The Pennsylvania State University

The Graduate School

SYSTEM DESIGN AND IMPLEMENTATION OF COLLABORATIVE MIXED-REALITY-BASED FIREFIGHTER TRAINING

A Thesis in

Electrical Engineering

by

Zhanchen Dong

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2022

The thesis of Zhanchen Dong was reviewed and approved by the following:

Bin Li
Associate Professor of Electrical Engineering
Thesis Advisor

Mahanth Gowda
Assistant Professor of Computer Science Engineering

Thomas La Porta
Professor of Electrical Engineering
Director, School of Electrical Engineering and Computer

ABSTRACT

Mixed reality (MR) is an approach to the hybrid of the physical world and the digital world. It expands the possibilities of the people's realistic surroundings rather than simply supplementing information or adding virtual items as an overlay. In this work, we develop an MR system that facilities firefighter training. It uses a classic server-client architecture, which is widely adopted in multi-user video games. For the accurate displaying of MR contents, we build a virtual laboratory, a 3D model with the same scale as the real one. This virtual laboratory is deployed on a server with a simulated fire accident (e.g., virtual flames and smoke), where each client corresponds with an avatar inside to represent its movement and action. This avatar is used to determine visible contents that will be sent to the corresponding client. Users can see these virtual flames and smoke in the proper position through mobile mixed-reality-compatible devices. Besides, users can put out flames through a virtual fire extinguisher and see the change in temperature through a temperature visualization bar through the user interface.

To further improve the synchronization between all connected clients (i.e., all clients should see the same status of the virtual fire accident), we propose a latency compensation algorithm. This algorithm can reduce the time difference of receiving the same message from the server for clients under different network conditions, in other words, decreasing the range of latency between the server and all clients. This algorithm can also reduce the extremes of latency so that more frames in the client can receive new information to update rendering. The latency compensation algorithm can decrease the maximum latency difference by 14.3% to 26%. However, the overall latency between clients and the server increases by about 32.5%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Foremost, I would like to express my deepest appreciation to my advisor and mentor, Dr. Bin Li, who gave me the golden opportunity to do this project. His dynamism, vision, sincerity, and motivation have greatly inspired me. He has taught me the methodology to carry out the research. It was a privilege and honor to work and study under his guidance over the past two years. I am extremely grateful for what he has offered me. I could not have imagined having a better advisor and mentor for my M.S. study.

I would also like to extend my deepest gratitude to Dr. Mahanth Gowda. I am very honored that he could join my M.S. committee. His course has not only benefited and inspired me greatly but will also become a valuable asset in my life.

My sincere thanks also go to other members of Smart Network and Computing Lab: Jiangong Chen, Xudong Qin, Xinyi Yao, Chih-Hsuan Sun and Xiaoyi Wu for the stimulating discussions, for all the problems you have helped with and for the fun we have had.

Last but not least, I would like to my parents for giving birth to me at the first place and supporting me spiritually through my life.

This material is based upon work supported by the National Science Foundation under Grant CNS-2152658. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

Chapter 1

Introduction

In recent years, virtual reality (VR) and augmented reality (AR) have brought people many new experiences in media, entertainment, education, and more fields. VR technology uses computers to generate a completely virtual environment in which users can interact with their surroundings through devices such as helmets and glasses. Compared with VR, AR technology emphasizes the connection between users and the physical environment by overlaying computer-generated content on reality. However, it is still difficult to blur the border between reality and virtualization with these two technologies. Thus, mixed reality (MR) is proposed to solve the challenge of merging the real and virtual worlds [1].

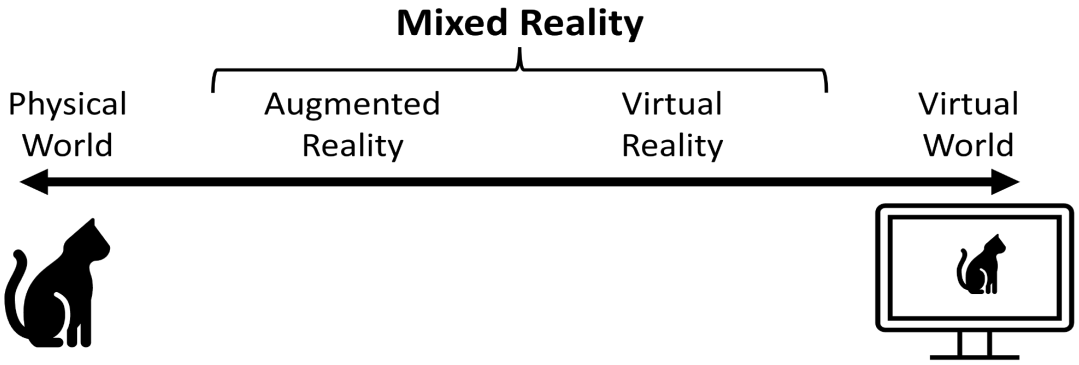

Figure **1-1**: Reality – Virtuality Continuum

The first published definition of "Mixed Reality" was introduced by Paul Milgram and Fumio Kishino in 1994 [2]. The "Reality – Virtuality Continuum" chart (Figure 1-1) in their work describes the spectrum from the complete physical world to the complete virtual world. Based on this continuum figure, MR covers the range of VR and AR. To classify VR, AR and MR more

precisely, Marc Parveau and Mehdi Adda provided a method called “3iVClass” [3]. Its main idea is summarized in the following figure (Table 1-1). Wolfgang Honig *et al.* proposed three specific requirements for a MR system: 1) combination of physical and virtual objects in the corresponding environment; 2) real-time interaction; 3) spatial mapping between physical and virtual objects [4].

Table **1-1**: “3iVClass” Method

	Augmented Reality	**Virtual Reality**	**Mixed Reality**
Immersion	Augmentation of the real world using virtual annotations	Fully virtual environment	Spatial mapping in real time allowing data contextualization
Interaction	Mediate interaction with physical objects	Mediate Interaction with virtual objects	Immediate interaction with virtual and physical objects
Information	Virtual annotation within the real environment; Not time-persistent; Decorrelated from user space	Virtual object registered in 3D space; Not time-persistent; Decorrelated from user space	Virtual object registered in 3D space; Time-persistent; Correlated to the user space

For VR and AR technologies, both hardware and software are so mature that many commercial devices available are for common customers, like Meta Quest 2, HTC VIVE Pro 2, and Microsoft HoloLens 2. However, each of them has its own limitations when applied in a collaborative system. MR technology has great potential in this area because of its ability to merge the physical world with the virtual world. However, the development of collaborative MR is still in the early stages of technical discussions and specialized applications. We design and implement a collaborative MR-based firefighter training in this thesis with a latency compensation algorithm to improve the synchronization among clients. We hope they are helpful for people when developing collaborative MR systems.

1.1 Problem Statement

Some kinds of traditional training can be dangerous. According to the National Fire Protection Association (NFPA), 7550 injuries were incurred during training activities in 2020 [8]. These risks usually come from real flames, smoke and other toxic chemicals, even if firefighters are wearing protective equipment, which is inevitable. In addition, setting up a new firefighter training facility can cost years of time and millions of money. For example, in Polk, Florida, a new firefighter training facility is expected to cost $14.5 million and will take more than two years to complete by 2024 [16]. Considering the potential risk, and high cost of time and money, available options for training locations are always finite.

