Intelligent Stereo Video Monitoring System for Paramedic Helmet

by

Yang Liu

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science
Faculty of Engineering, University of Ottawa
Ottawa, Canada

Abstract

During the first aid process, when patients are threatened by poor medical conditions, ambulance paramedics are required to administer emergency treatment based on instructions provided by a remote emergency doctor through voice communication. However, such voice communication is always limited in expressing abundant detailed information for the patient.

This thesis presents a framework for a stereoscopic and intelligent telemedicine system that can provide 3D live video communication between paramedics and emergency doctors. The proposed system captures 3D video from the paramedic headset carried by the paramedics, transmits the video through wireless live streaming, and displays the video with a 3D effect for emergency doctors in the hospital. The video can be analyzed to extract information about the patient through embedded algorithm such as face detection algorithm. In this thesis, the hardware, functional mechanism and face detection algorithm are introduced separately.

The hardware of the system consists of a paramedic headset, a server box and a 3D PC, which are used to capture 3D video, transmit video through live streaming and display video with a stereo effect, respectively. The functional mechanism includes two subsystems, which work for pushing the stereo video to multiple live streams and displaying the 3D video from the live stream. In order to detect the patient information from the video, a multi-task face detection algorithm is applied to analyze the stereo video using deep learning technology. We improved the neural networks of face detection by utilizing 1×1 convolutional layers and retrain the network based on the transfer learning to achieve better and faster performance.

This system has achieved good and stable performance in network delay (0.0489ms) and objective video quality evaluations. The face detection algorithm has achieved notable accuracy (91.78% In FDDB dataset) and efficiency (19.71 ms/frame).

Acknowledgements

I would like to express my sincerest appreciation and thanks to my supervisor, Professor Abdulmotaleb El Saddik, for his guidance, support and patience during my graduate experience. I would like to thank him for encouraging my research, giving me useful remarks and offering me help in the research of this master thesis. His trust in my abilities and the academic latitudes he provided have proved to be extremely invaluable during my M.A.Sc. study. He has provided far more than just the required supervisory feedback and excellent guidance throughout the journey of the research and the production of this work.

I would like to thank all the members in MCRLab for their useful help for my research and thoughtful comments on my thesis. Sincere thanks go to Dr. Haiwei Dong for the invaluable assistance and guidance provided throughout my research.

I would like to thank my family and Shuang Xie, who have supported me throughout the entire process of striving towards my goal in Canada. This work is dedicated to them.

Table of Contents

List of Figures

Chapter 1

Introduction

The time period that lasts for one hour or less following a traumatic injury being sustained by a casualty or medical emergency is vital to improve the likelihood of patient survival and could prevent death through careful medical treatment [26]. When patients are threatened by poor medical conditions, ambulance paramedics are required to administer emergency treatment based on instructions provided by a remote emergency doctor. Although paramedics always have varying levels of training in different resuscitation procedures and opportunities to utilize these skills, there is often a need to directly communicate by phone with an emergency physician to discuss the proper action for a complex patient in need of resuscitation. Such scenarios include major trauma, cardiac arrest, and respiratory emergencies, among others.

However, such voice communication is always limited in expressing abundant detailed information for the patient. The patient's information is helpful to provide targeted treatment for the medical emergency. Consequently, researchers focus on developing video communication between paramedics and emergency doctors to enhance the treatment procedure for the telemedicine in the pre-hospital stage. Andrea Corradini *et al.* proposed a video streaming solution that facilitates dialogues between ambulance paramedics and the doctor in the emergency vehicle. [9] Companies such as ADLINK have also constructed ambulance-based telemedicine systems to provide wireless connections between ambulances and hospitals with live video streaming and complex medical facilities. [2]

In the current stage, the assistance solution for remote paramedics is limited to the ambulance. However, paramedics often need to administer emergency treatment to patients before transporting the patients to the ambulance. In this case, a telemedicine assistance solution is needed to provide video communication for the period between the first contact with the patients and transporting the patients to the ambulance.

The emerging augmented reality, virtual reality and 3D display devices also provide good platforms for telemedicine. Chen *et al.* provided an intuitive approach to collaborate in a 3D environment using HoloLens and communicate with the remote participant using Skype. [8] According to their demonstration study, their approach could also work for telemedicine, which could help doctors located in remote clinics better understand the situation through viewing the 3D video. The rapid development of image analysis technology also allows the telemedicine system to extract useful information for the doctor. Esteva *et al.* [10] proposed a dermatologist-level classification of skin cancer using image analysis technology. Hwang and Kim [18] explored a weakly supervised deep learning solution that is used to detect nodules in chest radiographs and lesions in mammograms. With the development of display technology and medical image analysis technology, it is possible to make the telemedicine system both stereoscopic and intelligent.

Based on the above considerations, we developed a telemedicine system that can provide 3D live video communication between paramedics and emergency doctors. To assist paramedics during their first contact with patients, our system promises solid 3D video communication for the paramedics by carrying a paramedic headset and a small

wireless server box. To allow doctors to better observe the situation of the patient, we developed a 3D player with an image analysis algorithm to display the 3D video with the patient information. In order to detect the patient information, we applied a state-of-the-art face detection algorithm to analysis the stereo image and extract the patient face.

The main contributions of our work are listed as follows:

- A telemedicine system with 3D video communication is developed. This telemedicine system assists paramedics during their first contact with patients.
- The hardware components of the telemedicine system are integrated, and these components include the paramedic headset, the server box and the 3D PC.
- Stereoscopic capture and display systems for telemedicine are developed, which are able to capture stereo video as multiple live streams and display the remote live streaming video with a 3D effect.
- A state-of-the-art face detection algorithm is applied to the system. The face detector is improved using feature pooling technology.

This thesis is organized as follows. Chapter 2 presents the background and related works of this research. The previously reported approaches are presented in this chapter. Chapter 3 first introduces the procedures of the system (user cases), and then the detailed information of each part is explained based on the hardware components. Chapter 4 introduces the functional mechanism of this system. This chapter focuses on the functional framework and its corresponding detailed information in this mechanism. Chapter 5 explains the proposed face detection algorithm. In Chapter 6, evaluations of the performance of the 3D video communication system and the face detection algorithm are presented. Chapter 7 concludes the thesis and discusses directions for future work.

Chapter 2

Related Work

Since our system constructs a telemedicine system with 3D video communication between paramedics and emergency doctors, the existing live streaming system in telemedicine is first introduced. The composition of the live streaming system is briefly introduced, including the video compression format and video transmission protocol. This chapter also introduces the methods for evaluating the video in the live stream. Since a self-developed face detector is applied in this system, a brief introduction of face detection is also included.

2.1 Telemedicine

This section introduces existing telemedicine systems that provide live streaming systems for paramedics and emergency doctors. It also shows the new technologies appeared in telemedicine.

Telemedicine uses information technology to monitor patients from a distance, which is also defined as telemonitoring. This approach helps to eliminate distance barriers and can improve access to medical services that would often not be consistently available in distant rural communities. Hospitals, clinics, and prisons have all used telemonitoring, as have ambulances equipped with systems connected to the receiving hospitals [30]. Telemedicine is also used to save lives in critical care and emergency situations.

