VR displays. Also, Liang Zhang, Carlos Vázquez and Sebastian Knorr provided an overview of automatic 2D-to-3D video conversion with a specific look at a number of approaches for both the extraction of depth information from monoscopic images and the generation of stereoscopic images[23]. But our research focuses on automatic generation of actions of main characters in the 2D video, but not 2D-to-3D video converting. So we continue to study the following work.

Howe Nicholas, Leventon Michael E. and Freeman William T. presented a system that reconstructs the 3D motion of human subjects from single-camera video[24]. Leonid Sigal and Michael J. Black proposed a hierarchical process for inferring the 3D pose of a person from monocular images[25].

4.3 Make Avatars More Realistic

Our research focuses on improving the user experience of remote communication. We design to use user avatars to simulate real people, so as to improve the user's immersion and realism. Therefore, how to make avatars more realistic is particularly important.

Dongsik Jo, Ki-Hong Kim and Gerard Jounghyun Kim did a study to show that the participants generally exhibited a higher sense of co-presence when situated with a real environment background (realistic video) and greater confidence/trust when interacting with a reconstructed realistic looking avatar[26]. Then they developed a system to project the traditional two-dimensional video-based tele-conference systems into the three-dimensional immersive and augmented reality (AR) based on which one can communicate with the tele-ported remote other as if present, moving and interacting naturally in the same location[27].

4.4 Interaction Methods of Remote Communication

In real life, in addition to language, people also have various communication methods such as body movements, drawings, and expressions. Our research wants to provide users with more communication methods to improve the user's remote communication experience.

Michael F. Deering developed HoloSketch, a virtual reality-based 3D geometry creation and manipulation tool[28]. In HoloSketch, virtual objects can be created with a 3D wand manipulator directly in front of the user, at very high accuracy and much more rapidly than with traditional 3D drawing systems. Microsoft also provides a 3D painting application on HoloLens 2 for users to draw mid-air. Inspired by these works, our research is also designed to provide users with the function of 3D painting to enrich users' communication methods.

Chapter 5

System Design

5.1 System Overview

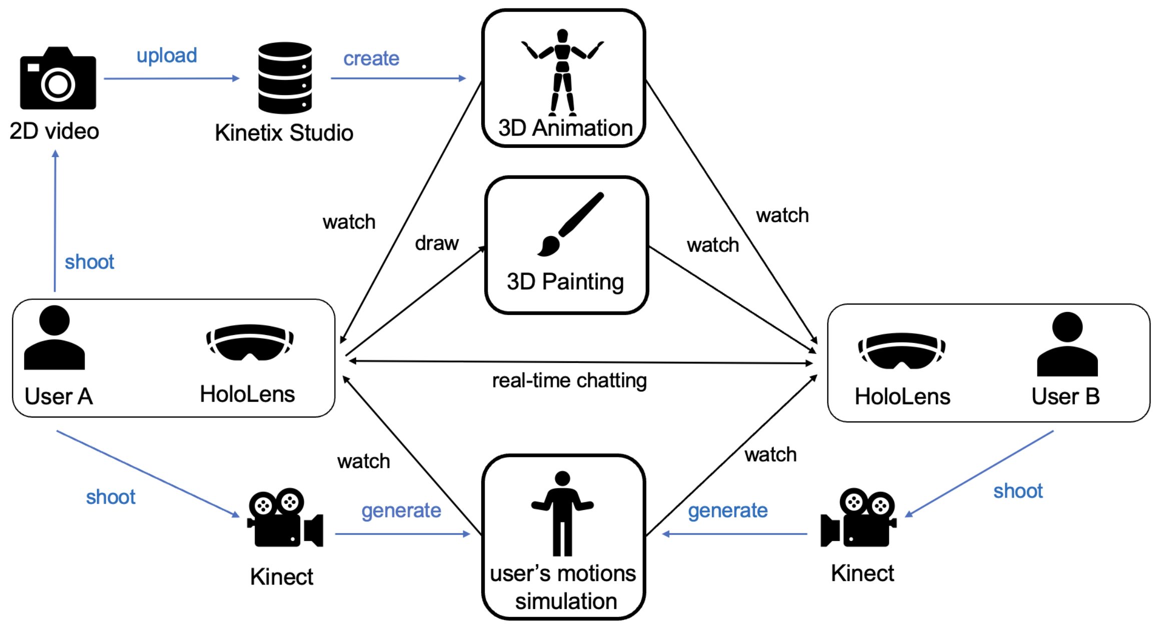


Fig. 5.1 System overview

In this chapter, we will introduce our system design and each important pieces of our approach. The Fig. 5.1 shows the overview of our system.

The system provides users a new remote communication method, "Scene Re-experience". With "Scene Re-experience", users can sit in their homes and talk with friend's avatar through voice, text or drawing in real time, just like talking face-to-face with their friend. At the same time, if they talk about recent events, one user can also share 3D animations of the characters in the event to the other and watch the 3D animation together in real time. In this way, users will obtain a higher sense of immersion and more detailed information than watching ordinary 2D videos together.

In order to convert ordinary 2D videos into 3D animation, our system uses Kinetix Studio[29] provided by Kinetix SAS Company. Kinetix Studio can generate 3D animation of main characters in 2D video.

In order to provide real-time chatting function for remote users on HoloLens 2, we designed a method combining Microsoft WebRTC Toolkit[30] and a local server. The Microsoft WebRTC Toolkit provides a convenient tool for developing real-time chatting function for universal windows platforms, including HoloLens 2. After getting audio streams, we develop a local server to transfer data streams between two HoloLens 2. Our system also supports other real-time chatting methods, such as typing, painting, etc.

As for the user's motions simulation, there are 3 steps. First, we create the user's avatar based on user's photos[31]. Second, we use Kinect to catch the movements of the user's bones and joints. Finally, we bind the user's motions to the user's avatar through Unity to make it come alive.

These are three main components of "Scene Re-experience". Following subsections are detailed information about key parts of this research.

5.2 3D Animation

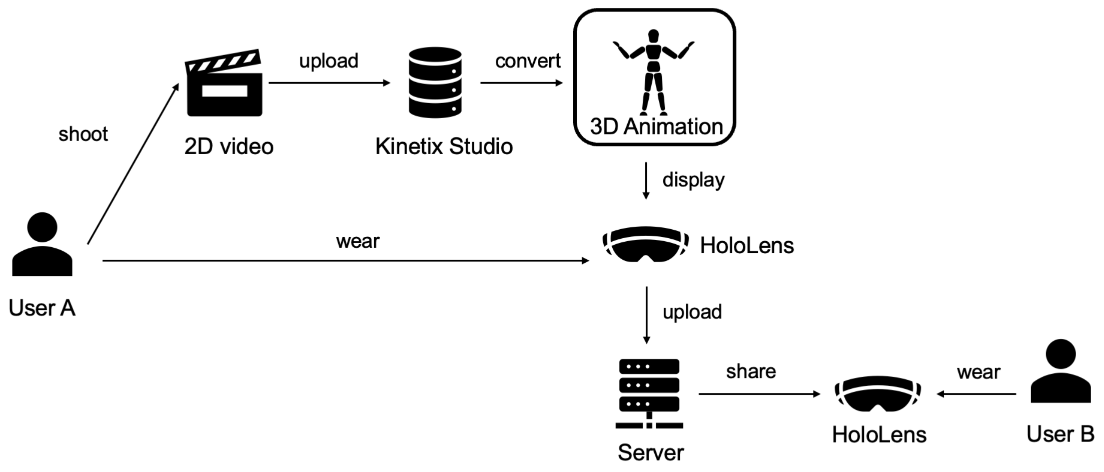


Fig. 5.2 The process of 3D animation generation and sharing

As the Fig. 5.2 shows, 3D animation contains three main operations:

1. 3D animation generation

   The user needs to shoot a 2D video first, and then generate a 3D animation based on the 2D video.

2. 3D animation display

   After generating the 3D animation, we use Unity to process the animation. Then we deploy the 3D animation to the HoloLens platform to display.

3. 3D animation sharing

   Any user using this system can choose to share their 3D animation. We provide a local server to process and forward the data, so as to realize the sharing of animation.

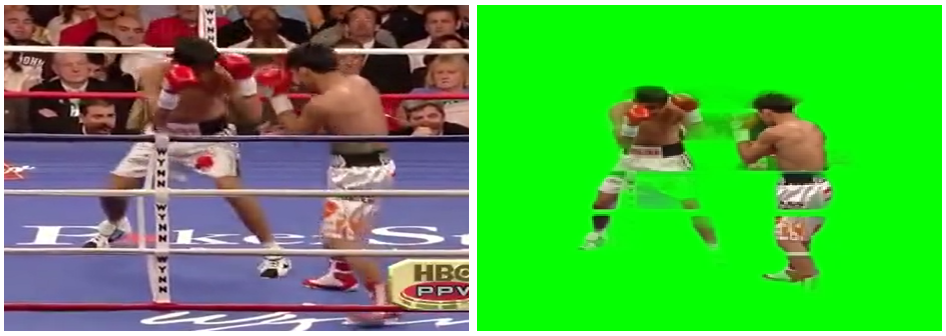
Every user is required to wear HoloLens to enter the virtual world and use our system. The HoloLens is required to be connected to the Internet through WIFI.

5.2.1 3D Animation Generation

(a) MeiCe (b) Kinetix Studio

Fig. 5.3 Software we used

Fig. 5.3 shows the software we used to generate 3D animation. There are three main steps. First, we use a camera (in our system, we used a Sony alpha 6100L camera) to shoot 2D videos.



(a) Original video (b) Result

Fig. 5.4 The result of "MeiCe"

Second, we use a Chinese application, "MeiCe"[32], to remove background in the 2D video. If the video background is too complex, it will affect the generation of 3D animation, so we need a pure color background. Fig. 5.4 shows the effect of MeiCe removing the video background.

Last, we upload the 2D video to the Kinetix Studio[29]. Kinetix Studio will automatically generate 3D animations of characters based on the actions of the characters in the video.

After these steps, we generated the 3D animation of a recent event based on the 2D video. Next, we can display it on HoloLens to have an immersive watching experience.