Therefore, we design and implement the collaborative MR-based firefighting training. This system can be deployed in any place with an accurate 3D model immediately without causing damage. Trainees can use a virtual fire extinguisher to put out a fire and observe the temperature change through a temperature visualization bar. This system also allows people to supervise trainees' actions on the server in the 3D virtual training locations. Moreover, we understand that every second counts in firefighter training. Thus, to improve the synchronization of clients' collaborative work, we propose a latency compensation algorithm to minimize the time difference between clients receiving the same message from the server.

1.2 Thesis Organization

This chapter gives an overview of this thesis. Chapter 2 covers the background information of some necessary development tools for MR applications, some examples of collaborative MR, and some VR-based and AR-based firefighting applications. Chapter 3 first introduces a general

MR system and then provides the details of our MR system design of collaborative fighter training, including the execution logic of the server and the clients. Chapter 4 proposes a latency compensation algorithm, which is used to improve the synchronization performance among all clients. We discuss why it is necessary in our system and how it work in detail. The performance evaluations of frame rate, latency, “glitch” and the trade-off of this algorithm are given in this chapter as well. The final chapter concludes our work in this thesis and lists some potential future directions.

Chapter 2

Related Work

In this chapter, we first introduce some development tools for MR applications. Then, there is a literature review on two topics: collaborative MR, including remote expert, shared workspace, and shared experience, and some AR-based and VR-based firefighting applications.

2.1 Development Tools

Developing an MR application from scratch is very difficult, especially regarding display, rendering, and tracking. Even if the developer has a target device for display in advance, how to render virtual objects and track real objects or even recognize environments are all open and complicated issues. Therefore, we need software development kits (SDKs) with more complete functions to help us deal with these problems. Many companies owning MR-compatible devices have launched their SDKs, like ARKit [5] of Apple, ARCore [6] of Google, and MRTK 2 [7] of Microsoft, to support and facilitate the development of MR content. The major software toolkits and their tracking methods, development languages, and supported platforms are listed in Table 2-1.

Table **2-1**: MR Software Toolkit

	Tracking Method	**Supported Platform**	**Development Language**
Vuforia	Marker based visual SLAM	Android, iOS, Windows	C++, C#, Objective-C, Java
Kudan	Visual SLAM	Android, iOS	Objective-C, Java
Mi	Visual SLAM	Android, iOS, macOS, Windows	C, C++, Objective-C, Java
ARKit	Visual and depth SLAM	iOS	Objective-C
ARCore	Visual and depth SLAM	Android, iOS	Java
Mixed Reality Toolkit	Visual, depth and orientation-based SLAM	Windows, Universal Windows MR (HoloLens)	C++, C#, JavaScript

2.2 Mixed-Reality-Based Collaboration

Collaboration technology is mainly about researching how people work together in a digital way. In 1968, Engelbart and English proposed what was perhaps the first collaborative system that enabled people to share screens and edit text in a video conference [11]. Based on the collaborative scenarios, the most popular collaborative MR can be divided into three dimensions: remote expert, shared workspace, and shared experience. The basic definitions and applications of these collaborative scenarios are given below:

2.2.1 Remote Expert

Remote expert aims to solve remote collaboration, especially when it is asymmetric and synchronous. The typical case is one or more knowledgeable people guiding local people for a

physical task at a distance, which usually requires a high level of expertise. For example, an expert from ASML in the Netherlands can guide a TSMC worker in Taiwan by overhauling of a high-precision, highly integrated machine on an assembly line from his office. In the boom of MR collaboration development in the last few years, the field of remote expert is the most popular one, accounting for most of the growth in papers over the years.

2.2.2 Shared Workspace

Shared workspaces focus on how virtual workspaces can be used to extend physical workspaces. In general, shared workspaces include collocated and remote types. For example, the application in [12] introduces how collocated shared workspace technology has been applied in the collaborative exploration of an archeological excavation. IllumiShare in [13], on the other hand, is an example of a remote shared space. Through multiple sets of cameras and pico-projectors, users in this system can share virtual and physical objects with each other. With technological advances in rendering and computational photography, shared workspaces are becoming a up-and-coming area for MR collaboration.

2.2.3 Shared Experience

The purpose of shared experience is to help people share their own experiences or knowledge. Although sometimes the way to share is through a specific piece of work, it presents a different approach and variety than remote experts. Similar to shared work, there are two types of shared experiences: collocated and remote. In addition to physical distance, their differences are also reflected in the styles of shared content. For example, the prototype of Facedisplay in [14]

aims to allow local collaborators to see virtual content that would otherwise be visible only to the sharers themselves. Remote shared experience, on the other hand, is biased toward aiding interpersonal relationships. For example, the system in [15] allows people in different locations to visit a virtual museum and share their experiences simultaneously.

2.3 Firefighting Applications

Compared with MR, AR and VR are more widely used in the field of firefighting, while their application scenarios are slightly different. The former is often used to help firefighters observe information in the fire scene and displayed on AR-compatible devices. The latter is often used to create a virtual space in which firefighters can conduct pre-set training by interacting with virtual objects.

An embedded system can process videos captured by cameras in the firefighter's personal protective equipment through deep learning algorithms and then visualize the parts worth firefighters' attention through AR [17]. Z.Xu et al. present a VR fire training simulator with a visualization technique to display the whole process of smoke evolution of smoke and an integrated assessment model of smoke hazards [18]. Moohyun Cha et al. focus on the real-time simulation of invisible physical quantities in VR fire training, which is based on computational fluid dynamics [19]. Sidh is a VR firefighter training system developed by using video-games-related techniques, simulating some common cases of fire recurring in a virtual cave [20]. They showed Sidh is helpful for firefighter students as a complement to traditional training.

Chapter 3

System Architecture and Design

In this chapter, we first discuss the general MR system, which consists of four parts: server, client, cloud, and network. Based on this, we design a system of collaborative MR-based firefighter training. On the server, we build a virtual 3D laboratory on the server for physical simulation and mapping. Each client has an avatar in this virtual laboratory. The server updates these avatars based on the information from clients to figure out visible contents and then sends the status of them back to clients.

3.1 General Mixed Reality System

The existing MR systems vary widely because many MR applications are in the experimental or beta stage and the hardware of MR devices is highly independent. Therefore, we conclude a relatively general MR system, shown in Figure 3-1. It consists of four parts: server, client, cloud, and network.

The server manages mapping, execution logic, data processing, physical simulation, communication management and so on. Mapping in MR means projecting virtual objects in the real world or real objects in the virtual world. The function of mapping requires accurate spatial information about the physical world. In our firefighter training system, we use a 3D model of our laboratory to solve this issue. Sometimes, the physical simulation needs the information of the real space to simulate the background. This 3D virtual laboratory is used as the background in simulation as well. The client is usually deployed on smartphones and headsets. It is mainly used to render MR contents, collect, and send users' information to the server, and interact with users

through the user interface (UI). A point worth paying attention to is that the smoothness of the client display, in other words, the frame rate, primarily affects the user's experience. Under normal circumstances, 30Hz and above can avoid the feeling of stuttering, and 60Hz and above can bring a very smooth experience. The frame rate is usually based on the number and quality of the 3D contents it has to render, which consumes the most computing power. In some specific applications, to reduce the computation consumption of mobile devices, these 3D contents will be rendered on the server and then sent to the client directly as the result of rendering [10]. As for the cloud, sometimes the system needs access to information on the internet, like weather, maps, or online media. The network enables communication among these parts to support the MR system working properly. Like other systems with communication, the MR system usually uses transmission control protocol (TCP) or user datagram protocol (UDP) as its communication protocol.