2.1.1 Live streaming system in telemedicine

Video streaming technology has been extensively used in the field of telemedicine to provide clinical health care from a distance.

The use of video streaming in remote fetal medicine consultation was introduced by Ann Thuring *et al.*. [41] Their study showed that another medical opinion can be accessed through video streaming without transferring the patient to a regional center. Thus, the use of video streaming in remote fetal medicine could be a cost-effective solution and an alternative to face-to-face consultations. However, in their work, the telecommunication and video compression were limited: MPEG1 and MPEG2 were used to compress the video, while the bandwidths of the video streams were 1.5 Mbit/s (MPEG 1) and 2 Mbit/s (MPEG 2).

Andrea Corradini *et al.* proposed a video streaming solution that facilitates dialogues between ambulance paramedics and the doctor in the emergency vehicle. [9] Their study allowed the doctor in an emergency vehicle to begin treatment in the event of an accident or illness in collaboration with the paramedics on the ambulance service already in transit to the hospital. In their system, the paramedics on the ambulance service utilized a 4G audio-video head-mounted camera to enable dialogue between the personnel on the ambulance and the anesthesiologist in the emergency vehicle.

There are several companies in the field of telemedicine aiming to provide remote medicine consultation. For example, ADLINK has constructed an ambulance-based telemedicine system to provide a wireless connection between the ambulance and the hospital. [2] The emergency medical services are recorded by digital cameras, while the telemedicine system must be integrated with the hospital's patient information system. Their telemedicine system allows the doctor in the hospital to clearly know the patient's medical history and current treatment situation.

Today, the use of telemedicine has rapidly spread and is now becoming integrated into the ongoing operations of hospitals, specialty departments, home health agencies, private physician offices and consumer's homes and workplaces. Telemedicine greatly improves the quality, equity and affordability of healthcare worldwide.

2.1.2 New technologies in telemedicine

With the development of new technologies, such as virtual reality, immersive environments and haptic feedback, human information and all their senses can be transmitted to remote locations. New technologies are being developed, such as handheld biosensors, holographic three-dimensional imaging, and augmented human machine interaction with haptic devices.

Haptic technology provides the sense of touch for the doctor and is widely used in remote rehabilitation. Popescu proposed the application of client/server telemedicine in orthopedic rehabilitation with a multipurpose haptic control interface. [37] Lamberto *et al.* introduced a virtual-reality-based system to provide motor tasks for patients from a remote rehabilitation facility. [36] These remote rehabilitation systems enabled automatic and transparent patient data collection in the clinic to monitor patient progress and obtain evaluations over time.

The emerging augmented reality, virtual reality and 3D display devices also provide a good platform for telemedicine. Onur *et al.* used a lensless incoherent holographic microscope in telemedicine to improve light collection efficiency. [32] This microscope utilized a simple light-emitting diode (LED) and a compact optoelectronic sensor array to record lensless holograms of objects, which then enables rapid digital reconstruction

of regular transmission or differential interference contrast (DIC) images of the objects. Chen *et al.* provided an intuitive approach to collaborate within a 3D environment using HoloLens and communicate with the remote participant via Skype. [8] According to their demonstration study, this approach could also work for telemedicine, which could help doctors in remote clinics better understand the situation through viewing the 3D video.

The new technologies allow doctors to perceive more information than 2D images and data. Doctors in remote clinics could experience a more realistic field environment and provide a more accurate and effective treatment. Such technologies promise new directions in the evolution of telemonitoring.

In our system, we combine these new technologies with the live streaming system in telemedicine and construct a 3D video communication system for paramedics and emergency doctors.

2.2 The composition of the live streaming system

A live streaming system consists of video capture, compression, transmission and playback. In this section, the video compression and transmission technologies are introduced in terms of the video codec format and video transmission protocol.

2.2.1 Video codec format

Codec is short for encoder/decoder. A codec is used to compress the original data by encoding the raw video data into a bitstream for low-burden transmission.

One of the basic video compression methods is M-JPEG. An intuitive video compression approach is applied in M-JPEG, which is that each video frame or interlaced field of a digital video sequence is compressed separately as a JPEG image. This intra-frame compression has good quality for the video, particularly for frames whose contents have large changes. However, due to the limitation of intra-frame compression, the compression rates of M-JPEG are quite low and not suitable for the live streaming system. Consequently, inter-frame compression is applied to achieve higher compression rates.

The main representation of interframe compression is a series of compression methods proposed by the Moving Picture Experts Group (MPEG), which are the widely used h.262/MPEG-2 and h.264/MPEG-4 Advanced Video Coding (AVC) standards. [47] The oldest MPEG compression technology is MPEG-1, which uses macroblocks and motion vectors to reduce the amount of temporal redundancy in a video. MPEG-1 defines a set of the block with 16 $\times$ 16 pixels as the macroblocks, and the encoder compares the current frame with adjacent parts of the video from the anchor frame. Only if the block is changed would the data be updated and added to the new bitstream; otherwise, this block would not add new data to the bitstream. In this case, the video data are compressed not only by the properties of each frame but also through the inter-frame similarity. The h.262/MPEG-2 standard has adopted a similar compression strategy as MPEG-1, but it added support for interlaced video, which was the format used by analog broadcast TV systems. In this case, MPEG-2 was also used in the HDTV transmission systems and became the standard format for over-the-air ATSC digital television.

However, due to the continuous improvements in video quality, the previous compression methods could not compress high-quality video into a bitstream with a small size for real-time transmission. A new generation of video compression formats was developed. The main representation of this format is the h.264/MPEG-AVC standard. Compared with previous compression methods, this new standard could achieve a high compression rate (200:1) through many new features, such as using previously encoded pictures as references in a considerably more flexible way and using variable block-size motion compensation with block sizes as large as 16×16 and as small as 4×4. High-efficiency video coding (HEVC), also known as h.265/MPEG-H, further enhanced the video compression capabilities through many new features, such as partitioning a picture into coding tree blocks and further partitioning the coding tree blocks into multiple coding blocks. The high compression rates of these approaches have been proven by Ohm's work [34]. Note that higher compression rates require more time for compression, as shown in Grois's work [13]

In parallel with the open video coding standardization processes of MPEG, a few companies individually developed their own video codecs, such as VP8, VP9 and AV1.[38]

These technologies were often based partially on their own proprietary technologies and partially on variants of the state-of-the-art technologies used in their standardized counterparts.

In this thesis, we adopt the popular h.264 as the video compression format to achieve the real-time high compression rate for the stereo video.

2.2.2 The protocols for the live streaming system

There are three types of protocols that are widely used in live streaming systems, which are real-time streaming protocol (RTSP), real-time messaging protocol (RTMP) and hypertext transfer protocol (HTTP)-based protocols. HTTP-based protocols include HTTP Live Streaming (HLS) and Microsoft Smooth Streaming Protocol (MS-SSTR).