5.2.2 3D animation display

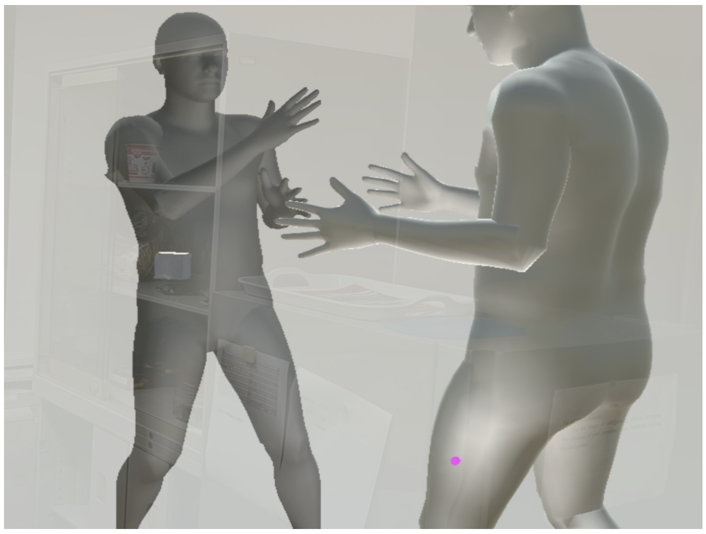


Fig. 5.5 The view of 3D animation on HoloLens

After 3D animation generation, we use Unity Engine to set the position, rotation and scale of the 3D animation. Then we deploy it to HoloLens platform through Visual Studio and WIFI. After users wear HoloLens, they can watch 3D animations in their room to get an immersive experience. They can also drag the 3D animation to place it where they like.

5.2.3 3D animation sharing

Anyone who uses our system can choose to share their own 3D animation. We provide a local server to achieve sharing function. When using our system, users first choose the friends they want to share, and then start sharing 3D animations. The 3D animation information will be collected and then sent to the local server. We use a laptop (MacBook Pro 13-inch, M1, 2020) as a local server. After the server receives the animation information, it forwards these messages to the corresponding user. The user's HoloLens will automatically play after receiving the 3D animation information from server.

5.3 Real-time Chatting

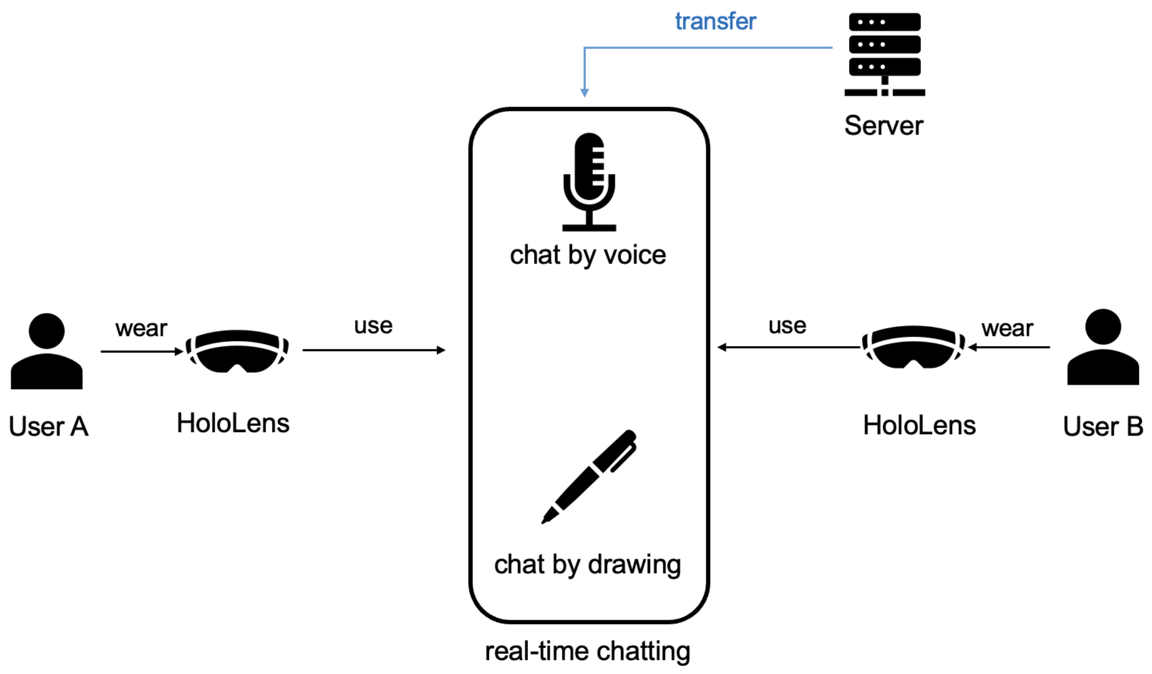


Fig. 5.6 The overview of real-time chatting

As the Fig. 5.6 shows, users are required to wear HoloLens to enter the virtual world and use our system. We provide two kinds of real-time chatting methods, one is through voice and the other is through drawing. We also provide a local server to transfer chatting data and achieve synchronization between two users.

5.3.1 Chat by Voice

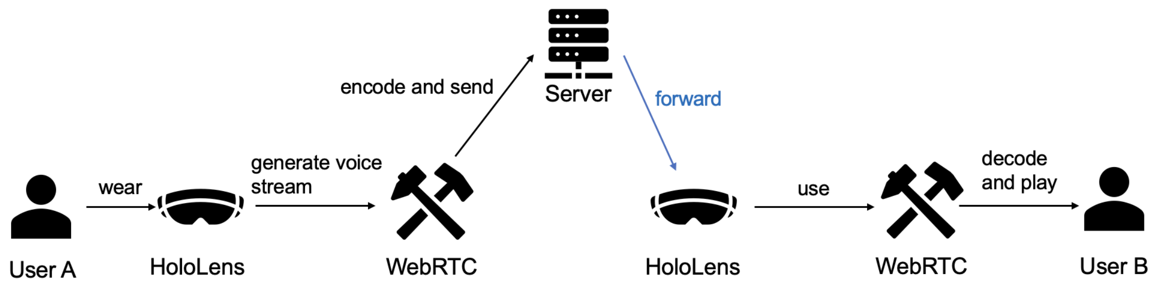


Fig. 5.7 The overview of real-time chatting by voice

Microsoft WebRTC Toolkit[30] is a development toolkit for audio and video chatting functions for universal windows platforms. As the Fig. 5.7 shows, we use HoloLens's microphone to collect the user's voice, and then use WebRTC to encode the audio stream. Next, we use the local server to forward the audio stream to the other user. After receiving the messages, the other user's HoloLens uses WebRTC to decode the audio stream and play it.

5.3.2 Chat by drawing

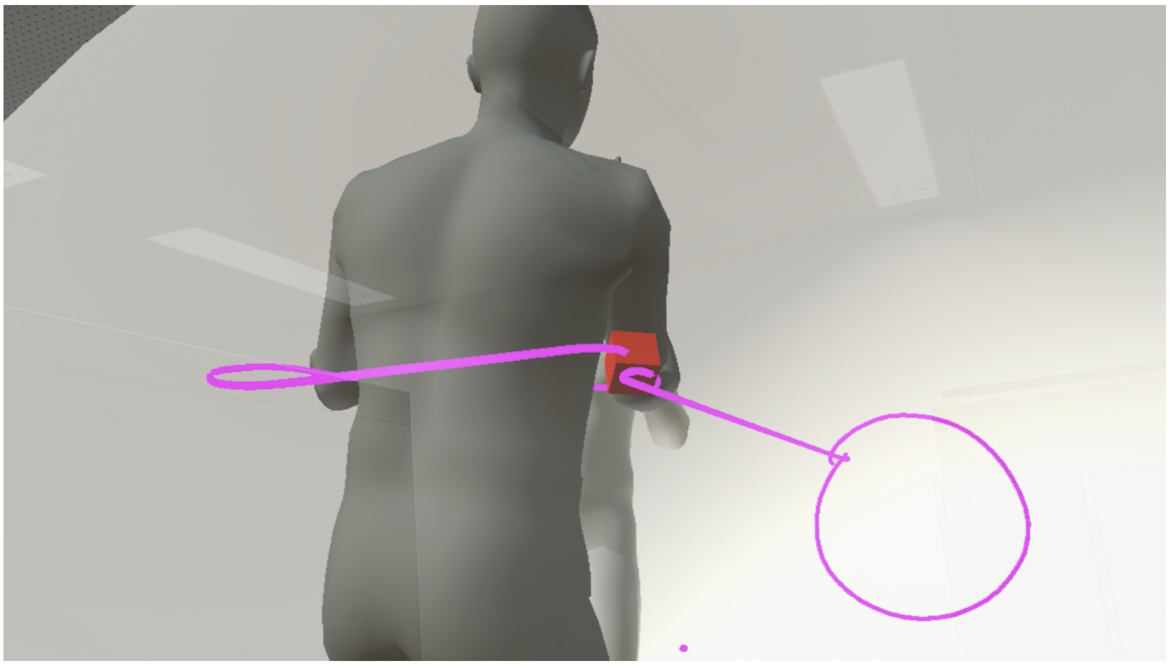


Fig. 5.8 The view of 3D painting on HoloLens

Only voice chat is not enough when watching 3D animations together. In our system, users can use brushes to draw in the virtual world, mark the parts of 3D animation that they are interested in, express their emotions, or write their own comments. Since the system provides three-dimensional paintings, the drawn paintings will remain in the space, which matches the 3D animation better and enhances the user's sense of immersion.

Our local server can also works for synchronizing paintings between two users. When a user holds the paintbrush and starts to paint, the movement of the brush and the drawn painting will be synchronized to the other user in real time, as if two users are watching the animation, drawing and communicating face-to-face.

5.4 User's Motions Simulation

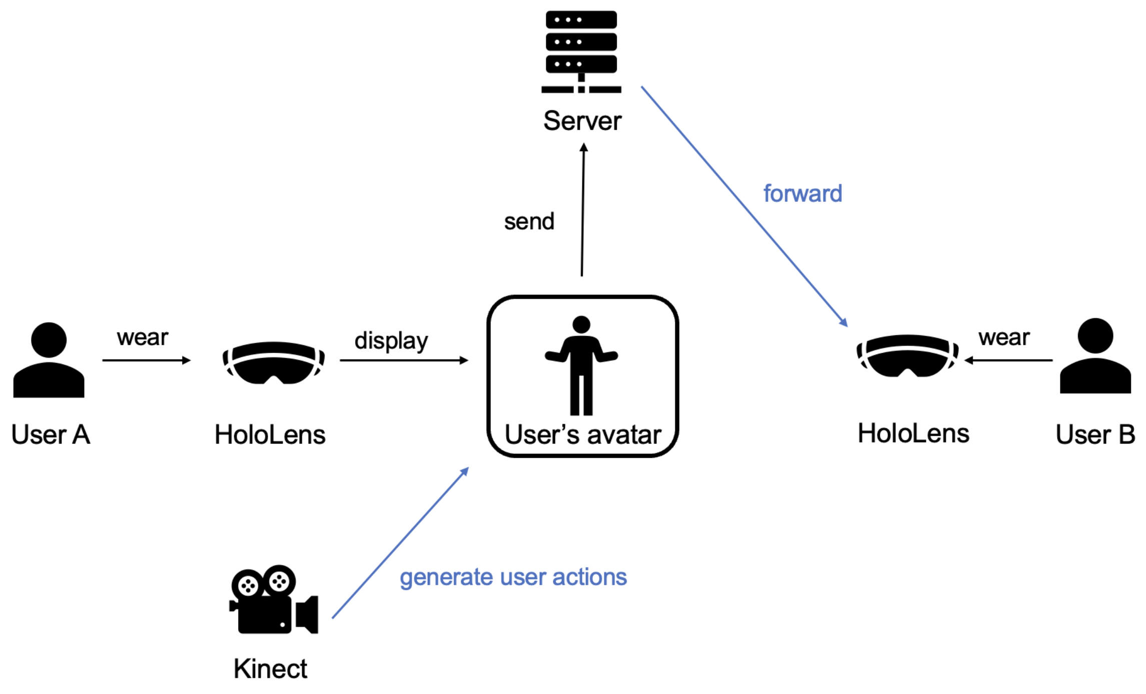


Fig. 5.9 The overview of user's motions simulation component

In order to enhance the user's immersion, the friend's avatar will appear in the user's room. The user can talk to the friend's avatar, as if communicating face-to-face with real people.