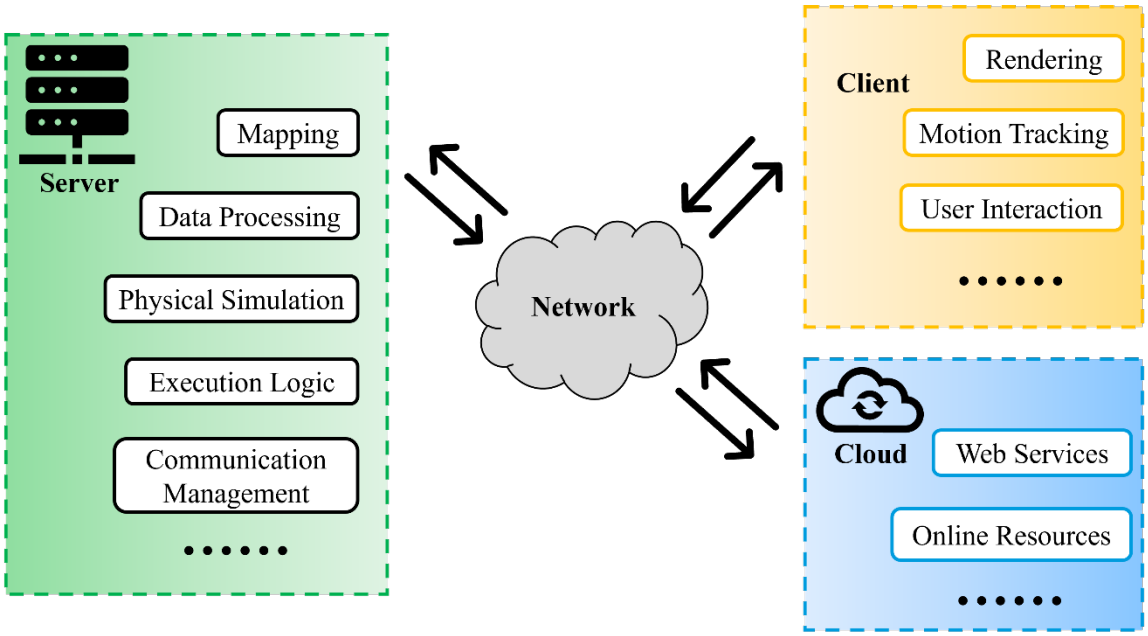

Figure **3-1**: General MR System

3.2 System of Firefighter Training

The hardware architecture is shown in Figure 3-2. We apply the classic server-client architecture, which is widely adopted in LAN multi-user video games [21]. This architecture is very flexible in adding and removing clients. It is also convenient to maintain the stability of the system when there is a small number of connected devices.

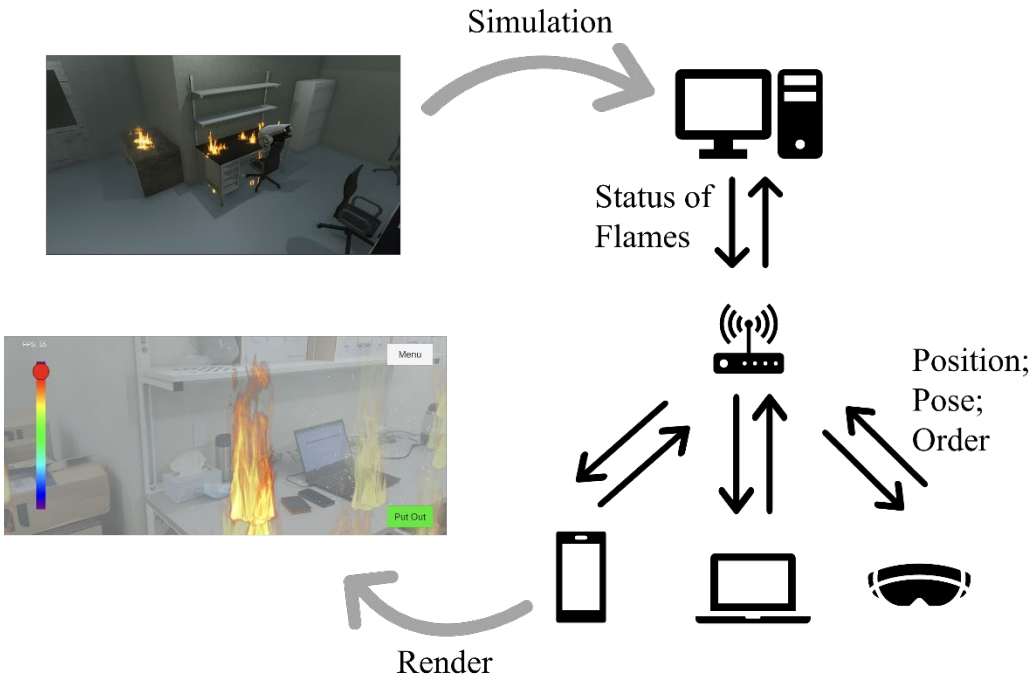

Figure **3-2**: Hardware Architecture

The system of our collaborative MR-based firefighter training is shown in Figure 3-3. Compared with the general MR system, it eliminates the part of the cloud and consists of the remaining three parts: server, client, and network. The user takes a mobile phone as the client.

Through ARCore, data collected from IMU are converted to position and pose information. This information, then combined with the user's commands, like putting out fires, is sent back to the server. After splitting the message from the clients, position and pose data are used to update the avatar in the 3D virtual laboratory to determine visible contents. Commands are applied to adjust the size of the chosen flames. The network communication protocol is TCP, which ensures the successful delivery of data and messages. The detail of latency compensation will be introduced in the next chapter.