Different protocols have different features: HTTP-based streaming protocols are widely supported in browsers, but the nature of the protocol means that it adds substantial latency to the stream. RTSP and RTMP are both low-latency protocols, but these protocols always are not compatible with most online and local players. [24]

Various studies explored the streaming quality evaluation by collecting parameters at the application level. Wang *et al.* collected streaming data using RealTracer, a collection tool that measured the performance of real video networks whose streaming protocol was RTSP. [43] Mok *et al.* characterized the correlation between the application and network with the HTTP streaming protocol. [31] They identified the rebuffering frequency as the main factor responsible for the mean option score variance. French *et al.* implemented a modified Flash player that collected quality of experience and service data during RTMP streaming. [11] Their preliminary study indicated that bitrate in combination with either frame rate or bandwidth serves as an accurate indicator of quality of experience for RTMP videos. Laine *et al.* studied the application performance for h.264 video with different streaming protocols. [24] In their studies, RTMP appeared to retain the quality slightly longer than the others, whereas HLS was the most resilient, not breaking the connection once during the tests. RTSP/TCP may induce more variation in quality when packet loss is present with low latency.

Based on the aforementioned video codec formats and live streaming protocols, the

stereo video live streaming system adopts h.264 as the video codec format and RTSP as the live streaming protocol.

2.3 Video quality evaluation

Video quality is a characteristic for evaluating the distortion of a video passed through a video transmission or compression system. In this section, the video quality evaluation is introduced to evaluate the quality of a set of video sequences. Since the video quality can be evaluated objectively (using mathematical models) or subjectively (by asking users for their rating), the detailed video evaluation models are introduced in the following subsections.

2.3.1 Objective video quality assessments

The objective video models are mathematical models that approximate results from a subjective quality assessment, in which human observers are asked to rate the quality of a video. Based on the amount of information available about the original signal, the objective video quality methods can be divided into 3 groups: full-reference methods, reduced-reference methods, and no-reference methods.

Full-reference methods compute the quality difference by comparing the original video signal against the received video signal. Typically, every pixel from the source is compared against the corresponding pixel in the received video, while more elaborate algorithms may choose to combine the pixel-based estimation with other approaches, such as the sliding window approach. Reduced-reference methods evaluate the video quality based on the extracted features from the system, such as a video transmission with a limited bandwidth. No-reference methods assess the quality of a distorted video without any reference to the original signal. Both reduced-reference methods and no-reference methods are applied to systems for which it is difficult to capture the original video. Since this stereo system has the ability to capture the original video and because the full-reference methods are generally the most accurate at the expense of higher computational effort, the full-reference methods are mainly introduced in the following parts.

The most traditional full-reference methods are mean squared error (MSE) and peak signal-to-noise ratio (PSNR) [17] These methods are commonly used to measure the reconstruction quality of lossy compression and transmission codecs by calculating the pixel difference between the original video and the distorted video. However, the study of Wang *et al.* showed that the MSE and PSNR evaluation methods cannot account for information communication frameworks, such as a video communication system involving lossy compression, noise contamination, and packet loss. [44] Thus, more complex video quality analysis methods are applied to evaluate the video quality.

Since the frames in the video are highly structured, the structural similarity (SSIM) index [46] and multiscale SSIM (MSSSIM) index [45] are used to measure the structural information of the video. Compared with MSE and PSNR, which use the pixel difference, SSIM and MSSSIM compare the difference of the pixels in blocks between the original frame and distorted frame to the structure of the objects represented in the scene. In addition, SSIM and MSSSIM also compare the average luminance and contrast for the original frame and distorted frame. Moreover, Hore's study has revealed that PSNR is more sensitive to additive Gaussian noise than is SSIM, whereas the opposite is observed for jpeg compression. [16]

SSIM and MSSSIM cause people to prefer to use more sophisticated algorithms, such as visual information fidelity (VIF) and most apparent distortion (MAD). [25] However, these complex algorithms are gradually being phased out due to their slower processing speeds. More efficient and accurate algorithms are being developed by researchers. Lin *et al.* proposed feature similarity (FSIM), which utilized the phase congruency and the image gradient magnitude feature as the low-level features for video quality evaluation. [51] Xue *et al.* explored gradient magnitude similarity (GMS) between the reference and distorted images through a novel pooling strategy and the standard deviation. Their study achieved high efficiency and accuracy.

However, all the methods mentioned above only evaluate the quality of the 2D video. The methods for evaluating the quality of stereo video are mainly based on the methods for evaluating the quality of 2D video. Benoit *et al.* summarized the current studies for the quality assessment of stereoscopic images, where the main evaluation methods

are the traditional objective video quality assessments (e.g., SSIM) and subjective video quality assessments. [5]

2.3.2 Subjective video quality assessments

The design of an objective quality assessment needs to be validated through a subjective quality assessment. Subjective video quality is video quality as experienced by humans. It is related to how video is perceived by a viewer and designates their opinion on a particular video sequence; therefore, it is related to the field of quality of experience. The mean opinion score of the viewers is the opinion score of the evaluated video. By comparing the opinion scores of the original video and the distorted video, the video quality is assessed in terms of human perception.

The main difference between the various subjective video quality evaluation methods is the order in which videos are played. The basic subjective video quality evaluation method is double stimulus impairment scale (DSIS), where the videos are shown in pairs: the first video is the reference, and the expert is informed about it, and the second video is impaired. After playback of the videos, the expert is asked to provide his opinion using an impairment scale. The videos in double stimulus continuous quality scale (DSCQS) are also played in pairs, where one of the videos is the reference video, but the expert is not informed about it. [49]

The subjective assessment method for video quality evaluation is the main method adopted for 3D video. [5] The proposed approach is based on a random access process to play sequence files. Observers can start and stop the evaluation process as they wish and can follow their own paces in rating, modifying grades, and repeating playback when needed. Therefore, SAMVIQ can be defined as a multi-stimuli continuous quality scale method that uses explicit and hidden references. It provides an absolute measure of the subjective quality of distorted sequences that can be directly compared with the reference. Because the assessors can directly compare the impaired sequences among themselves and against the reference, they can grade them accordingly.

Based on the aforementioned evaluation approaches, we comprehensively evaluated the stereo video live streaming system.

2.4 Face detection

Since a state-of-the-art face detection algorithm has been applied in the telemedicine system, the related face detection algorithms are introduced in this section.

The cascade face detector proposed by Viola and Jones utilized Haar-like features and AdaBoost to train cascaded classifiers. [42] Since convolutional neural networks (CNN) have achieved remarkable progress in a variety of computer vision tasks, Girshick *et al.* introduced the R-CNN into face detection, which has achieved state-of-the-art results on VOC 2012. [12]

Li *et al.* then combined the cascaded structure and CNNs to improve the accuracy of detecting faces. [27] In their framework, 3 face detection CNNs and 3 bounding box calibration CNNs were cascaded at intervals. The face detection CNN was used to classify images as face or non-face, whereas the bounding box calibration CNN was applied to obtain high-quality localization. Since each network only has one task, this framework spends extra computation on different tasks and ignores the inherent correlation among similar tasks.

Based on cascaded CNN, Zhang *et al.* proposed a new framework called Multitask CNN (MTCNN) to integrate these two tasks using unified cascaded CNNs by multitask learning. [50] Multitask learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. This approach learns the multiple tasks in parallel by using a shared representation. The parameters learned for each task can help better learn other tasks. [7] In Zhang's work, MTCNN integrated the detection net and calibration net into one net to save computational resources and to achieve real-time performance. The parameters of the neural networks were also jointly trained by face detection and alignment to utilize the correlation between the face detection and alignment, which can improve the detection performance.