In the real world, body language is also important when people communicate face to face. In order to improve users' sense of immersion, enhance remote communication, we provide the function that the user's avatar simulating user actions in real time. In detail, User A can watch User B's avatar simulating User B's actions in real time and User B can watch User A's avatar simulating User A's actions as well.

To achieve the above function, there are three main steps:

1. Build user's avatar

   First of all, we must build user's avatar[31] and deploy it to HoloLens.

2. Action feedback

To make the avatar more realistic, user's action feedback are needed. If the avatar can imitate the user's actions in real time, it will look more realistic.

3. Synchronization

   Our system is oriented to real-time communication between users. So synchronization of user avatar actions is indispensable.

5.4.1 Build User's Avatar

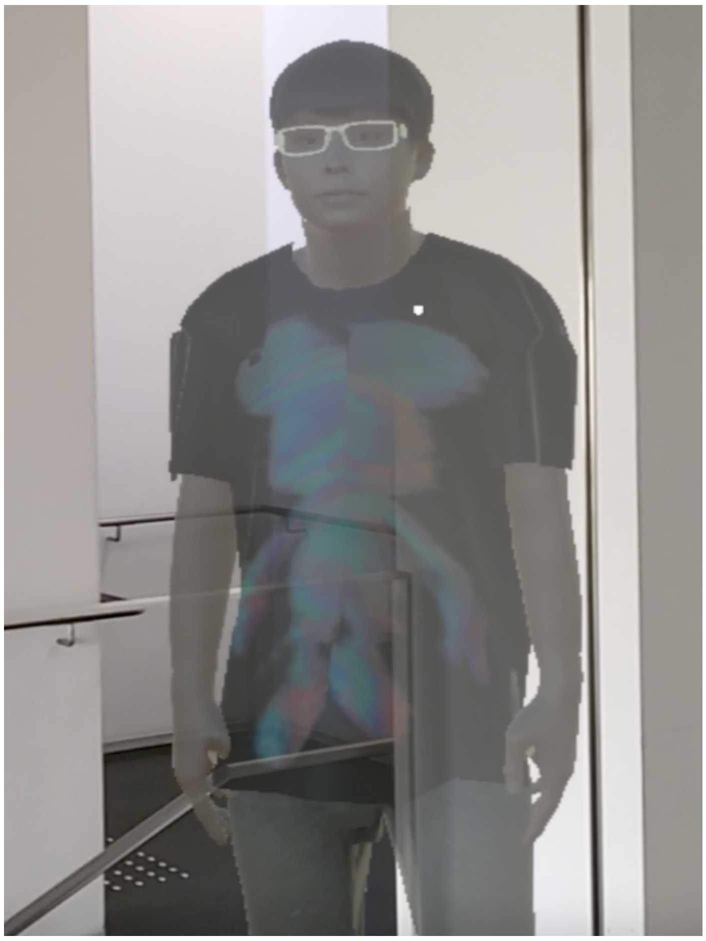


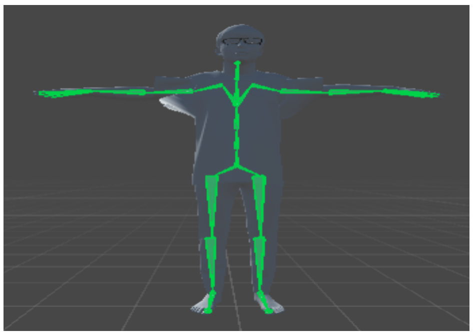
Fig. 5.10 The View of user's avatar on HoloLens

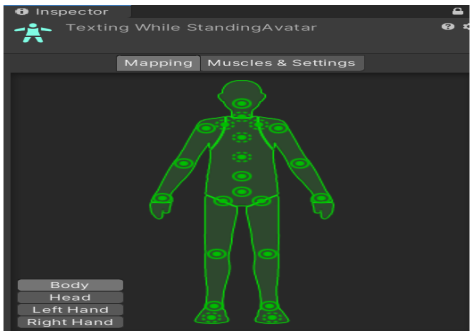
To build user's avatar, we used 3D LOOK[33] and PIFuHD[31]. 3D LOOK provides services for avatar modeling based on photos. First, we take photos of the front and side view of the user. Then we upload the photos to the 3D LOOK, and finally get the user's avatar.

After downloading the user's avatar, we process the avatar by Unity Engine. We provide the user's avatar with clothes, skin, bones and joints to make it more realistic. Also, we set the position, rotation and scale of the avatar to fit the indoor environment.

Fig.5.10 shows our result, the view of the user's avatar on HoloLens.

5.4.2 Action Feedback


(a) Bone structure of the user's avatar



(b) Inspector view of avatar joints

Fig. 5.11 Schematic diagram of the bones and joints structure of the user's avatar

To make the user's avatar more realistic, to make users more immersive, we provide the function that the user's avatar simulating the user's actions in real time.

To achieve this function, we use Microsoft Kinect, which can detect the user's bones and joints and then record actions of them. At the same time, we use Unity Engine to bind the actions of these bones and joints to the the user's avatar. In this way, the user's avatar can have real-time action feedback based on the user's real actions.

5.4.3 Synchronization

Our local server can also receive and forward action information of the user's avatar. There are five main steps:

1. Kinect generates the user's actions and transfers action messages to the user's HoloLens.

2. The HoloLens sends action messages to the server.

3. After receiving the action messages, the server forwards them to the other user.

4. The other user’s HoloLens receives the action messages and binds them to the avatar.

5. After binding, the avatar will start actions.

In our system, User A can only see User B’s avatar in his or her room. Also, User B can only see User A’s avatar. This is to simulate a real face-to-face communication experience.

5.5 Use Case Scenario

In this part, to provide readers a better understanding of our system, we will introduce a use case scenario. We use a practical example to tell the user the process of using this system and what the user can do in this system.

5.5.1 Scenario: Boxing Match

When people chat remotely with their relatives or friends, they always talk about recent events. At this time, our system will provide a high-quality remote communication experience. Users can record recent events and re-experience the events together with their friends or relatives when using our system, so as to get an immersive experience.

Suppose User A watched a boxing match a few days ago. He has a friend, User B, who is very interested in this match, but can't go to watch it. So User A recorded the match with a camera and re-experience the match with User B today.

To use the system, both User A and User B need to follow these steps:

1. User A watched and recorded a boxing match.
2. User A removed the background information in the 2D boxing video by some tools, such as: "Meice", "unscreen" and "Adobe Premiere Pro". Then User A transfers the video into 3D animation.
3. User A and User B finish the software and hardware preparations of the system, such as: preparing HoloLens 2 devices, connecting HoloLens 2 to Internet, installing the system, etc. Then they start to user our system.
4. User A selects and shares the scene he wants to re-experience with User B. In the use case scenario, he selects the scene of the boxing match.
5. User A and User B re-experience the boxing match together by our system.

Fig. 5.12 shows the timeline of how can users use the system.

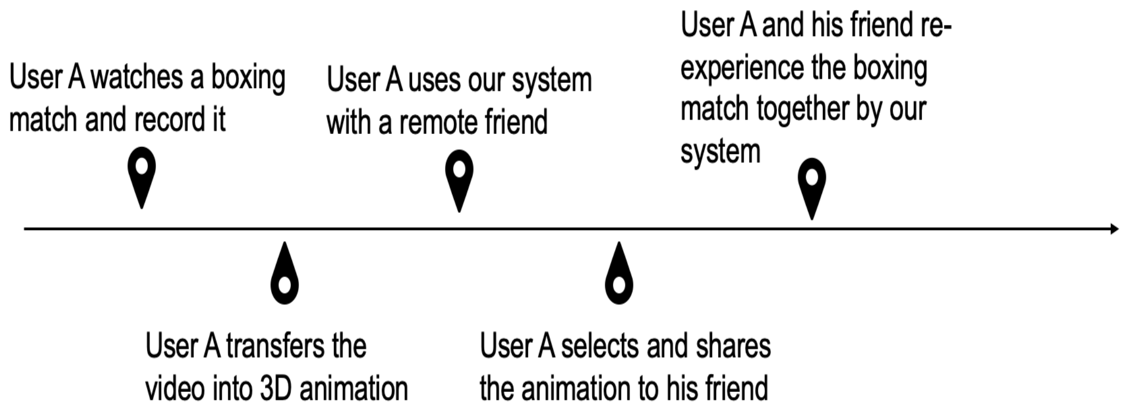

User A watches a boxing match and record it
User A transfers the video into 3D animation
User A uses our system with a remote friend
User A selects and shares the animation to his friend
User A and his friend re-experience the boxing match together by our system

Fig. 5.12 The timeline of the use case scenario

5.5.2 What Can Users Do in the System

In this system, we provide users with rich functions to interact. After the user connection is established, the scene is selected and shared, while users are re-experiencing the scene, the system provides users with the following functions:

1. User A and User B can watch 3D animation of athletes of the boxing match in virtual environment together.
2. User A and User B can check any angle of the animation and get more detailed information, for example, the angle, speed of the athlete's punch.
3. User A and User B can chat by voice in real time.
4. Each of the user can use virtual brush to write comments mid-air. They can also highlight interesting part of the animation by the virtual pen.
5. Each of the user can see the other user's avatar and talk to the avatar as if talking with a real person.

Chapter 6

System Implementation

6.1 Hardware Setup

We divided hardware of this system into three parts: recording devices, head-mounted display device and local server device. To achieve this system, we need the following hardware devices:

1. A camera which can record 2D videos;
2. An augmented reality head-mounted display device;
3. A sensor which can detect the movement of the user's joints and bones and generate a data stream;
4. A computer as a server to forward messages, store data and maintain synchronization.

Fig. 6.1 Required Hardware

Therefore, for camera, we choose Sony Alpha ILCE-6400L. It is a half-frame mirrorless camera produced by Sony Company. Its fast hybrid auto-focus allows for rapid, accurate subject tracking for great photographic experience. With this camera, we can easily shoot 2D videos with clear portraits, which is convenient for this research.