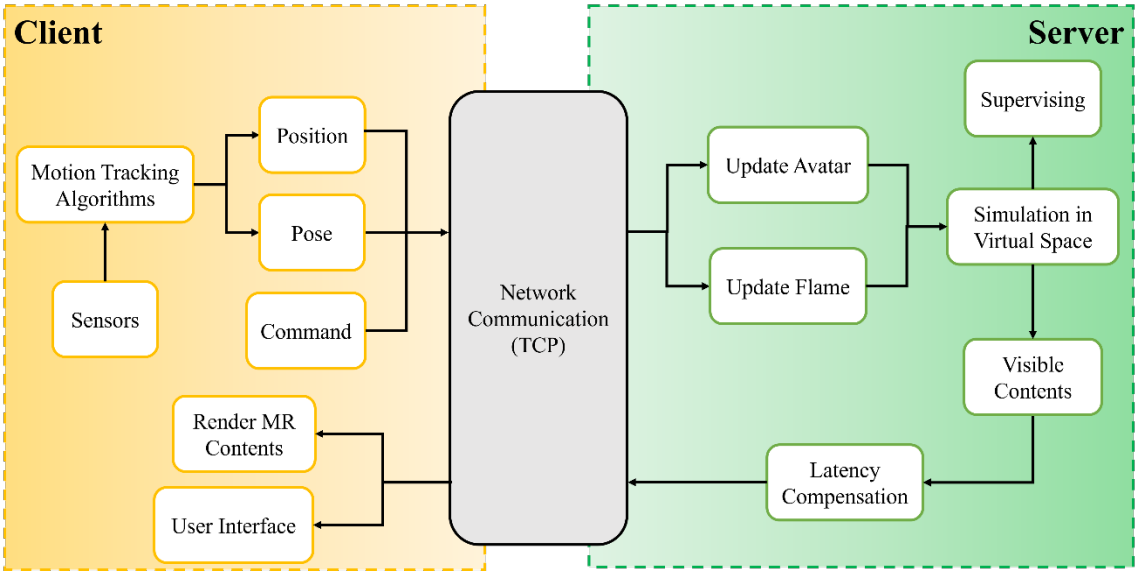

Figure **3-3**: System Design

3.2.1 Server

Hardware: Dell Precision 7920 workstation (Intel Xeon Gold 6242R CPU @ 3.10Ghz, 128 GB RAM, Windows 10 Enterprise)

As we discussed earlier, in order to create an accurate mapping between MR content and the physical world, we need to build a 3D model of the training space that is identical to the real one. Thus, we build a virtual laboratory on the server through ProBuilder, a Unity 3D modeling tool. As is shown in Figure 3-4(a)(b), the virtual laboratory has the same scale and facilities, like tables, chairs, and cabinets, as our real laboratory. Besides, people can supervise or record trainees' actions and movements for further usage with a visible virtual space. In this virtual laboratory, we set some flames around tables and smoke covering the whole room for fire simulation. These flames and smoke are generated by particle systems in Unity.

Whenever a client connects, the server generates and initializes a corresponding virtual avatar (Figure 3-5(a)) in this virtual laboratory. Through messages transmitted from the client, the server can update the pose and position of this avatar and figure out all flames and smoke in its field of view (Figure 3-5(b)). Because all rendering of MR contents is done on the client side, when too many contents are rendered at the same time it can cause a significant increase in computations on the mobile device, thus increasing the rendering time per frame and reducing frame per second (FPS). In addition, when the entire scene is complex, such as a multi-story building with several rooms, rendering MR contents in other rooms or even floors not only wastes computing power of mobile devices but also confuses the user. Thus, in each frame, the server needs to calculate the field of view of each avatar, collect their visible contents, like flames, and then send the status of these contents to different clients with the compensated latency.

Figure **3-4(a)**: Picture of Real Laboratory

Figure **3-4(b)**: Picture of 3D Virtual Laboratory

Figure **3-5(a)**: The Avatar in 3D Virtual Lab

Figure **3-5(b)**: Avatar's View

3.2.2 Client

Hardware: Google Pixel 6 (Google Tensor, 8 GB RAM, Android 12)

There are now many mature motion tracking algorithms for mobile devices. In this application, we use ARCore to deal with the data collected from Inertial Measurement Unit (IMU) sensors. The client reads the latest data once per frame and sends its position and pose to the server 50 times per second. Sometimes the client's frame rate will be below 50hz because of some unexpected fluctuations, resulting in not enough different messages per second to send. In this case, for those communications between two frames, we repeat sending the status of the last frame. The 3D models of smoke and flame are prestored in the mobile phone application. The client can render, update, and adjust these MR contents using these models after receiving the message from the server. We also add a temperature visualization function in the client. The circle's position and color in the left colorful bar represent the user's current surrounding temperature. It is related to the scale of the flames and the distance between the user and the flames. The UI is shown in Figure 3-6.

Figure **3-6**: Client's UI

Chapter 4

Latency Compensation Algorithm and Performance Evaluation

4.1 Latency Compensation Algorithm

For most multi-user systems, people pay more attention to the synchronization between the server and clients. However, in some collaborative scenarios, the synchronization among clients is more critical. Especially in our collaborative MR-based firefighter training, trainees should see all changes in the training space simultaneously, even a tiny change of flame.

In the system architecture mentioned earlier, the client updates its rendering based on the latest information from the server, which is generated from the simulation of fire in the virtual laboratory. Although this simulation is driven by the information from each client, there is no direct communication between clients, and even local changes need to be simulated by the server before they can be rendered. Thus, the time difference in transmission latency from the server to each client is usually the leading cause of asynchronization among clients. The purpose of this algorithm is to reduce such an asynchronization and ensure each client to receive the same message from the server at approximately the same time.

4.1.1 Analysis of Latency in Communication

In the most ideal case, since there is no latency, any communication between the server and the client is instantaneous, and there is no asynchronization. Based on this, it would be a relatively ideal situation if each client has the same latency of communicating the server without any fluctuations. Figure 4-1(a) shows the ideal communication situation. There are *n* clients

$C_1, C_2, \ldots, C_n$ connected to the server. The server sends the first message *msg1* to all clients at time TS_1. The time of client C_k receiving this message *msg1* is $TR_{1,k}$. The latency between the server and the client C_k of this term of communication is $L_{1,k}$. Now, for the client C_k we have the relationship between the send time TS_i, the receive time $TR_{i,k}$, and the latency $L_{i,k}$:

$$TR_{i,k} = TS_i + L_{i,k}$$

Because all n clients have the same sending time TS_1 and the same latency values ($L_{1,1} = L_{1,2} = \ldots = L_{1,n}$), the receive time will be the same ($TR_{1,1} = TR_{1,2} = \ldots = TR_{1,n}$) as well. Therefore, in this case, all clients can receive the same message from the server at the same time.