In this thesis, we developed the face detector based on the MTCNN proposed by Zhang *et al.* [50] and added the "feature pooling" technique to the MTCNN.

Chapter 3

Hardware Structure and Components of the System

This telemedicine system strongly relies on sufficient hardware devices that are able to capture, transmit and display the 3D live video. In this chapter, based on the motivation for the system, the structure and requirements of the hardware are presented. To meet these requirements, we developed a set of hardware components for the paramedic and emergency doctor. The hardware components and their detailed information are presented.

3.1 Hardware Structure

Based on the motivation for the telemedicine system, a simple workflow of the system is that the paramedic establishes a visitable 3D video live stream during the emergency process, while the emergency doctor accesses the video stream and establishes stereo video communication with the paramedics. The general hardware structure is designed based on this workflow and is shown in Figure 3.1, where the red arrows represent the data flow in the system, and the ellipses indicate that this system allows multiple paramedics and doctors to create multiple live streams and establish n-to-n video communication.

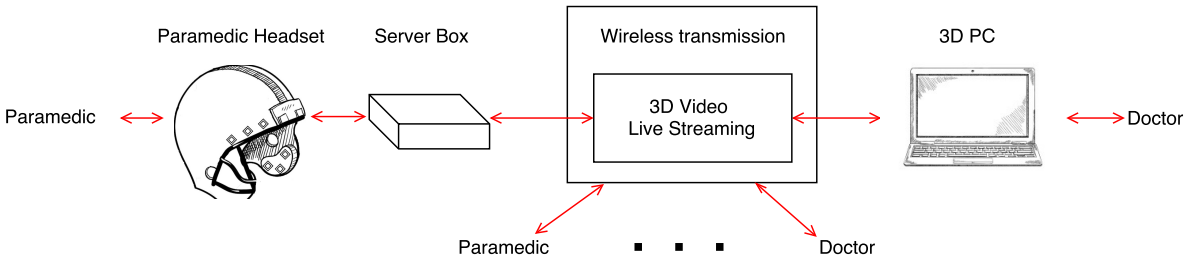


Figure 3.1: The general hardware structure of the system.

In this hardware structure, the hardware components are divided into three main parts: paramedic headset, server box and 3D PC. The paramedic headset and server box are carried by the paramedic, while the 3D PC is located in the remote hospital and used by the emergency doctor. Through these three hardware components, this structure promises the capability to capture, transmit and display 3D live video for stereo video communication between the paramedic and emergency doctor.

Specifically, for the paramedic's devices, the 3D paramedic headset is used to capture 3D video using a binocular camera. The binocular camera is covered by a metal shell and mounted on the 3D paramedic helmet by a support. The 3D paramedic headset is connected to a server box through a cable. The server is used to capture video from the headset, to transmit it as a live stream through Wi-Fi, and to provides power for the electrical devices on the paramedic headset. The server is also cover by a shell. The hardware structure of the paramedic's devices is shown in Figure 3.2, where the red arrows represent the flow of data and energy.

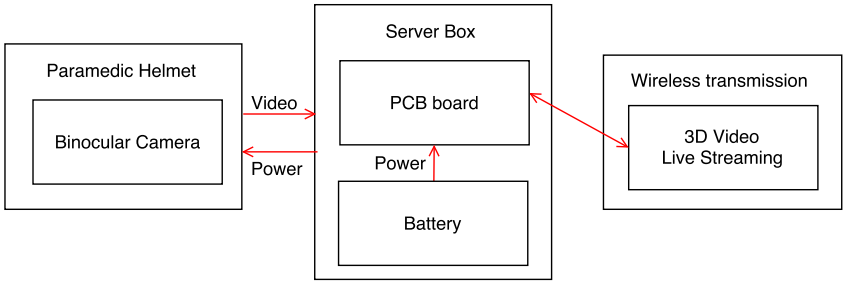


Figure 3.2: The hardware structure of the paramedic's devices.

Meanwhile, the doctor in the hospital is able to visit the live stream based on its published address through a 3D PC. The 3D PC in the hospital allows the doctor to watch the stereo video from 3D Vision Kit, which is rendered through the GPU. The hardware structure of the emergency doctor's device is shown in Figure 3.3, where the red arrows represent the flow of information.

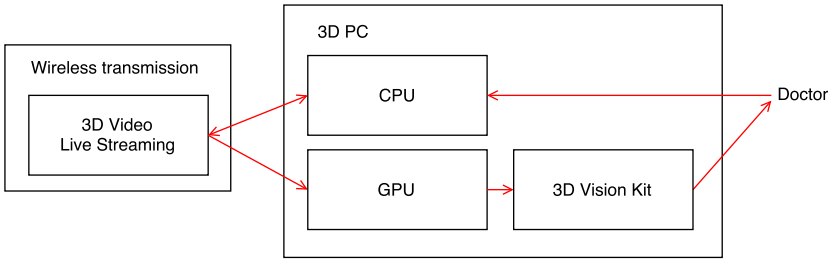


Figure 3.3: The hardware structure of the emergency doctor's device.

When the doctor successfully connects with the paramedic through the 3D PC, the headset starts to capture and transmit data to the hospital, and the received data are displayed on the 3D PC until the connection is terminated.

3.2 Detailed Information of Hardware Components

In this section, the detailed information of hardware components is presented for each component separately. Based on the hardware structure mentioned above, all of the hardware components and their features in the system are listed in the Table 3.1, where the first column is the three main parts of the system, the second column is the hardware components in the system, and the third column lists their corresponding features.

The detailed information and features of each component in this table are introduced separately in the following three subsections.

Table 3.1: The Hardware List of the Entire System and Its Related Features

Paramedic Headset	Binocular Camera	3 MP Efficient Pixels
		Captured Video as 30 $\sim$ 60 fps
		Compressed video as H.264
		Transmitted video by 2 USB 2.0 Port
	Shell of Cameras	Box contained binocular camera
		Connected with support through converter
	Support	GoPro Helmet Front Mount
		Mounted the shells on the helmet
Server Box	PCB board	Feed Data by 4 USB 2.0 Port
		Push Data through 802.11n Wireless LAN
	Battery	3-4 Hours of Active Use
		Up to 2 Weeks of Standby Time
		Passively cooled (no fans)
	Shell	Box contained the PCB board and battery
3D PC	3D Monitor	120 fps & 1920*1080 Monitor
	Host Computer	With NVIDIA GPU
	Infrared Emitter	Connect with PC by USB port
	Active Shutter	3D Vision 2 Wireless Glasses

3.2.1 Paramedic Headset

The paramedic headset along with the hardware components are shown in Figure 3.4, where each number label indexes a hardware component. There are three hardware components that are contained on the paramedic headset: the binocular camera (component 1), the shell of the camera (component 2) and the support (component 3).