Fig. 6.2 Sony Alpha ILCE-6400L

Because this system is based on augmented reality technology and will display augmented reality content such as: 3D painting, user avatar and 3D animation, a device that can display AR content is needed.

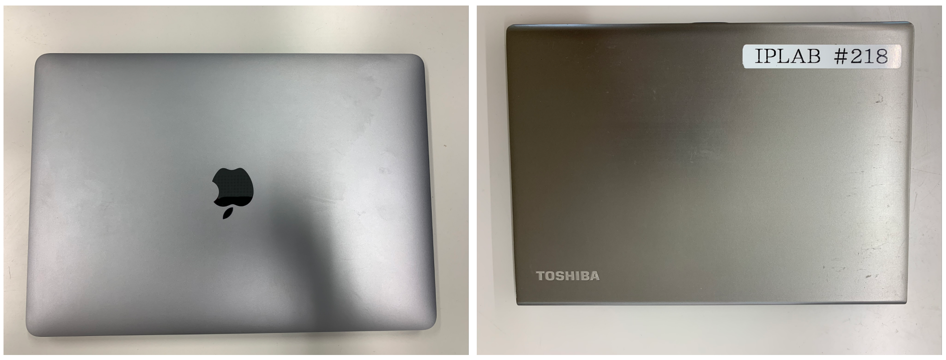
In daily life, body language is essential when people communicate with friends. In order to provide a more realistic communication experience, the system is designed to require the both of the user's hands to participate in the operation. Therefore, we chose Microsoft HoloLens 2, which is a mixed-reality head-mounted display that can support free-hand operation.

Fig. 6.3 Microsoft HoloLens 2

In order to make the user's avatar more realistic, in addition to detailed modeling work, we designed to use the Microsoft Kinect sensor to detect user actions. Microsoft Kinect can detect the movements of human bones and joints and generate a data stream. After processing, we apply the data stream to the user's avatar, allowing the avatar to simulate the user's actions in real time, which is more realistic.

Fig. 6.4 Microsoft Kinect

This system contains two users who use this system for remote communication through HoloLens 2. We use the Client-Server communication architecture, and choose a laptop as the local server to forward messages, store data, send instructions, maintain synchronization, etc. In the actual implementation process, we use an Apple Macbook Pro (13-inch, M1, 2020) as the local server.



(a) Apple Macbook Pro (b) TOSHIBA Dynabook

Fig. 6.5 Laptop

In addition, we used a TOSHIBA Dynabook laptop equipped with Microsoft Windows 10 operating system as the software system development tool.

Hardware Device	**Main Unit**
Sony Alpha ILCE-6400L	Record 2D video
Microsoft HoloLens 2	Display AR content, run the system
Microsoft Kinect	Generate user's motion
Apple Macbook Pro	Server
TOSHIBA Dynabook	Software development

Table 6.1 Hardware Setup

6.2 Software Environment

We used the following software and toolkit as technical supports:

1. "Meice", a Chinese video processing software. Based on machine learning and image recognition, it can automatically remove the background in the video and add it to a green screen that is easy to distinguish. Users of our system can use this software to do video preprocessing.

2. "Kinetix Studio", a web-side video processing tool. It can automatically extract the action information of the characters in the video and generate 3D models and animations. We use it to convert 2D video into 3D animation.

3. Microsoft Mixed-Reality Toolkit, a toolkit developed by Microsoft for the development of universal windows platforms mixed reality applications. We use this toolkit to develop the AR content of this system.

4. Microsoft WebRTC, a toolkit developed by Microsoft for the development of multi-platform real-time voice call function. We use it to achieve real-time voice chatting function on HoloLens 2 platform.

To develop AR application, we use Unity Engine 2019.4.24f1. To write c# scripts, we use Microsoft Visual Studio 2017.

Software	**Main Unit**
Meice	Remove background
Kinetix Studio	Generate 3D animation
Microsoft MRTK	Develop AR content
Microsoft WebRTC	Achieve real-time chatting
Unity Engine	Software development
Microsoft Visual Studio	Code

Table 6.2 Software Environment

6.3 Real-time Chatting

6.3.1 Voice Chatting

Based on Microsoft WebRTC toolkit, we developed real-time voice chatting function. The first thing we need to do is to establish a connection with the server. In Unity project, we created a "Peer Connection" component to mount the script that connects to the server. Fig. 6.6 shows the component we set.

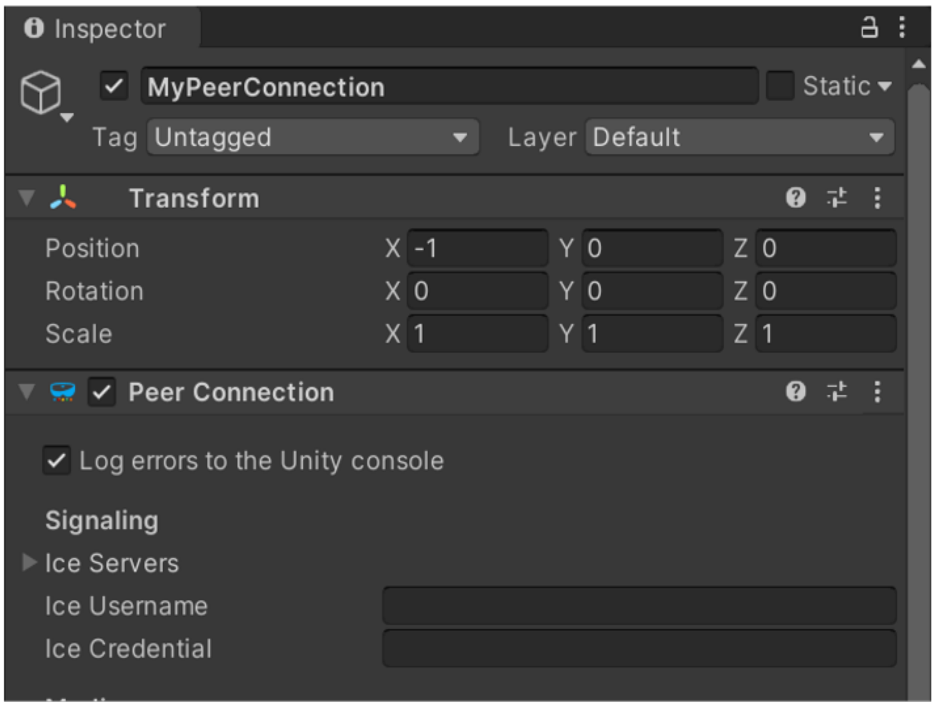
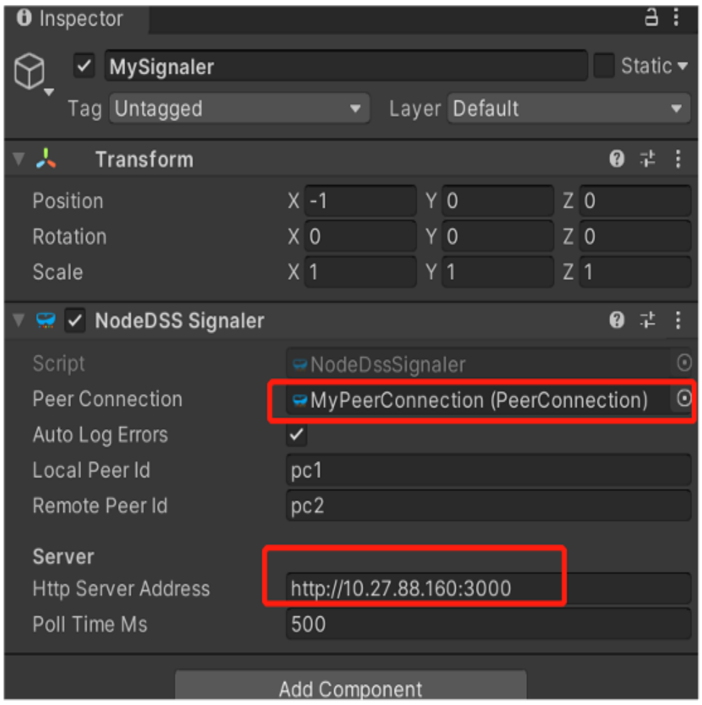

Inspector
✓ MyPeerConnection   Static
Tag Untagged   Layer Default
Transform
Position X -1 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
✓ Peer Connection
✓ Log errors to the Unity console
Signaling
Ice Servers
Ice Username
Ice Credential

Fig. 6.6 Peer Connect Component

Then we created a "NodeDSS Signaler" component to control the users who are chose to establish the connection, the network IP address of the server, and the maximum allowable network delay. This component controlled the connection through the "Peer Connection" component we created before, so we dragged the "Peer Connection" component to this signaler component. Then we set the ID of both local and remote users, the server's IP address and the allowable delay time. Fig 6.7 shows the component we set.


Inspector

MySignaler Static

Tag Untagged Layer Default

Transform

Position X -1 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

NodeDSS Signaler

Script NodeDssSignaler

Peer Connection MyPeerConnection (PeerConnection)

Auto Log Errors ✓

Local Peer Id pc1

Remote Peer Id pc2

Server

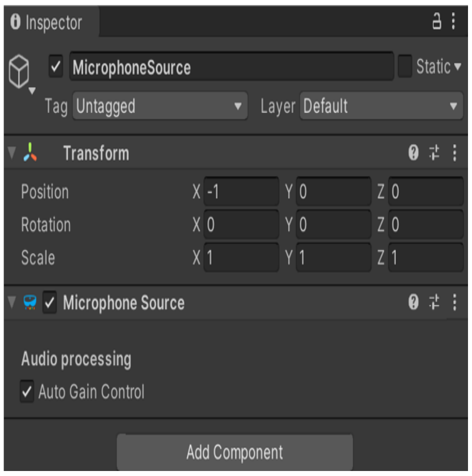
Http Server Address http://10.27.88.160:3000

Poll Time Ms 500

Add Component

Fig. 6.7 Signaler Component

After the connection was established, we need to obtain audio information. We created a game object and added the "Microphone Source" component. After the user's authorization, this component can use the device's microphone to record the user's voice and store it in the form of an audio stream. Fig. 6.8 (a) shows this component.