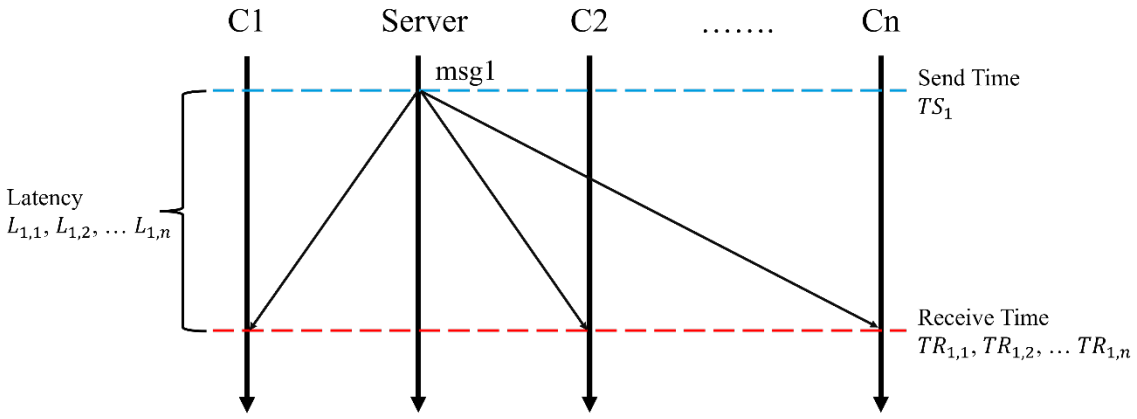

Figure **4-1(a)**: Ideal Communication

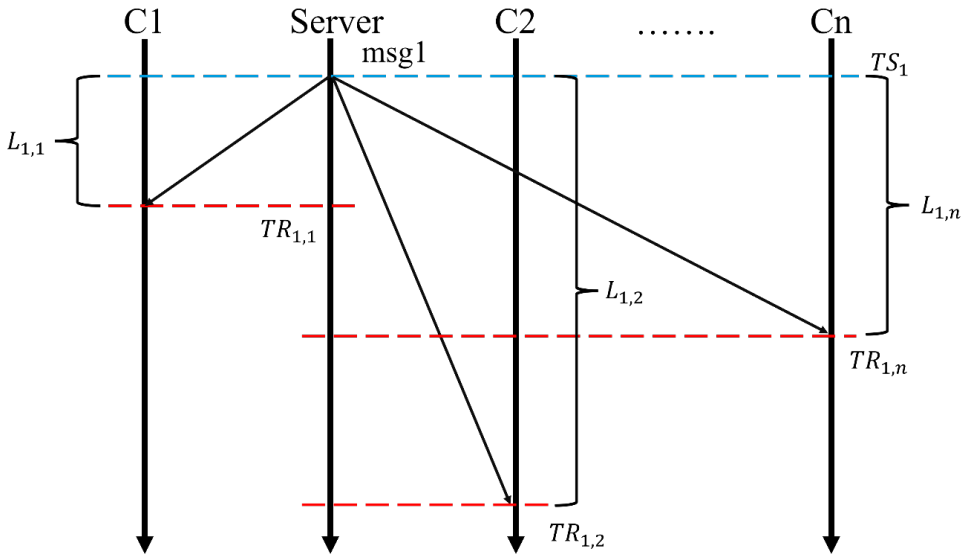

Figure **4-1(b)**: Real Communication

For real-world communication, clients have heterogenous latency of communicating with the server. The heterogeneity of latency is caused by shared resource, maintenance activities, queueing, daemons and so on [22]. Figure 4-1(b) shows the real communication situation. The server still sends the first message *msg1* to all clients at time TS_1. However, in this case, each client receives the same message from the server at different times ($TR_{1,1} \neq TR_{1,2} \neq \ ... \neq \ TR_{1,n}$) due to different latency ($L_{1,1} \neq \ L_{1,2} \neq \ ... \neq \ L_{1,n}$). To evaluate the synchronization between clients, we need to find the range of all receive time, in other words, the difference between the maximum value and the minimum value of all receive times in the same term of communication. Its mathematical expression is given below:

$$Range_i = Max(TR_{i,1}, \dots, TR_{i,n}) - Min(TR_{i,1}, \dots, TR_{i,n})$$

The send time TS_i is the common element for all receive time, which can be eliminated during calculation. Thus, the expression for the range is modified to:

$$Range_i = Max(TR_{i,1}, \dots, TR_{i,n}) - Min(TR_{i,1}, \dots, TR_{i,n})$$

$$= Max\left(TR_{i,1}, \dots, TR_{i,n}\right) - TS_i - \left(Min\left(TR_{i,1}, \dots, TR_{i,n}\right) - TS_i\right)$$

$$= Max\left(L_{i,1}, \dots, L_{i,n}\right) - Min(L_{i,1}, \dots, L_{i,n})$$

4.1.2 How to Compensate

Because the asynchronization is caused by the latency difference, in order to make each client receive the same message from the server at approximately the same time, we need to minimize this difference. Our idea is to add extra time to those smaller latencies as compensation so their values can get closer to the maximum value.

This method is shown in Figure 4-2. When the server is about to send the first message *msg1* to the client C_k at time TS_1, it postpones sending this message for a latency compensation $LComp_{1,k}$. This latency compensation is defined by:

$$LComp_{i,k} = Max(L_{i,1}, \dots, L_{i,n}) - L_{i,k}$$

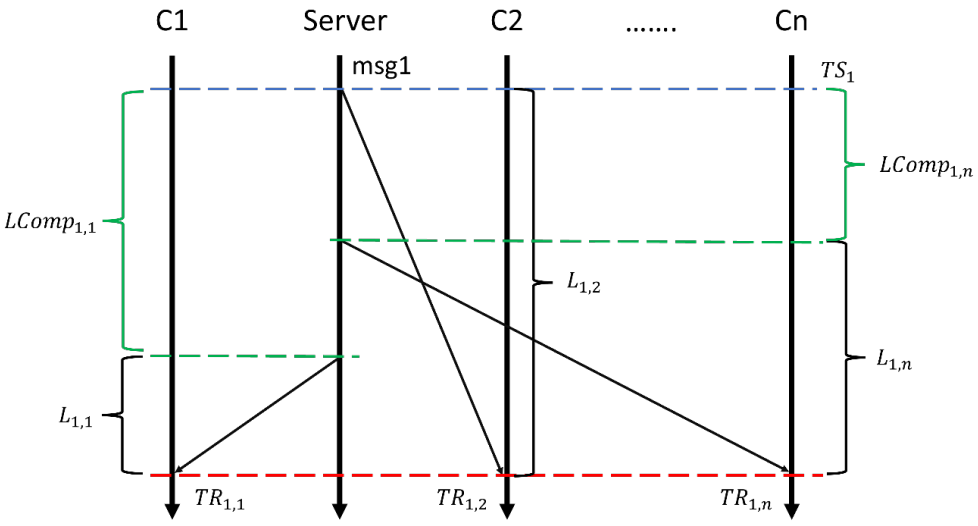

Figure **4-2**: Communication After Latency Compensation

Now, the receive time for the client C_k is

$$TR_{i,k} = TS_i + L_{i,k} + LComp_{i,k}$$

However, for TCP communication, the latency between the server and the client cannot be measured directly. Usually, half of the round-trip time (RTT) can be considered as the latency [23]. We cannot get the RTT until a whole communication process is completed. Besides, a single latency is meaningless because the fluctuations of latency in network communication are ubiquitous. Thus, we need to make use of the historical data of latency instead of the latency for this term of communication to calculate the latency compensation. Although the latency is unpredictable, we can use the running average to indicate its trend. There are four main kinds of running average: simple running average (SRA), cumulative average (CA), exponential running average (ERA), and weighted running average (WRA). The mathematical expressions of these four methods are given below

$$SRA_{i+1} = SRA_i + \frac{1}{k}(p_{i+1} - p_{i-k+1}),\ k\ is\ sampling\ width$$

$$ERA_{i+1} = ERA_i + \sigma(p_i - ERA_i), \sigma\ represents\ the\ degree\ of\ weighting\ decrease$$

$$CA_{i+1} = CA_i + \frac{p_{i+1} - CA_i}{i+1} = \left(1 - \frac{1}{i+1}\right)CA_i + \frac{1}{i+1}p_{i+1}$$

$$WMA_i = \frac{np_i + (n-1)p_{i-1} + \cdots + 2p_{((i-n)+2)} + p_{((i-n)+1)}}{n + (n-1) + \cdots + 2 + 1}$$