The binocular camera (component 1) is the sensor for capturing the stereo video, and

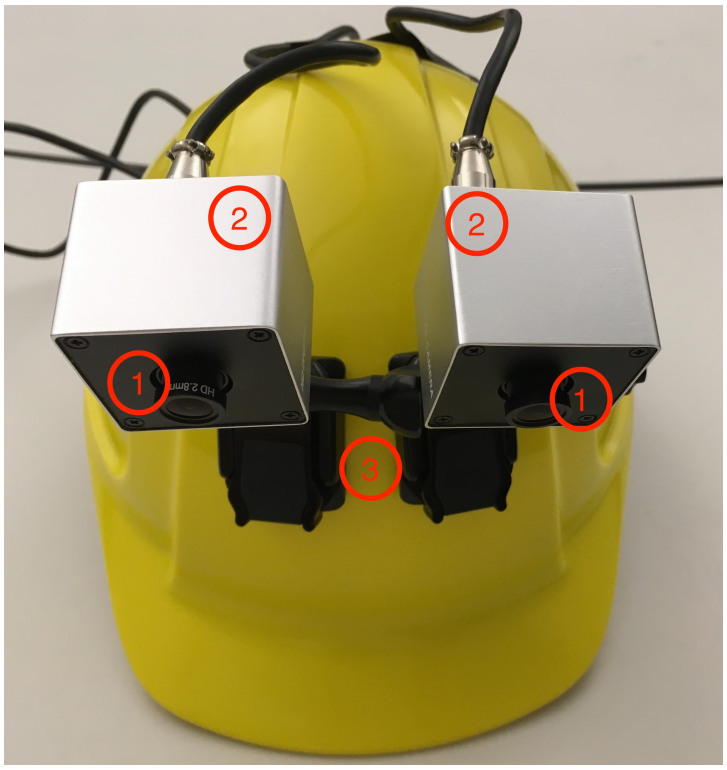


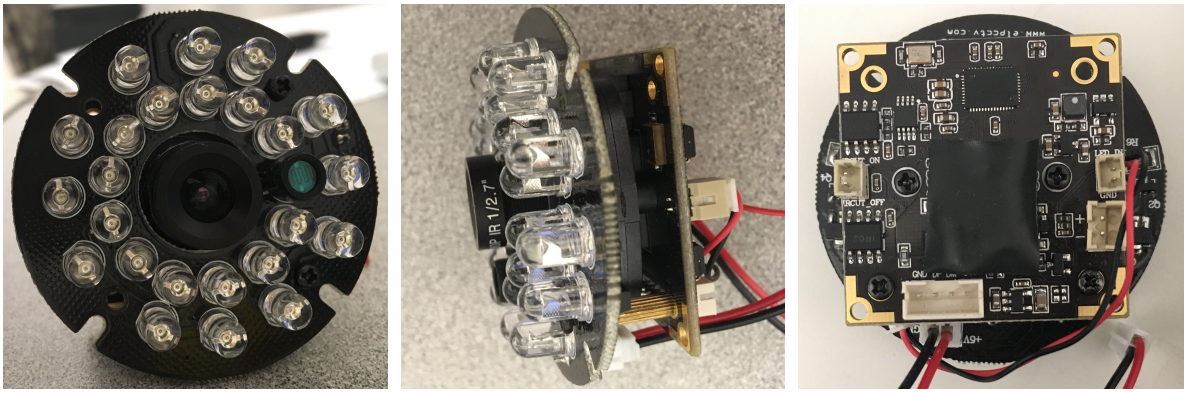
Figure 3.4: The paramedic headset with hardware components.

it is composed of two camera units and is protected by the shell (component 2). The shell also provides the cable and interface to connect the camera units to the server box and attach to the support (component 3). The support mounts the two camera units at the same level and allows them to group as a binocular camera. The details of the three components are presented separately in the following subsections.

3.2.1.1 Binocular Camera

The purpose of the binocular camera on the paramedic headset is to collect the stereo video and audio data for the server. The binocular camera is composed of two camera units. Each camera unit contains two sensor units, which are the camera and microphone. The three views of a single camera unit of the binocular camera are shown in Figure 3.5, where Figure 3.5a, Figure 3.5b and Figure 3.5c present the front, side and back views of the single camera unit, respectively.

The binocular camera consists of the two cameras fixed on the same surface (Figure



(a) (b) (c)

Figure 3.5: The three views of a single camera unit: (a) front view, (b) side view, and (c) back view.

3.5(a)). The video resolution of each camera is 1920 * 1080 pixels, and its frame rate can reach 60 frames per second. The high-resolution cameras ensure that the video can easily generate the depth map, while the high-frequency video is greatly helpful for the fluent 3D display on the 3D PC. The high resolution can also help the doctor obtain a clear understanding of the situation and provide a more accurate treatment. Moreover, the high resolution is useful for reducing the sense of vertigo and improving the viewing experience. The lens of the camera has a focal length of 2.8 mm to achieve a wider field of view. The wider field of view is closer to human perception.

The cameras are individually located on two PCB boards (Figure 3.5(b)). The PCB boards are not only used to provide power for the sensor and to capture the digital data from the sensor but also work for the video compression. The video compression process is shown in Figure 3.6. The boxes represent the components on the PCB board, while the text on the red arrows show the data transferred between the components.

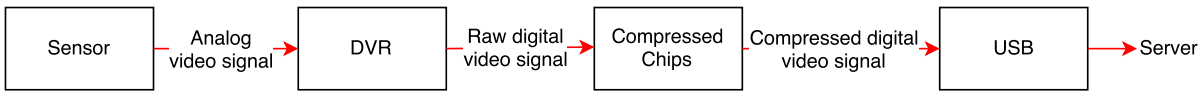


Figure 3.6: The flow chart of the video compression process in the sensor's PCB board.

As shown in Figure 3.6, when the sensor is powered, the analog video signal is inputted to the DVR card from the sensor. The DVR card converts the analog signal into a digital

signal and stores it in temporary storage. The digital video is compressed to different sizes by the video compression chips within the integrated circuit. The compressed data are finally sent to the server.

The compression procedure on the PCB board can compressed the raw data to the compressed data with h.264 format. The compression rate between the raw data and the compressed data is 20:1, which can greatly reduce the bandwidth. Besides, since PCB board uses the hardware compressor to compress the video, its speed is faster than the regular compression programs in the server box. [23] Consequently, the video data are compressed by the compression chips on the PCB board to reduce the potential delay.

There are four pin interfaces located on the back of the PCB board (Figure 3.5(c)). The 4-pin interface located on the bottom of the PCB board provides the power for the PCB board and transmits the video from the PCB board to the server. The other three pin interfaces can supply power for the auxiliary components, which operate at 5 V, 200 mA. The auxiliary components could be IRCUT or LED (Figure 3.5(a)), which can improve the quality of video in low light environments. These pin interfaces provide the possibility to extend the functionality of the sensor.

Moreover, the microphone, which can capture audio signals, is also located on the PCB board. The sound at the first aid scene can be captured by the microphone. The microphone also captures the voice of the first person to communicate with the remote doctor. In addition, the PCB board simultaneously captures the video and audio signals.