Inspector

MicrophoneSource Static

Tag Untagged Layer Default

Transform

Position X -1 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

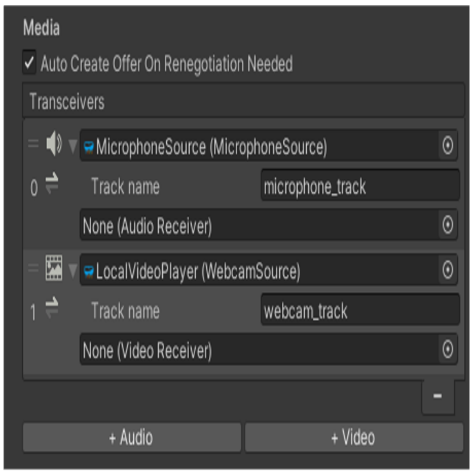
Microphone Source

Audio processing

✓ Auto Gain Control

Add Component

(a) Microphone Source Component


Media

✓ Auto Create Offer On Renegotiation Needed

Transceivers

MicrophoneSource (MicrophoneSource)

0 Track name microphone_track

None (Audio Receiver)

LocalVideoPlayer (WebcamSource)

1 Track name webcam_track

None (Video Receiver)

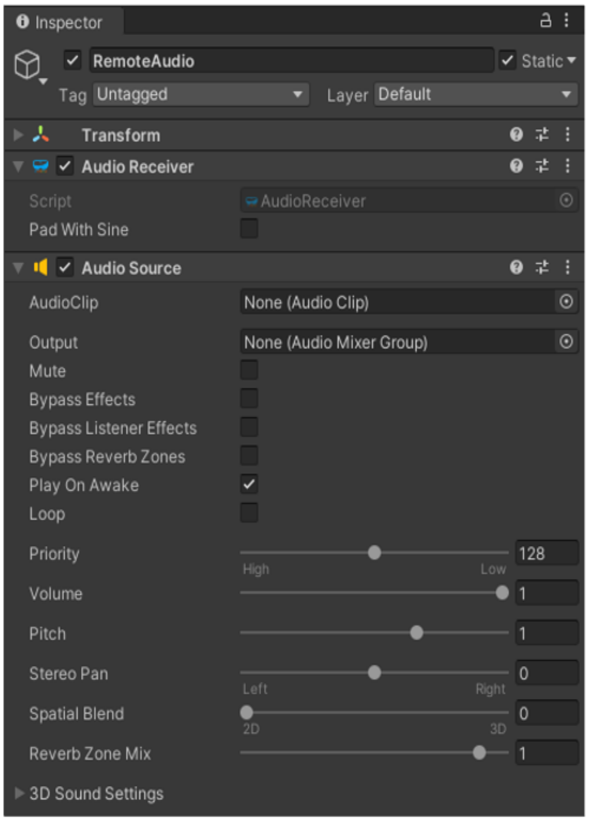
+ Audio + Video

(b) Transceiver Component

Fig. 6.8 Unity Component Used to Generate Audio Source

After we got the audio stream, we need to create a transceiver in order to send it to the other user. The transceiver is used to convert the audio stream into a binary stream for sending, or to convert the received binary stream back into an audio stream. We modified the previously created "Peer Connection" component, added a transceiver, and selected the "Microphone Source" component as the local input audio stream. Fig 6.8 (b) shows the transceiver we set.

Next we need to create a tool to play audio. We created a game object and added "Audio Receiver" and "Audio Source" components. These two components were used to receive and play audio streams. Fig 6.9 shows these two component.


Inspector

✓ RemoteAudio ✓ Static

Tag Untagged Layer Default

Transform

✓ Audio Receiver

Script	AudioReceiver
Pad With Sine	

✓ Audio Source

AudioClip	None (Audio Clip)
Output	None (Audio Mixer Group)
Mute	
Bypass Effects	
Bypass Listener Effects	
Bypass Reverb Zones	
Play On Awake	✓
Loop	
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

3D Sound Settings

Fig. 6.9 Audio Receiver and Audio Source Component

Last thing we need to do is setting up media lines. We selected the "Audio Receiver" and "Audio Source" components as remote audio source in transceiver. Fig 6.10 shows the final state of the transceiver.

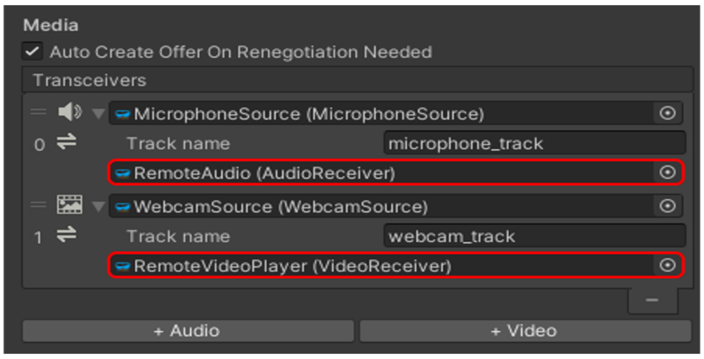

Media
✓ Auto Create Offer On Renegotiation Needed
Transceivers
0 MicrophoneSource (MicrophoneSource)
Track name microphone_track
RemoteAudio (AudioReceiver)
1 WebcamSource (WebcamSource)
Track name webcam_track
RemoteVideoPlayer (VideoReceiver)

Fig. 6.10 Final State of the Transceiver

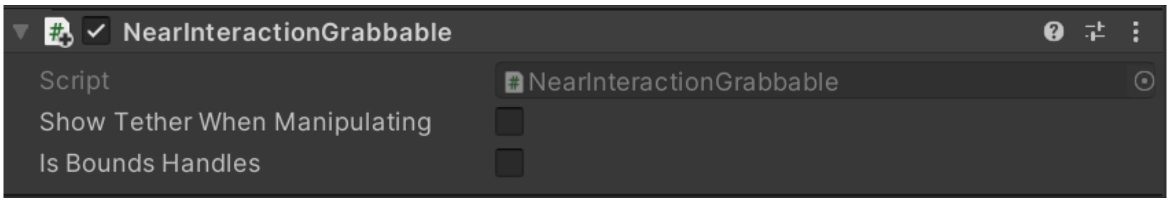
After the above steps, the two HoloLens 2 devices will be connected together through the server and can perform real-time voice chatting function.

6.3.2 3D Painting

In daily communication, the communication method through drawing is also very popular. People can express their ideas and designs more vividly through drawings, making it easier to understand. Therefore, in this system, we designed and implemented the 3D painting function. The users participating in the conversation can use the brush provided by the system to draw 3D paintings to express their ideas. The painting will be synchronized to the other user in real time. The steps to achieve this function can be divided into the following modules:

1. How can users use the brush by their hands?
2. How can the brush draw?
3. How can the painting be synchronized to the other user?

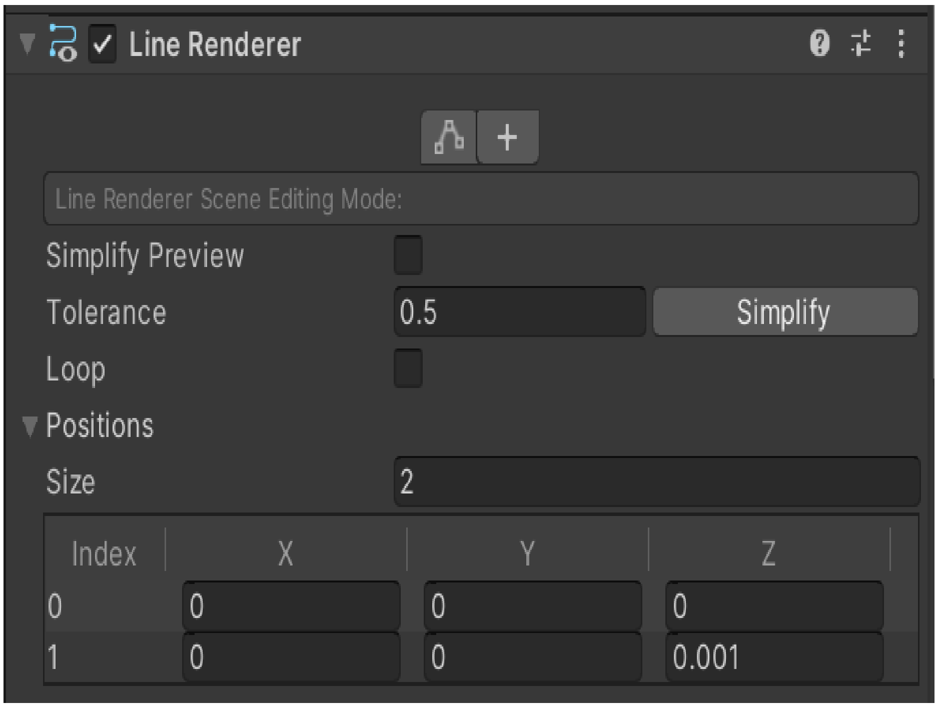
First, we used MRTK to allow users to use hands to control the brush in the virtual environment. We modeled a pen and used this model as the brush in the virtual environment. Then, we added "Interactable" and "NearInteractionGrabbable" scripts to the brush. The "Interactable" script makes the brush interactable and the "NearInteractionGrabbable" script allows users to grab the brush by hand. Fig. 6.11 shows the script.


NearInteractionGrabbable

Script	NearInteractionGrabbable
Show Tether When Manipulating	☐
Is Bounds Handles	☐

Fig. 6.11 "NearInteractionGrabbable" Script

Second, we used "Line Renderer" component to draw. We created an empty game object called "Line", which is used to draw lines. Then we added "Line Renderer" component to it. Next, we wrote scripts for the brush to control the "Line" to draw. In detail, we added "Object Manipulator" script to the brush to detect whether the user is holding the brush. If the brush was held, every time the coordinates of the brush changed, we drew a point at the position of the brush. These points were connected to become the trajectory of the brush movement, that is, the drawn paintings. Fig. 6.12 shows the "Line Renderer" component and Fig. 6.13 shows the "Object Manipulator" script.