Through these four methods, we have the moving-averaged latency $LAvg_{i-1,k}$ for C_k of receiving the message msg_i. Now the latency compensation algorithm is

$$LAvg_{i-1,k} = Moving\ Average(L_{1,k}, \dots, L_{i-1,k})$$

$$LComp_{i,k} = Max(LAvg_{i-1,1}, \dots, LAvg_{i-1,n}) - LAvg_{i-1,k}$$

$$TR_{i,k} = TS_i + L_{i,k} + LComp_{i,k}$$

$$Range_i = Max(TR_{i,1}, \dots, TR_{i,n}) - Min(TR_{i,1}, \dots, TR_{i,n})$$

$$= Max(L_{i,1} + LComp_{i,1}, \dots, L_{i,n} + LComp_{i,n}) - Min(L_{i,1} + LComp_{i,1}, \dots, L_{i,n} + LComp_{i,n})$$

4.2 "Glitch" of Frames

In our system, the client needs information from the server to update MR contents for each flame. In most cases, the frequency of the server sending messages to clients is much higher than the frame rate of clients, and each frame has new information to update rendering (Figure 4-3(a)). However, sometimes the latency can change abruptly. In other words, the extreme value of latency occurs, resulting in some frames not having new information for updating (Figure 4-3(b)). In this case, the frame missing the information from the server will be a "glitch". When the frame rate is around 60Hz, a glitch or two in a second will not matter. However, if the glitch occurs too often, it can cause very noticeable stuttering.

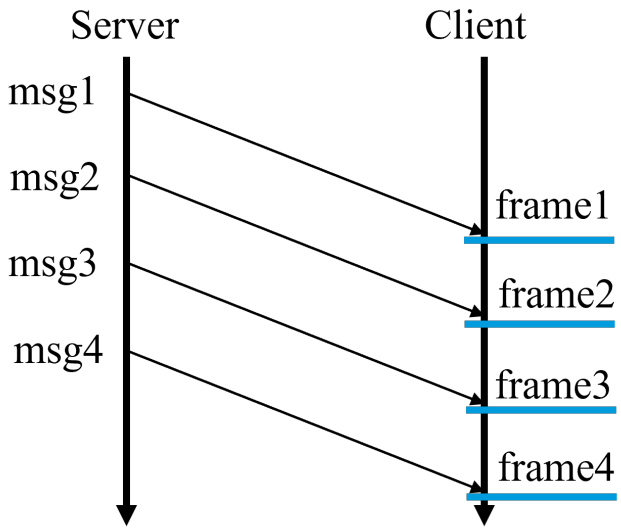

Figure **4-3(a)**: Each Frame with Information

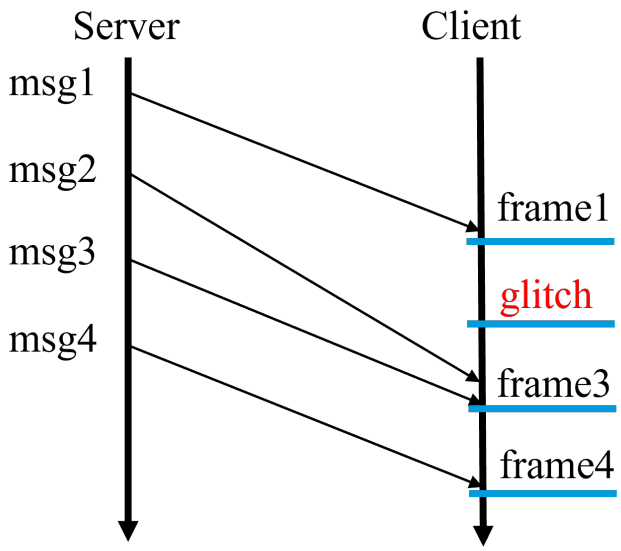

Figure **4-3(b)**: "Glitch" Frame without Information

Since the occurrence of "glitch" is caused by the extreme value of latency, we analyze the distribution of latency to evaluate the "glitch" of frames. Therefore, we use the standard deviation, skewness, and kurtosis to check this issue. The primary usages [24] and mathematical expressions of them are given below:

- Standard deviation is a measure of the amount of variation or dispersion of a set of values. Its mathematical expression is:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^3}$$

where μ is mean. A low standard deviation indicates that the values tend to be close to the mean.

- Skewness is a measure of the asymmetry of the probability distribution. Its mathematical expression is:

$$S = \frac{1}{n}\sum_{i=1}^{n}(\frac{X_i - \mu}{\sigma})^3$$

where μ is mean and σ is standard deviation. The closer the skewness to 0, the more symmetric the distribution is.

- Kurtosis is a measure of the tailedness of a distribution. The kurtosis of the normal distribution is 3. Its mathematical expression is:

$$K = \frac{1}{n}\sum_{i=1}^{n}(\frac{X_i - \mu}{\sigma})^4$$

where μ is mean and σ is standard deviation. A distribution with kurtosis > 3 is fat-tailed, meaning it has more outliers (extreme values).

4.3 Performance Evaluation

4.3.1 Frame Rate

For clients in a collaborative system, keeping the camera working and rendering MR content usually requires a high computing power. Especially in this application, we use the particle system in Unity to simulate fire and smoke, which makes scenes more vivid at the cost of high GPU usage. We synchronize the playing time of all the flames' animations, which helps the client to keep the FPS around 60Hz when rendering only the flames.

However, when smoke is applied, there is a significant decrease in frame rate to 35Hz. The main reason lies in that Unity's default rendering order is according to the distance between the

object and the main camera (i.e., rendering from the closest object to the main camera). Since when the smoke is applied, it is always the closest object to the camera, consuming limited computing power on each frame. The other reason is that the smoke needs to spread the entire virtual laboratory. The number of particles and the volume of its particle system distribution are much larger than the particle system of the flame, which increases amount of computation. Thus, to improve the frame rate when the smoke is applied, we move its rendering order to the last on each frame and reduce its particle number without significantly affecting visual performance. As the result, the final FPS with smoke is generally above 50 Hz.

4.3.2 Latency

To simulate a more realistic and complex network status, we make use of a dataset containing cloud network variability [9]. This dataset contains bandwidth variability data for Amazon EC2, Google Compute Engine, Microsoft Azure, Scaleway, SURFsara and HPCCloud; and TCP latency data for Amazon EC2 and Google Compute Engine.

We pick 8 groups of data from the TCP latency part of this database. To avoid some unnecessary interference, such as initialization of communication, we intercept 12,000 samples from the middle segment of these data, as is shown in Figure 4-4. The mean values of these 8 groups are 35.05ms, 30.99ms, 30.15ms, 22.82ms, 32.11ms, 35.00ms, 39.00ms and 34.86ms.

The comparison between the range of compensated latency and the range of raw latency is in Figure 4-5. The mean value for the range of raw latency is 29.74ms. For the methods of SRA, CA, ERA, and WRA, the mean values of the range of the compensated latency are 23.55ms, 25.50ms, 22.68ms, and 21.98ms. Generally, the range of latency is decreased by 14.3% to 26% under our algorithm.