3.2.1.2 Shell

The shell of the camera on the paramedic headset is used to protect the camera units and to provide the interfaces for the support and server box. The side view of the shell is shown in Figure 3.7, where each number label indexes a part of the shell. Part 1 is the shell, which is used to protect the camera units; parts 2 and 3 are the connection interfaces for the support; and part 4 is the cable that is used to connect the 4-pin interface from the camera unit to the USB 2.0 port on the server box.

The camera unit is protected by the 41 $\times$ 41 $\times$ 41 mm metal box (part 1). The camera unit is fixed to the shell through four peripheral holes, which are shown in Figure

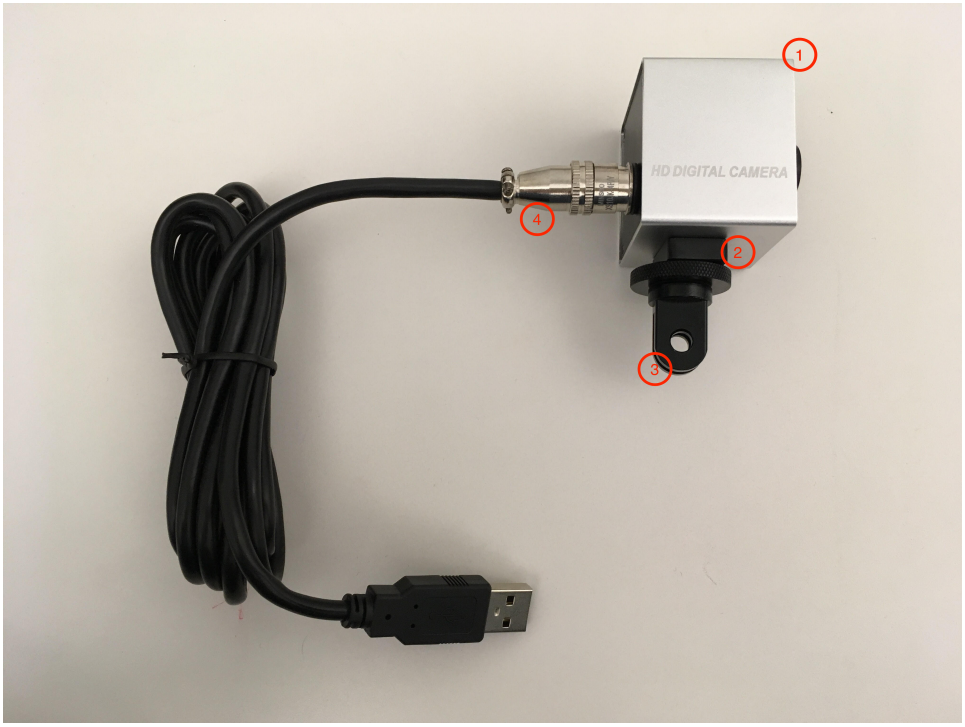


Figure 3.7: The side view of shell of the camera. (1) The metal shell. (2) The 6 mm screw thread. (3) The converter from the 6 mm screw thread to two 6 mm holes. (4) The cable that connects the binocular camera and the server box.

3.5c. There is a hole on the front surface of the shell for the camera lens, and its radius is 7 mm. On the bottom of the shell, there is a connection device that is used to fix the camera to the support. One end of the device is fixed to the shell with a 6 mm screw thread (part 2), whereas the other end is mounted on the support through two 6 mm holes (part 3). The connection device is made from metal to ensure the stability of the binocular camera. The camera unit is connected to the server box with the cable at the back surface of the shell (part 4). One end of the cable connects to the camera unit through a 4-pin interface, while the other end connects to the server box through a USB 2.0 port. The total length of the cable is 2 m. With this cable, the binocular camera is able to transmit the stereo video to the server box.

3.2.1.3 Support

The support of the camera is used to mount the camera on the paramedic headset, which utilizes a GoPro helmet mount. The front view of the support is shown in Figure 3.8, where each number label indexes a part of the shell.

Figure 3.8: The front view of the support of the camera. (1) Long thumb screw. (2) Vertical mounting buckle. (3) Curved adhesive mount.

The camera unit is fixed on the support with the long thumb screw (part 1) through the two holes on its shell and the three holes of the vertical mounting buckle (part 2). The vertical mounting buckle can adjust the angle of the camera. There is a cured adhesive mount at the bottom of the vertical mounting buckle. The curved adhesive mount can attach the camera to the curved paramedic helmet. When the binocular camera needs to be mounted on the paramedic helmet, two vertical mounting buckles should first be mounted on the helmet at the same level. Then, the camera units are inserted into the vertical mounting buckle, and its angle is adjusted. When the camera units have the same angle, it is fixed using the long thumb screw. Since the material of the support is metal, it is stable for recording 3D video without shifting and shaking.

3.2.1.4 Calibration the Binocular Camera

Since the camera units are mounted on the helmet separately, in order to make transferred video with stereo effect, the camera units need to calibrate properly as binocular camera. The calibration steps of the calibration is shown in following:

- Draw the center line of the helmet and its perpendicular (Line 1) on the helmet;
- Draw two parallel lines which are symmetrical to the center line, while the distance between two parallel lines (Line 2 and 3) is 1 cm;
- Fix the curved adhesive mount to the helmet and let their upper edge close to Line 1 and side edge close to the Line 2 and 3;
- Mount the vertical mounting buckle to the curved adhesive mount;
- Fix the interface converter to the shell of the camera units;
- Insert the converter to the vertical mounting buckle and fix them by the long thumb screw;
- Fine tune the angle of each interface component to make sure the camera units are symmetrical to the center line visually;

- Visualize the two camera and find three matching points in the center of the two captured image (avoiding camera distortion);
- Fine tune the angle of the interface components until the match points only have the horizontal translation in the two captured image.

Based on the above steps, two camera units can be composed as a binocular camera with suitable stereo effect.

3.2.2 Server Box

The server box in this system is used to capture the data from sensors and transmit the data to the 3D PC in the hospital through wireless communication. The hardware of the server box is shown in Figure 3.9, where each number label indexes a hardware component.

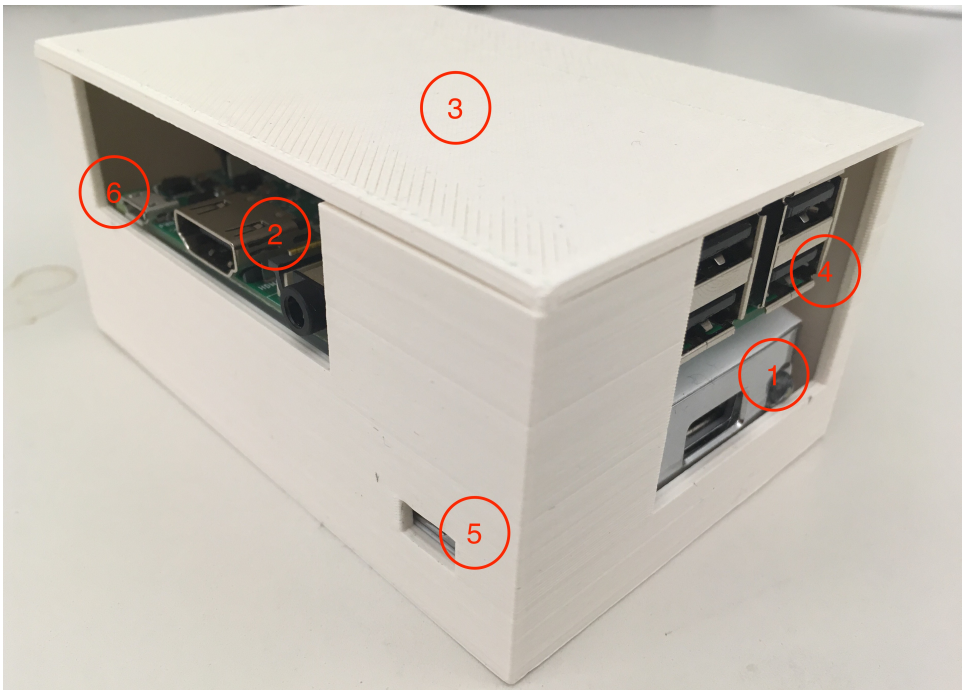


Figure 3.9: The server box with the hardware components.