Line Renderer

Line Renderer Scene Editing Mode:

Simplify Preview ☐

Tolerance 0.5 Simplify

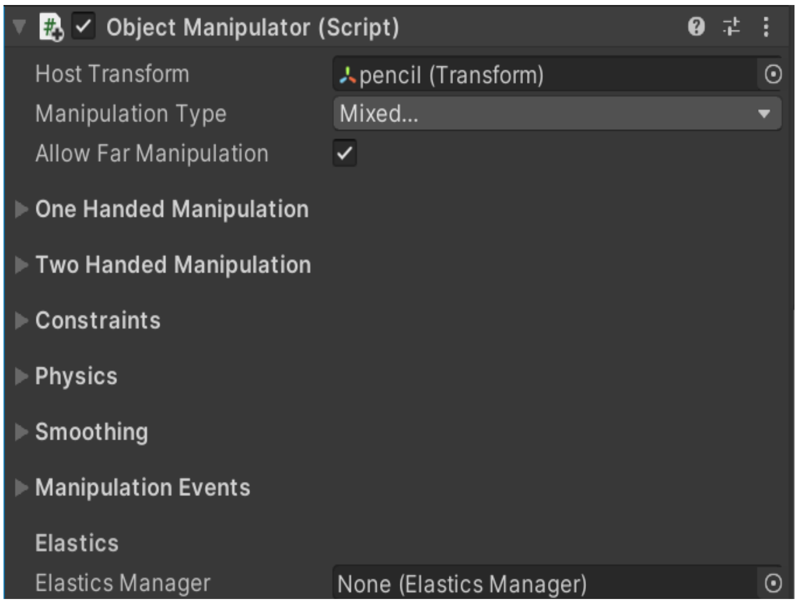
Loop ☐

Positions

Size 2

Index	X	Y	Z
0	0	0	0
1	0	0	0.001

Fig. 6.12 "Line Renderer" Component


Object Manipulator (Script)

Host Transform	pencil (Transform)
Manipulation Type	Mixed...
Allow Far Manipulation	✓

One Handed Manipulation

Two Handed Manipulation

Constraints

Physics

Smoothing

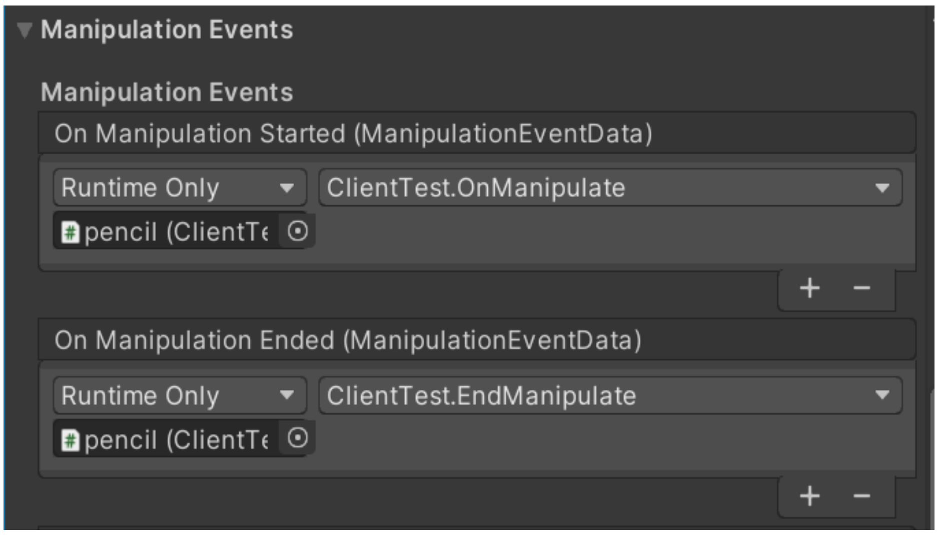
Manipulation Events

Elastics

Elastics Manager None (Elastics Manager)

Fig. 6.13 "Object Manipulator" Script

What's more, in reality, users do not always paint while holding the brush. They may also pick up the brush to a new position and start painting. Therefore, we have set two states for the brush, the "paintable" state and the "unpaintable" state. The brush is initially in "unpaintable" state, and the user can click on the brush to change the state. In this way, users can draw more detailed paintings. Fig. 6.14 shows two states we set on the brush.


Manipulation Events

Manipulation Events

On Manipulation Started (ManipulationEventData)

Runtime Only | ClientTest.OnManipulate

pencil (ClientTe

On Manipulation Ended (ManipulationEventData)

Runtime Only | ClientTest.EndManipulate

pencil (ClientTe

Fig. 6.14 Two States of the Brush

Third, the synchronization of 3D painting can be divided into two parts: the person who paints and the person who watches. The part about network will be discussed in the server section at the end of this chapter. This section introduces the overall workflow of synchronization.

In the synchronization of 3D painting, the following types of information are synchronized:

1. The state of the brush.
2. The position of the brush.

Every time the user clicks on the brush to change the state, the state change information will be synchronized to the other user. Also, every time the brush is moved by the user and the coordinate changes, the new coordinate information will be synchronized to the other user.

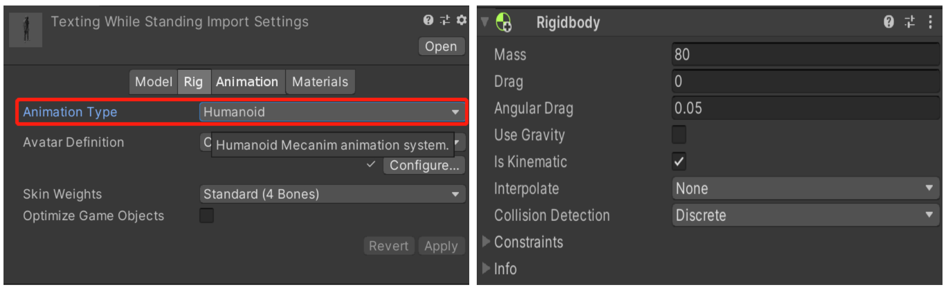
When the user receives the information of synchronizing the brush from the server, we wrote a "Client" script to process the information. In detail, when the information received is the state changing information, the system will change the state of the brush. When the new coordinate information is received, the system will determine whether it is consistent with the current coordinates, if not, the system will move the brush. If the brush is in the "paintable" state, the system will draw the paintings.

6.4 Users Action Simulation

6.4.1 Action Capturing and Binding

To achieve this function, we used Microsoft Kinect 2.0 Sensor. Kinect is a sensor including RGB cameras, infrared detectors and other components designed and manufactured by Microsoft. It can be used in fields such as user motion capture and gesture recognition. We used two Kinect 2.0 sensors, one for User A and the other for User B.

Fist of all, we created human avatars based on the front and side photos of the users. To make the avatar more realistic, we added some texture and clothes materials to the avatar. Then, we set the attributes of the avatar as humanoid and added a "Rigid Body" component to it to make it have physical characteristics. Fig. 6.15 shows these two settings.



Texting While Standing Import Settings

Animation Type	Humanoid
Avatar Definition	C Humanoid Mecanim animation system.
Skin Weights	Standard (4 Bones)
Optimize Game Objects	

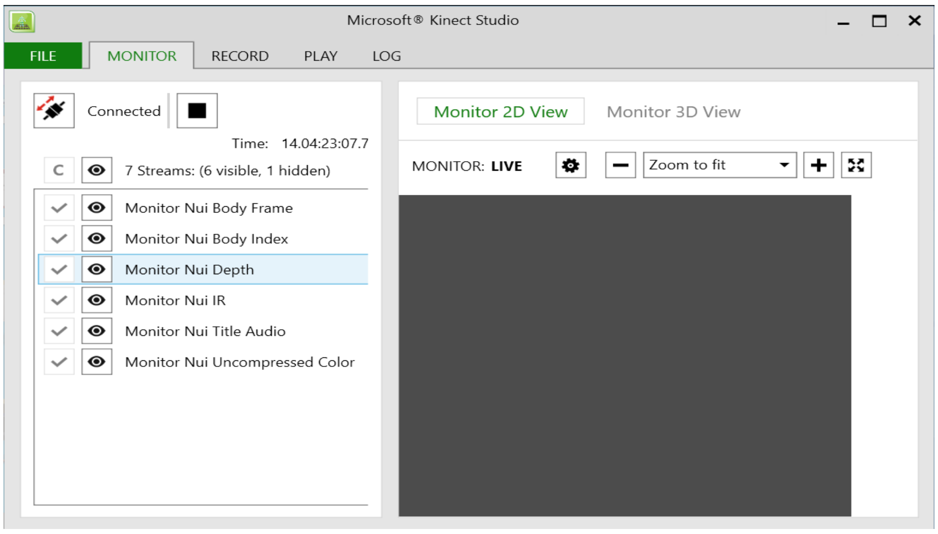
Rigidbody

Mass	80
Drag	0
Angular Drag	0.05
Use Gravity	
Is Kinematic	✓
Interpolate	None
Collision Detection	Discrete

(a) Set the Humanoid Attribute (b) "Rigid Body" Component

Fig. 6.15 User Avatar Settings

Next, we used "Kinect Studio v2.0" to connect the laptop to the Kinect sensor. Fig.6.16 shows the user interface of "Kinect Studio v2.0".



Time: 14.04:23:07.7

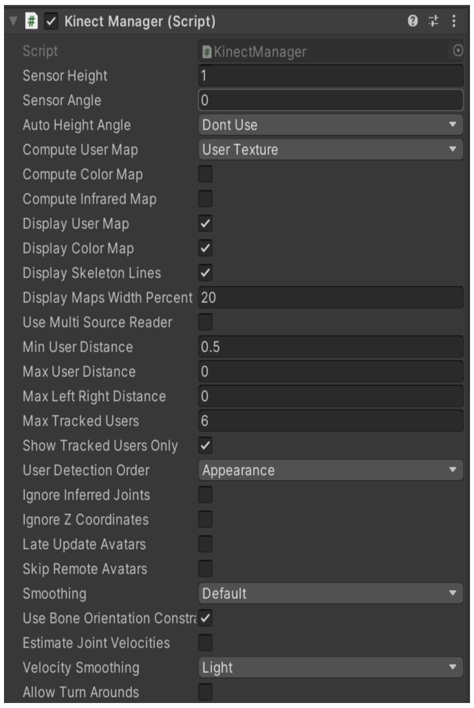
7 Streams: (6 visible, 1 hidden)

Monitor Nui Body Frame
Monitor Nui Body Index
Monitor Nui Depth
Monitor Nui IR
Monitor Nui Title Audio
Monitor Nui Uncompressed Color

Fig. 6.16 Kinect Studio

After that, we created an empty game object and added "Kinect Manager" script to it. This script can generate the user's action information from Kinect sensor. Then, we modified and added "Avatar Controller" script to the user's avatar. This script can bind the action information to the avatar and control the animation of the avatar. Fig. 6.17 shows these two scripts.

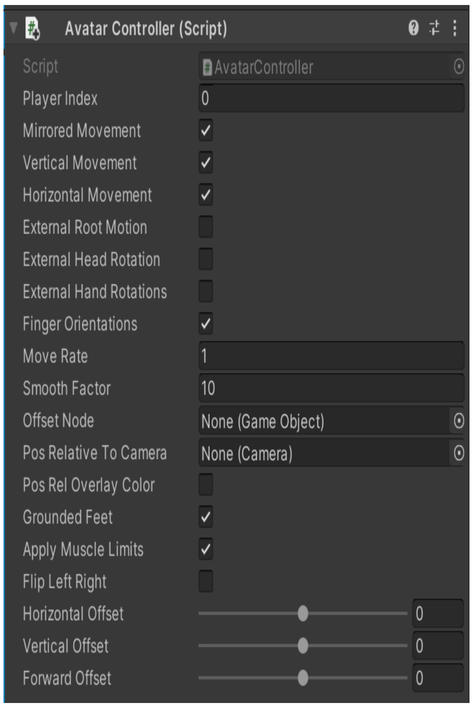
Last, since the avatar has a "Rigid Body" component, it will be affected by gravity. So we created a game object as the floor to prevent the avatar from falling down all the time. We created an empty game object and added "Mesh Collider" component to it. After that, we set the width, height and position of the game object to make it more like a floor.