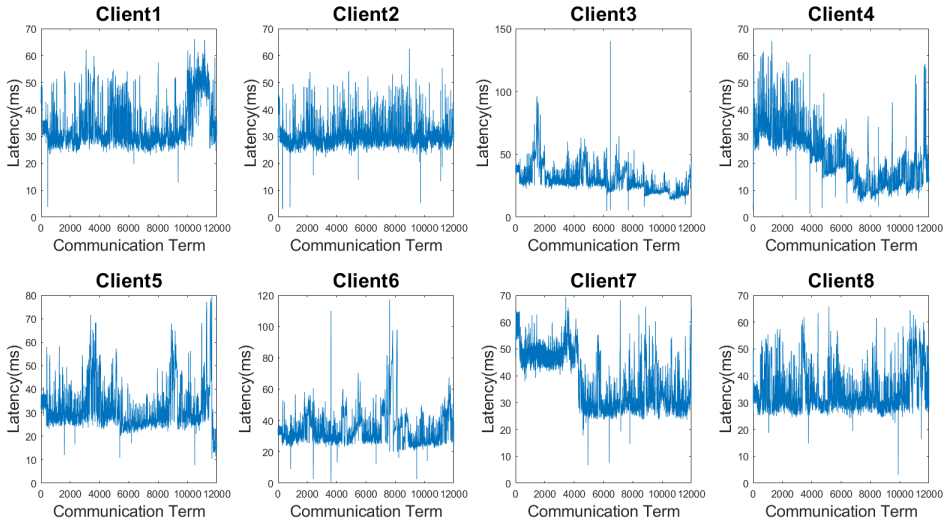

Figure **4-4(a)**: 8 Groups of Raw Latency

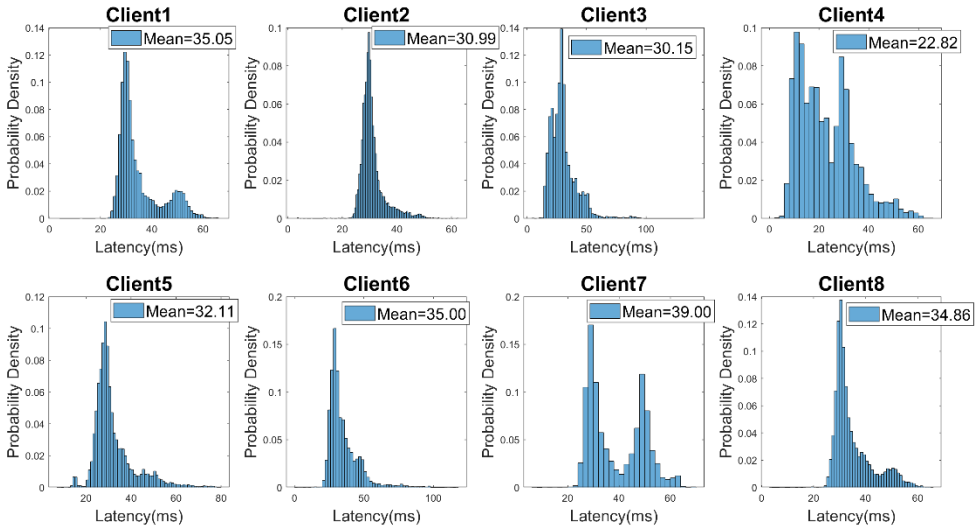

Figure **4-4(b)**: PDF of 8 Groups of Raw Latency

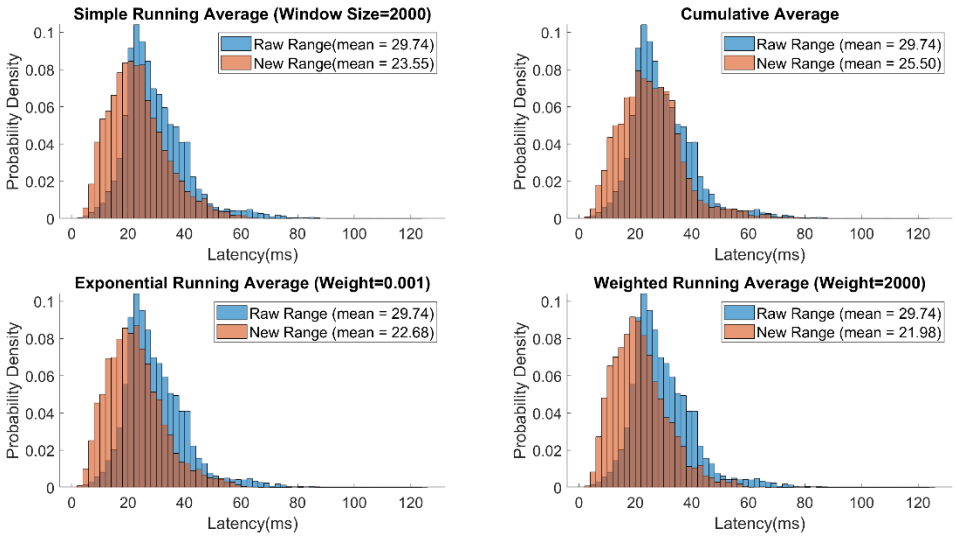

Figure **4-5**: Range of Compensated Latency and Raw Latency

4.3.3 "Glitch"

The comparison of standard deviations, skewness, and kurtosis of the range of compensated latency and raw latency is shown in Figure 4-6. For the standard deviations, all latency compensation methods have a better performance than the raw data, meaning that their latency values are closer to the mean value. For the skewness, all latency compensation methods' values are closer to zero than the raw data, so their distributions are more symmetric. For the kurtosis, since all four values are higher than 3, a larger value means more extreme values in its distribution. The kurtosis of the raw data is much larger than others, which means it causes more "glitch" of frames. Overall, for all these three evaluation methods, the data with latency compensation have better performance (i.e., our algorithm reduces extreme values of latency and decreases the occurrence of the "glitch" of frames).

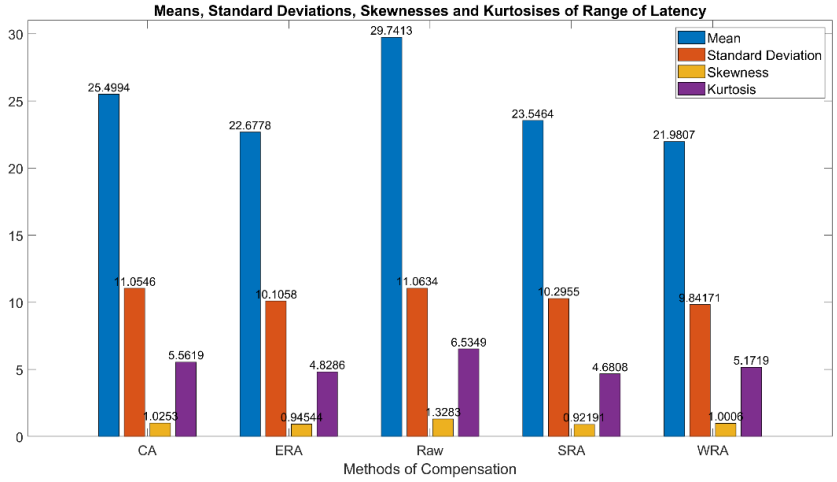

Figure **4-6**: Standard Deviation, Skewness and Kurtosis of Range of Latency

4.3.4 Trade-off

As a trade-off, the average latency between the server and clients is larger than before (Figure 4-7) after applying the latency compensation algorithm because all smaller latency has to increase to the maximum values.