The server box contains a PCB board (component 1), a battery (component 2) and a shell (component 3). The shell of the server box contains a hole for the USB port

(component 4) to connect the binocular camera and the micro USB ports (components 5 and 6) to provide power for the PCB board and battery. The details of the three components are presented separately in the following subsections.

3.2.2.1 PCB board

The PCB board is responsible for the live streaming server. When the PCB board is powered, a live streaming server (described in Section 4.1) is executed. This server transmits the stereo video captured by the binocular camera to the hospital when the 3D PC establishes the connection with the server. Since the video is compressed by the camera unit, the server does not have a complex computing process, which reduces the requirements for the performance of the processor. Compared with the powerful computing capability, light, small and low power consumption are more necessary for the server. Based on these requirements, a Raspberry PI 3 is adopted as the PCB board of the server. The top view of the PCB board is shown in Figure 3.10, where each number label indexes a hardware component.

The processor (component 1) of the board is a 1.2 GHz 64-bit quad-core ARMv8 CPU with 1 GB of RAM. The 802.11n wireless LAN (component 2) is integrated in the development board and provides the basic wireless communication ability of the server. The USB ports (component 5) of the server provide the power for the cameras and allow the sensor units to transmit data to the server. An extra wireless connection module can also be embedded on the server through the USB port. The server also contains a 40-pin extended GPIO (component 4), which allows the system to add extra auxiliary elements, such as indicator lights.

A micro SD card (component 2) is used to stored the program of the live streaming server which is described in Section 4.1. The micro SD card is inserted in the micro SD card slot on the development board. The development board has a full HDMI port (component 6) and a USB port (component 5), which can be connected to a monitor and keyboard. These components allow the status of the live streaming server to be checked and debugged locally. In addition, it also allows a remote PC to control the server through ssh. Moreover, the PCB board is powered by the micro USB port (component

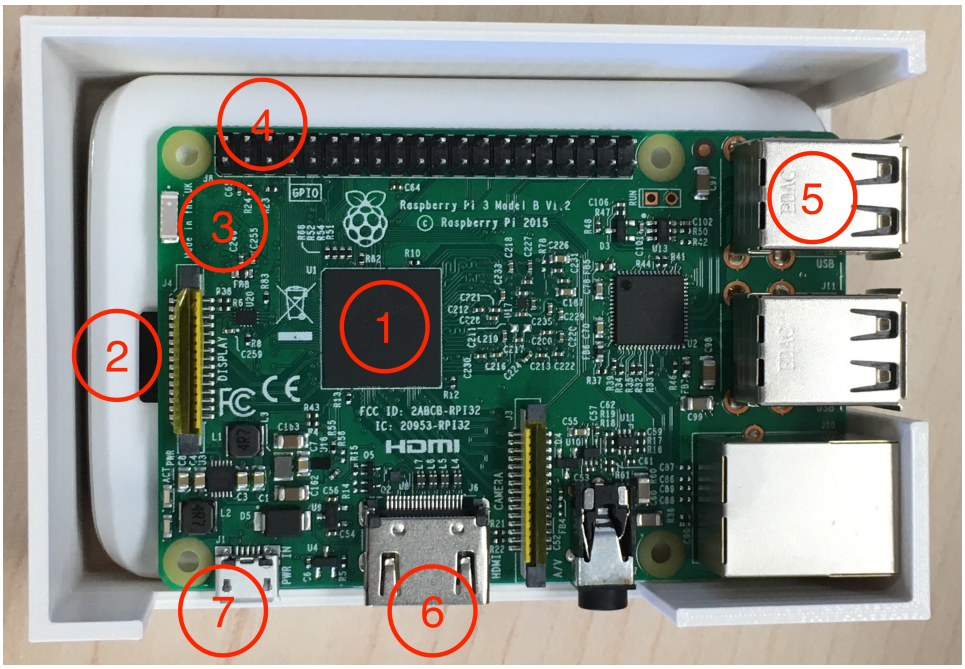


Figure 3.10: The top view of the PCB board with the hardware components. (1) Processor. (2) Micro SD card. (3) Wi-Fi. (4) GPIO. (5) USB ports. (6) HDMI port. (7) Micro USB power input port.

7), which requires a 5 V, 2 A DC input.

3.2.2.2 Battery

A rechargeable battery provides qualified power for the PCB board. The output voltage of the battery is 5 V, and its output current is 2 A. This battery can support the server for 3 to 4 hours in working mode or for 2 weeks in standby mode if it is fully charged.

A 6,600 mAh lithium ion battery is located inside. The charging circuit of the battery limits the charge voltage to 5 V. When the remaining charge is low, the battery output voltage would be unstable, which would cause the server to fail to work properly.

Although passive cooling is adopted by the server, the temperatures of the processor and battery are acceptable when the server is operating. Since there are no fans to facilitate cooling, the server is quiet during operation.

3.2.2.3 Shell

The shell is used to protect the PCB board and battery. The shell is produced by 3D printing with polylactic acid filament, and its size is 102 * 68 * 48 mm. The thickness of the shell is 2 mm. There are several interfaces on the shell, such as the interface for the PCB board and the battery switch.

3.3 3D Personal Computer

The hardware components (a total of 4 components) of the 3D PC are shown in Figure 3.11, where each number label indexes a hardware component. The 3D PC has 4 hardware components: NVIDIA 3D Vision Kit (components 1 and 2), 3D Vision Ready Display (component 4), NVIDIA GPU and a PC with Microsoft Windows 7 (component 3).