Kinect Manager (Script)

Script	KinectManager
Sensor Height	1
Sensor Angle	0
Auto Height Angle	Dont Use
Compute User Map	User Texture
Compute Color Map	☐
Compute Infrared Map	☐
Display User Map	✓
Display Color Map	✓
Display Skeleton Lines	✓
Display Maps Width Percent	20
Use Multi Source Reader	☐
Min User Distance	0.5
Max User Distance	0
Max Left Right Distance	0
Max Tracked Users	6
Show Tracked Users Only	✓
User Detection Order	Appearance
Ignore Inferred Joints	☐
Ignore Z Coordinates	☐
Late Update Avatars	☐
Skip Remote Avatars	☐
Smoothing	Default
Use Bone Orientation Constra	✓
Estimate Joint Velocities	☐
Velocity Smoothing	Light
Allow Turn Arounds	☐

(a) Kinect Manager



Avatar Controller (Script)

Script	AvatarController
Player Index	0
Mirrored Movement	✓
Vertical Movement	✓
Horizontal Movement	✓
External Root Motion	☐
External Head Rotation	☐
External Hand Rotations	☐
Finger Orientations	✓
Move Rate	1
Smooth Factor	10
Offset Node	None (Game Object)
Pos Relative To Camera	None (Camera)
Pos Rel Overlay Color	☐
Grounded Feet	✓
Apply Muscle Limits	✓
Flip Left Right	☐
Horizontal Offset	0
Vertical Offset	0
Forward Offset	0

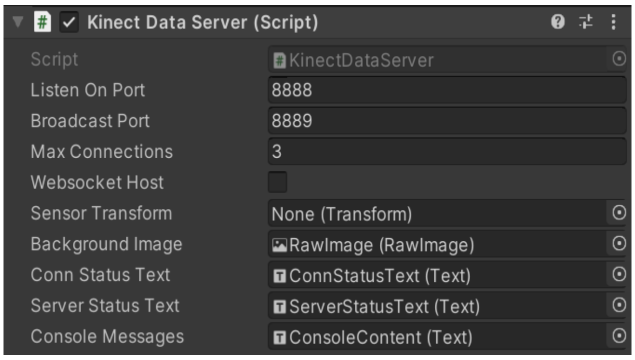
(b) Avatar Controller

Fig. 6.17 Generate and Bind User Action to the Avatar

6.4.2 Remote Action Information Transmission

In face-to-face communication in the real world, people can only see the full body of other people. Therefore, in this system, the user can only see the avatar and actions of the other user. In order to achieve this function, we used the network to transmit information about remote actions.

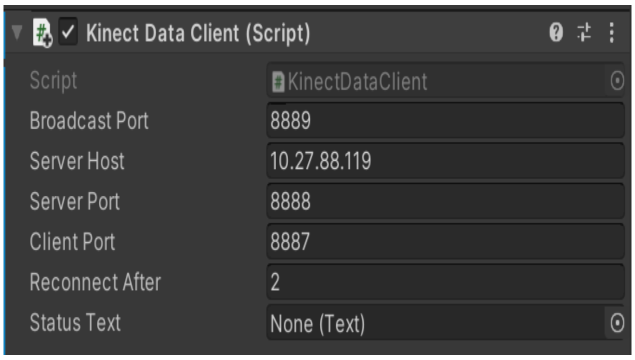
We used a laptop as the "Kinect Server", which is only used to transmit action information generated by Kinect sensor. We added a "Kinect Data Server" script to control the connection and message processing of the server. For the client side, which means the user's avatar on HoloLens in our system, we modified a "Kinect Data Client" script and added it to the avatar. Then we set the IP address, server host port number and broadcast port number for both the client side and the server side. Fig. 6.18 and Fig. 6.19 shows these two scripts.



Kinect Data Server (Script)

Script	KinectDataServer
Listen On Port	8888
Broadcast Port	8889
Max Connections	3
Websocket Host	☐
Sensor Transform	None (Transform)
Background Image	RawImage (RawImage)
Conn Status Text	ConnStatusText (Text)
Server Status Text	ServerStatusText (Text)
Console Messages	ConsoleContent (Text)

Fig. 6.18 Kinect Data Server



Kinect Data Client (Script)

Script	KinectDataClient
Broadcast Port	8889
Server Host	10.27.88.119
Server Port	8888
Client Port	8887
Reconnect After	2
Status Text	None (Text)

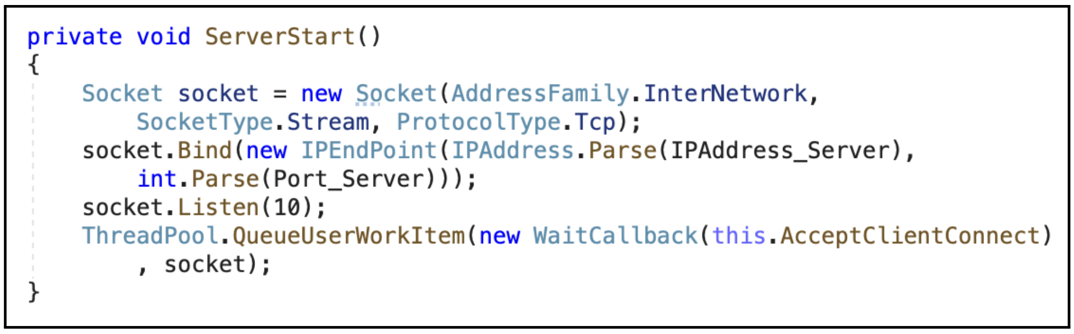
Fig. 6.19 Kinect Data Client

6.5 Server

6.5.1 Keep Listening and Establish Connections

Our system provides remote users with an immersive communication experience, so a server is needed to maintain the connection between users, send and receive messages, and process messages. We used Client-Server communication architecture and TCP protocol.

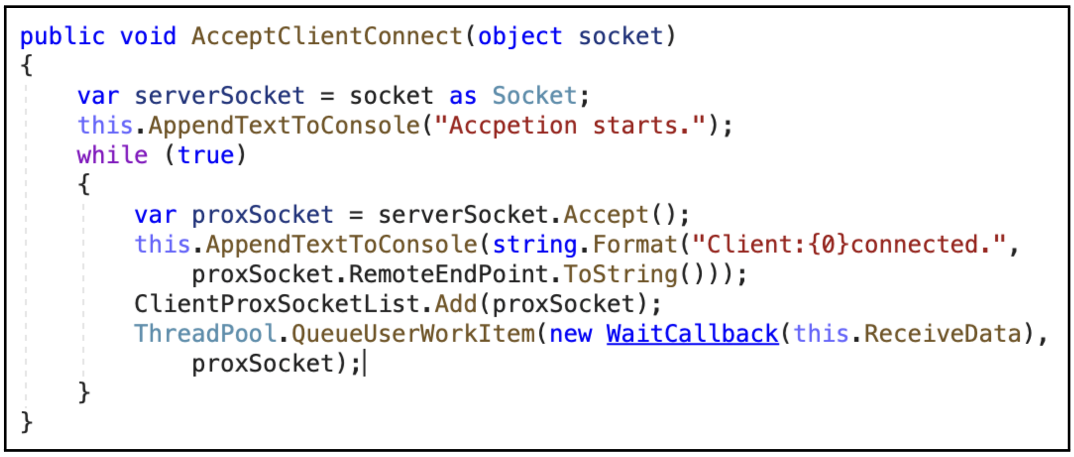
As a server, it needs to keep the listening state all the time to wait for the client's connection request. Because we use the TCP protocol, the connection request message sent by the client needs to be verified and processed before the server establishes a connection with the client. We bind the server process to port 5000 of the computer, made the server keep listening and wait for the connection. We set the maximum number of connections to 10 (in our system there are only two users, so there are only two connections). We used thread pool to enable multiple threads to wait for connection establishment. Fig. 6.20 shows these steps.



```
private void ServerStart()
{
    Socket socket = new Socket(AddressFamily.InterNetwork,
        SocketType.Stream, ProtocolType.Tcp);
    socket.Bind(new IPEndPoint(IPAddress.Parse(IPAddress_Server),
        int.Parse(Port_Server)));
    socket.Listen(10);
    ThreadPool.QueueUserWorkItem(new WaitCallback(this.AcceptClientConnect)
        , socket);
}
```

Fig. 6.20 Keep Listening and Establish Connections

Then, we used asynchronous threads to establish connections and keep receiving messages. If we do not use asynchronous threads, when the server establishes a connection with a client at the first time, the main thread can only receive messages from this client. If no messages are sent, the main thread will be blocked, making other clients unable to establish connections with the server. After that, the connection are established and the server will keep on the state of waiting and receiving data from the client. Fig. 6.21 shows these steps.



```
public void AcceptClientConnect(object socket)
{
    var serverSocket = socket as Socket;
    this.AppendTextToConsole("Accpetion starts.");
    while (true)
    {
        var proxSocket = serverSocket.Accept();
        this.AppendTextToConsole(string.Format("Client:{0}connected.",
            proxSocket.RemoteEndPoint.ToString()));
        ClientProxSocketList.Add(proxSocket);
        ThreadPool.QueueUserWorkItem(new WaitCallback(this.ReceiveData),
            proxSocket);
    }
}
```

Fig. 6.21 Use Asynchronous Threads to Establish A Connection

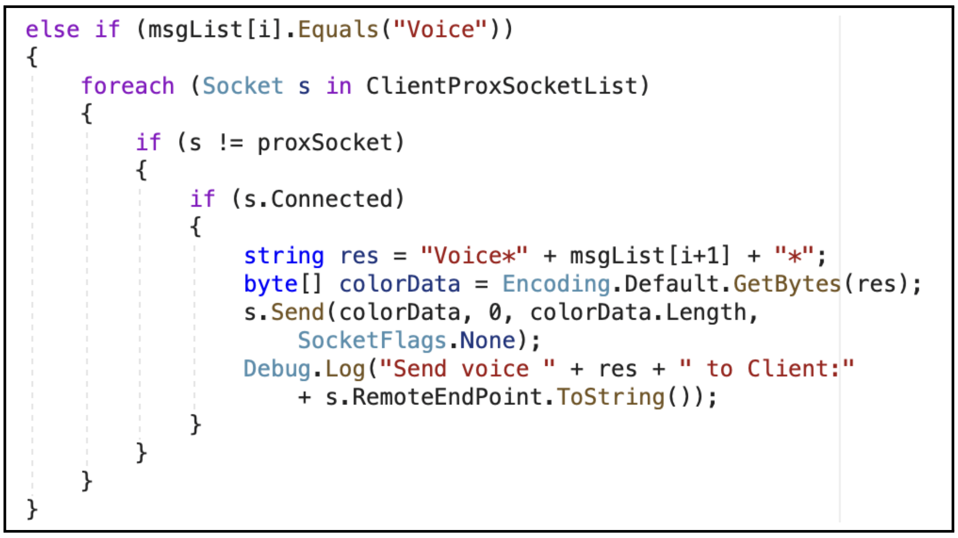
6.5.2 Realize Real-Time Voice Chatting Function

To realize real-time voice chatting function, we first designed the format for theses messages as:

$$Voice * Content *$$

Here, we used "*" as a delimiter to split the message. "Voice" is the message type, which means this is a voice message. "Content" is the binary content of the voice message.