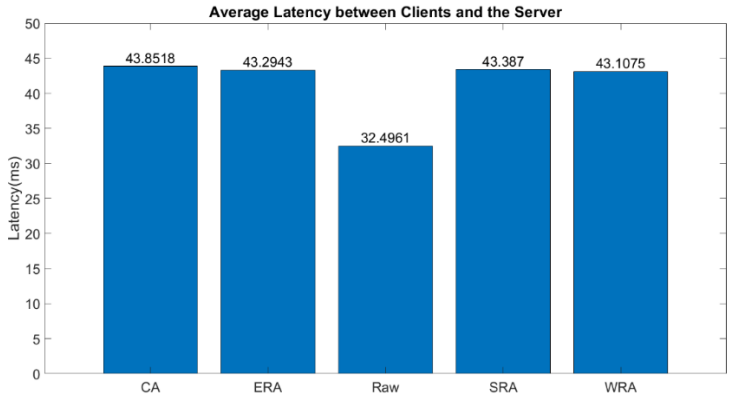

Figure **4-7**: Average Latency between Clients and the Server

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we first review some related works, including development tools and collaboration in MR. Then, we describe a general MR system consisting of server, client, network, and cloud. We further design and implement the system for firefighter training. Considering the flexibility and stability, we use the classic server-client architecture to organize our hardware. A 3D model of our virtual laboratory is established for the physical simulation, supervising, and accurate mapping of MR contents on the server. The client sends its position and pose to update the status of the corresponding avatar on the server. Besides, commands like putting out flames will be sent to the server to update the status of flames as well. After updating the simulation, the server figures out visible contents for the avatar and then sends their status to clients for rendering. The basic frame rate is around 60Hz. When the smoke is applied, it is still above 50Hz.

To improve the synchronization among clients in this system, we propose a latency compensation algorithm by adding extra latency manually on the server before it sends information to the client with lower latency. We use four running average methods to calculate the value of latency compensation. With our compensation algorithm, the range of latency decreases from 14.3% to 26%. Meanwhile, the data distribution is more concentrated towards the mean and the proportion of extreme values is reduced as well, decreasing the occurrence of the “glitch” of the frame. However, this is at the const of increasing the communication latency.

5.2 Future Work

Although the latency compensation algorithm improves the synchronization among clients, if there is a client with very high latency, all other clients have to increase their latency. This can cause potential problems like the supervising on the server cannot monitor trainees' actions in the first place. This issue requires some additional optimization for significant latency.

Besides, based on the implementation of our collaborative MR system, when rendering consumes too much computing power on mobile devices, it will cause noticeable visual stuttering for users. One possible solution is using streaming to send MR contents as video to the clients. However, this method is usually applied in traditional 2D rendering or VR rendering. To apply it in MR applications, we need to extract useful pixels and combine them with the reality captured by the camera.

REFERENCES

1. Rokhsaritalemi, Somaiieh, Abolghasem Sadeghi-Niaraki, and Soo-Mi Choi. "A review on mixed reality: Current trends, challenges and prospects." Applied Sciences 10, no. 2 (2020): 636.
2. Milgram, Paul, and Fumio Kishino. "A taxonomy of mixed reality visual displays." IEICE TRANSACTIONS on Information and Systems 77, no. 12 (1994): 1321-1329.
3. Parveau, Marc, and Mehdi Adda. "3iVClass: a new classification method for virtual, augmented and mixed realities." Procedia Computer Science 141 (2018): 263-270.
4. Hoenig, Wolfgang, et al. "Mixed reality for robotics." 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015.
5. Google LLC. 2022. ARCore. Retrieved October 24, 2022 from https://developers.google.com/ar
6. Apple Inc. 2022. ARKit 6. Retrieved October 24, 2022 from https://developer.apple.com/augmented-reality/arkit/
7. Microsoft Corporation. 2022. Mixed Reality Toolkit 2. Retrieved October 24, 2022 from https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05
8. Campbell, Richard, and Ben Evarts. "United States Firefighter Injuries in 2020." National Fire Protection Association (NFPA) (2021).
9. Uta, Alexandru, et al. "Is big data performance reproducible in modern cloud networks?." 17th USENIX symposium on networked systems design and implementation (NSDI 20). 2020.
10. Chen, Jiangong, et al. "Motion-prediction-based wireless scheduling for multi-user panoramic video streaming." IEEE INFOCOM 2021-IEEE Conference on Computer

Communications. IEEE, 2021.

11. Engelbart, Douglas C., and William K. English. "A research center for augmenting human intellect." Proceedings of the December 9-11, 1968, fall joint computer conference, part I. 1968.
12. Benko, Hrvoje, Edward W. Ishak, and Steven Feiner. "Collaborative mixed reality visualization of an archaeological excavation." Third IEEE and ACM International Symposium on Mixed and Augmented Reality. IEEE, 2004.
13. Junuzovic, Sasa, et al. "IllumiShare: sharing any surface." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2012.
14. Gugenheimer, Jan, et al. "Facedisplay: Towards asymmetric multi-user interaction for nomadic virtual reality." Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 2018.
15. Brown, Barry, et al. "Lessons from the lighthouse: collaboration in a shared mixed reality system." Proceedings of the SIGCHI conference on Human factors in computing systems. 2003.
16. Dustin Wyatt. 2022. Polk commissioner embarrassed by how county trains firefighters. New $14.5M facility will change that. Retrieved October 24, 2022 from https://www.theledger.com/story/news/local/2022/02/03/polk-county-build-14-5-million-fire-rescue-training-facility/9254293002/
17. Manish Bhattarai, Aura Rose Jensen-Curtis, and Manel Martínez-Ramón. 2020. An embedded deep learning system for augmented reality in firefighting applications. In 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 1224–1230
18. Zhen Xu, XZ Lu, Hong Guan, C Chen, and AZ Ren. 2014. A virtual reality based fire

training simulator with smoke hazard assessment capacity. Advances in engineering software 68 (2014), 1–8.

19. Moohyun Cha, Soonhung Han, Jaikyung Lee, and Byungil Choi. 2012. A virtual reality based fire training simulator integrated with fire dynamics data. Fire safety journal 50 (2012), 12–24.
20. Per Backlund, Henrik Engstrom, Cecilia Hammar, Mikael Johannesson, and Mikael Lebram. 2007. Sidh–a game based firefighter training simulation. In 2007 11th International Conference Information Visualization (IV'07). IEEE, 899–907.
21. Eric Cronin, Burton Filstrup, Anthony R Kurc, and Sugih Jamin. 2002. An efficient synchronization mechanism for mirrored game architectures. In Proceedings of the 1st workshop on Network and system support for games. 67–73.
22. Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. Commun. ACM 56, 2 (2013), 74–80.
23. Neal Cardwell, Stefan Savage, and Thomas Anderson. 2000. Modeling TCP latency. In Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064), Vol. 3. IEEE, 1742–1751.
24. N Alan Heckert, James J Filliben, C M Croarkin, B Hembree, William F Guthrie, P Tobias, J Prinz, et al. 2002. Handbook 151: NIST/SEMATECH e-handbook of statistical methods. (2002)