When the server receives this type of message from the client, the server splits the message with "*" first to distinguishes the type, and then forwards it to the other user. After the user receives this type of message from the server, the type is first identified, and then processed. The binary data is encoded into audio stream and distributed to the audio player through the transceiver for playback.



```
else if (msgList[i].Equals("Voice"))
{
    foreach (Socket s in ClientProxSocketList)
    {
        if (s != proxSocket)
        {
            if (s.Connected)
            {
                string res = "Voice*" + msgList[i+1] + "*";
                byte[] colorData = Encoding.Default.GetBytes(res);
                s.Send(colorData, 0, colorData.Length,
                    SocketFlags.None);
                Debug.Log("Send voice " + res + " to Client:"
                    + s.RemoteEndPoint.ToString());
            }
        }
    }
}
```

Fig. 6.22 Process Voice Messages

6.5.3 Realize the Synchronization of 3D Painting Function

In the process of synchronizing 3D painting, we need to synchronize two kinds of information: the state of the brush and the position of the brush.

For the state of the brush, the message format is:

$$Flag * State *$$

Here, "Flag" means the type of this message is the state of the brush. "State" is a number, which can be zero or one.

1. When "State" is ’0’, it means the brush is in the "unpaintable" state and cannot draw anything.
2. When "State" is ’1’, it means the brush is in the "paintable" state and can draw lines.

When the server receives this types of messages, it will forward it to the other user. When the user receives this type of messages, it will set the state of the brush based on the message.

For the position of the brush, the message format is:

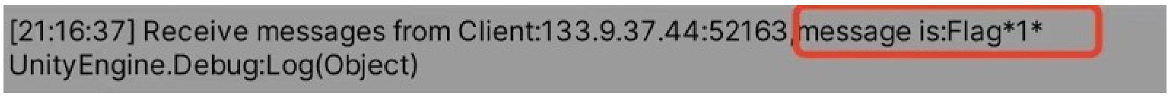
$$Pos * Vector *$$

Here, "Pos" means the type of this message is the new position of the brush. "Vector" is a "Vector3" type attribute in Unity, which is the new coordinate of the brush. The server forwards the messages as well, the user processes this type of messages based on the state of the brush. When the brush is in the "unpaintable" state and the new coordinate is different from the current coordinate of the brush, the brush will be move to the new position without drawing anything.

On the other hand, when the brush is in the "paintable" state and the new coordinate is different from the current coordinate of the brush, the brush will be move to the new position and the movement trajectory will be drawn.

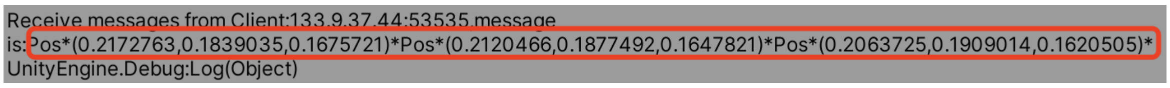
What's more, since the accuracy of Unity's "localPosition" method is 0.1 meters and is low, it will seriously affect the synchronization of the drawn paintings. So we used the scale factor N (N=10,0000). When the client sends coordinate information, it will multiply the coordinates by N to improve the accuracy. When the client receives the coordinate information, it will divide the coordinates by N to obtain the real coordinates, thus avoiding the accuracy problem of the Unity "localPosition" method.

Fig. 6.23 and Fig.6.24 show these two types of messages.



```
[21:16:37] Receive messages from Client:133.9.37.44:52163,message is:Flag*1*
UnityEngine.Debug:Log(Object)
```

Fig. 6.23 State Message



```
Receive messages from Client:133.9.37.44:53535,message
is:Pos*(0.2172763,0.1839035,0.1675721)*Pos*(0.2120466,0.1877492,0.1647821)*Pos*(0.2063725,0.1909014,0.1620505)*
UnityEngine.Debug:Log(Object)
```

Fig. 6.24 Position Message

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In terms of design, our research compared current remote communicating applications, then proposed a new remote communicating method, "Scene Re-experience".

Technically, we enhanced current remote communicating experience by "Scene Re-experience", which means re-experiencing recent events in the virtual environment to get a better understanding. We used the method of converting 2D video into 3D animation to include more spatial information, so that users can have an immersive feeling. Other then that, we provided users with real-time voice chatting function and 3D painting function to communicate better. Last, in order to improve users' sense of realism, we used user avatars to simulate users' actions in real time, so that the system can provide users with a feeling of they are talking to their partners face-to-face.

Overall, we think we have successfully accomplished these following requirements.

1. Providing real time voice chatting function in virtual world;
2. Using 3D animation to include more spatial context;
3. Providing more communicating methods;
4. Providing a more authentic experience by user avatars.

7.2 Future Work

Although we provided a new method, "Scene Re-experience", to enhance remote communicating experience, the company of family and friends in the real world is still indispensable, and the system still has some limitations to be improved.

For the 2D video to 3D animation converting part, we still need to use some other tools and our system cannot automatically achieve this function. A possible improving way is to integrated some tools into our system. The other way is to learn and achieve some converting algorithm in our system.

On the other hand, our system needs to be connected to many devices when using it, such as servers and Kinect Sensors. Network latency still has a significant impact. Also, when using HoloLens 2 indoors, it may trip over too many cables, which may cause safety hazards. To solve these problems, we can optimize the server logic and reduce the amount of data transmission. To prevent users from tripping, we recommend that users sit on a sofa or chair to use this system and reduce movement, just like when communicating with friends face-to-face.

References

[1] Sonja Utz. Media use in long-distance friendships. *Information, Communication & Society*, 10(5):694–713, 2007.

[2] Xincan Yang. Port-all: Reproduction of the sense of companionship in a long-distance relationship. 2021.

[3] Fernando Poyatos. Interactive functions and limitations of verbal and nonverbal behaviors in natural conversation. 1980.

[4] Fernando Poyatos. Language and nonverbal behavior in the structure of social conversation. *Language Sciences*, 4(2):155–185, 1982.

[5] Donghee Shin. How does immersion work in augmented reality games? a user-centric view of immersion and engagement. *Information, Communication & Society*, 22(9):1212–1229, 2019.

[6] Benjamin B Bederson and Angela Boltman. Does animation help users build mental maps of spatial information? In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 28–35. IEEE, 1999.

[7] Thomas Waltemate, Dominik Gall, Daniel Roth, Mario Botsch, and Marc Erich Latoschik. The impact of avatar personalization and immersion on virtual body ownership, presence, and emotional response. *IEEE transactions on visualization and computer graphics*, 24(4):1643–1652, 2018.

[8] Tencent. Wechat. https://www.wechat.com/, 2021.

[9] LINE Corporation. Line. https://line.me/en/, 2021.

[10] Ruofei Du, David Li, and Amitabh Varshney. Geollery: A mixed reality social media platform. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[11] Mayo Morimoto, Yosuke Motohashi, and Sawako Mikami. Development of vr lecture system for speaker and audience at the conference. In *Proceedings of Asian CHI Symposium 2019: Emerging HCI Research Collection*, pages 23–30, 2019.

[12] Tuomas Kantonen, Charles Woodward, and Neil Katz. Mixed reality in virtual world teleconferencing. In *2010 IEEE Virtual Reality Conference (VR)*, pages 179–182. IEEE, 2010.

[13] Thammathip Piumsomboon, Arindam Day, Barrett Ens, Youngho Lee, Gun Lee, and Mark Billinghurst. Exploring enhancements for remote mixed reality collaboration. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, pages 1–5. 2017.

[14] Jie Li, Vinoba Vinayagamoorthy, Julie Williamson, David A Shamma, and Pablo Cesar. Social vr: A new medium for remote communication and collaboration. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2021.

[15] Costas Boletsis. Virtual reality for prototyping service journeys. *Multimodal Technologies and Interaction*, 2(2):14, 2018.

[16] Toni Alatalo, Timo Koskela, Matti Pouke, Paula Alavesa, and Timo Ojala. Virtualoulu: collaborative, immersive and extensible 3d city model on the web. In *Proceedings of the 21st International Conference on Web3D Technology*, pages 95–103, 2016.

[17] Thomas Hilfert and Markus König. Low-cost virtual reality environment for engineering and construction. *Visualization in Engineering*, 4(1):1–18, 2016.

[18] Mark Billinghurst, Adrian Cheok, Simon Prince, and Hirokazu Kato. Real world teleconferencing. *IEEE Computer Graphics and Applications*, 22(6):11–13, 2002.

[19] Hideaki Kuzuoka. Spatial workspace collaboration: a sharedview video support system for remote collaboration capability. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 533–540, 1992.

[20] Cha Zhang, Qin Cai, Philip A Chou, Zhengyou Zhang, and Ricardo Martin-Brualla. Viewport: A distributed, immersive teleconferencing system with infrared dot pattern. *IEEE MultiMedia*, 20(1):17–27, 2013.

[21] Ruofei Du, David Li, and Amitabh Varshney. Project geollery. com: Reconstructing a live mirrored world with geotagged social media. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019.

[22] Junyuan Xie, Ross Girshick, and Ali Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016.

[23] Liang Zhang, Carlos Vazquez, and Sebastian Knorr. 3d-tv content creation: automatic 2d-to-3d video conversion. *IEEE Transactions on Broadcasting*, 57(2):372–383, 2011.

[24] Nicholas Howe, Michael E Leventon, and William T Freeman. Bayesian reconstruction of 3d human motion from single-camera video. *Advances in neural information processing systems*, 12:820, 2000.

[25] Leonid Sigal and Michael J Black. Predicting 3d people from 2d pictures. In *International Conference on Articulated Motion and Deformable Objects*, pages 185–195. Springer, 2006.

[26] Dongsik Jo, Ki-Hong Kim, and Gerard Jounghyun Kim. Effects of avatar and background types on users' co-presence and trust for mixed reality-based teleconference systems. In *Proceedings the 30th Conference on Computer Animation and Social Agents*, pages 27–36, 2017.

[27] Dongsik Jo, Ki-Hong Kim, and Gerard Jounghyun Kim. Spacetime: adaptive control of the teleported avatar for improved ar tele-conference experience. *Computer Animation and Virtual Worlds*, 26(3-4):259–269, 2015.

[28] Michael F Deering. Holosketch: a virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3):220–238, 1995.

[29] Kinetix SAS. Kinetix studio. https://www.kinetix.tech/, 2021.

[30] Microsoft. Microsoft mixedreality-webrtc. https://microsoft.github.io/MixedReality-WebRTC/index.html, 2021.

[31] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[32] iMMcque Inc. Meice. https://vcore.hk/home.html, 2021.

[33] 3DLOOK Inc. 3d look. https://3dlook.me/, 2021.