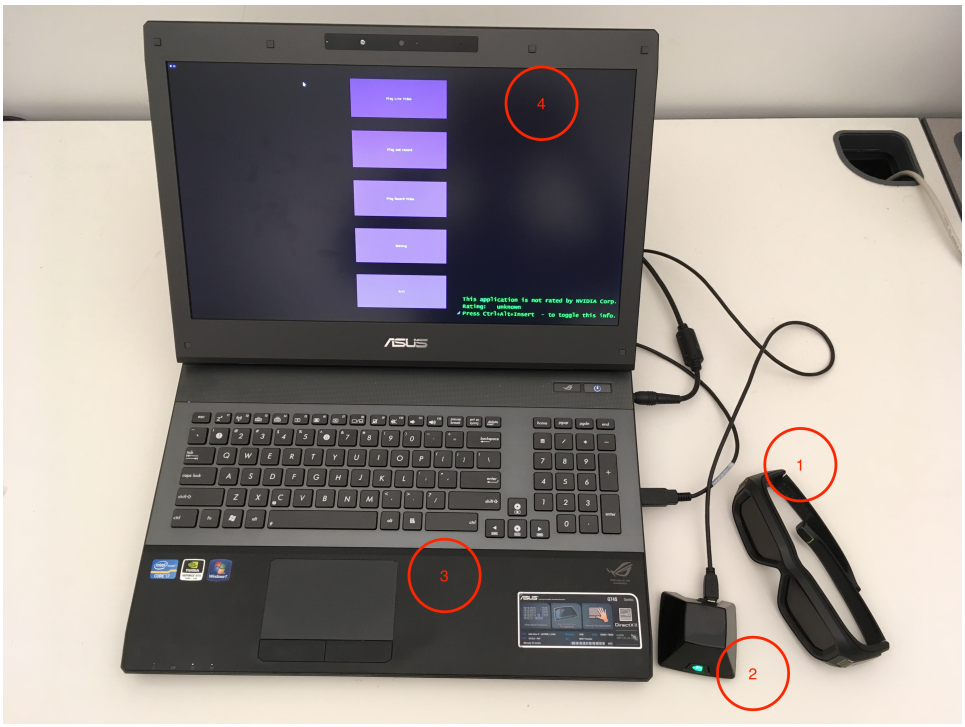


Figure 3.11: The hardware components of the 3D PC. (1) NVIDIA 3D active shutter glass, (2) NVIDIA IR emitter, (3) PC with NVIDIA GPU and Microsoft Windows 7, and (4) 3D Vision Ready Display.

The functions of these hardware components and their relationships are shown in Figure 3.12. The large boxes are the hardware components, and the small boxes represent the objects contained in each component. The red arrows and their corresponding text show the data flow between each component.

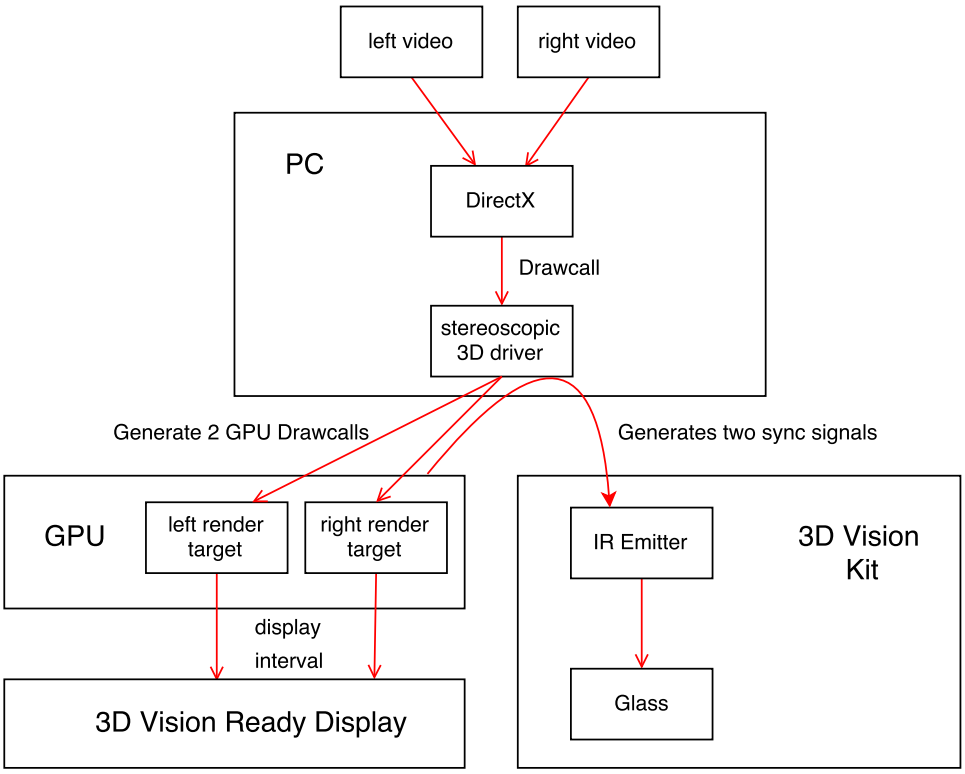


Figure 3.12: The functions of the hardware components of the 3D PC and their relationships.

In Figure 3.12, the left and right video streams from the server are captured by the 3D PC. The 3D PC sends the video to the graphic engine - DirectX. DirectX calls the stereoscopic 3D driver to render the video. This driver generates 2 commands for the NVIDIA GPU and lets it draw 2 render targets separately. The GPU then sends the render target to the screen and sends two sync signals to the NVIDIA 3D Vision Kit. The IR emitter (component 2) controls the shutter switch of the active shutter glass (component 1). Finally, the 3D vision monitor displays the left render target and right render target interval, while the 3D wireless glass switches the shutter correspondingly at the same time.

Due to their physical and functional relevance, in the following subsection, the hardware components of the 3D PC are introduced separately.

3.3.1 PC and GPU

The NVIDIA GPU is located inside the PC. It is used to accelerate the creation of images in a frame buffer intended for output to the 3D display device. Its rendering approach is controlled by the stereoscopic driver on the PC, and the driver is called through DirectX.

When the PC receives the stereo live video data, it reads the data to a frame buffer through DirectX. DirectX calls the stereoscopic driver and sends the 3D video data to it. When the driver receives the stereo tag, it takes the 3D video data and renders each scene twice – one for the left eye and one for the right eye. Meanwhile, the GPU also sends the two sync signals to control the shutter of the 3D glasses.

The stereoscopic 3D driver is combined with the NVIDIA GPU driver. When the GPU driver is installed, stereoscopic 3D can be enabled in the NVIDIA GPU control panel. To successfully initialize the stereoscopic 3D, there are several requirements that need to be met. The stereoscopic 3D driver is called by DirectX. The version of DirectX embedded in the system needs to be higher than DirectX 9.0c. DirectX 12 does not support the stereoscopic 3D driver perfectly, which means that the 3D vision would encounter driver problems in Windows 10. The PC needs at least two-core processors and 1 GB of memory. In addition, when the stereoscopic 3D is initializing, the IR emitter of the 3D Vision Kit needs to be connected to the PC and keep opening. When the stereoscopic 3D driver is initialized and its correlated test is successfully running with the stereo effect, the 3D vision is able to run properly on this computer.

3.3.2 3D Vision Kit and Display Monitor

The 3D Vision Kit and 3D Vision Ready Display are responsible for visualizing a 3D effect for the human eyes.

The 3D Vision Ready Display is the display machine with the NVIDIA 3D Vision Ready logo. It could be a high-frequency digital monitor, HDTV or projector. [33] The

3D Vision Ready monitor displays the rendered target from the GPU. The 3D monitor shows the stereo video interval - the left eye view for even frames (0, 2, 4, and so on) and the right eye view for odd frames (1, 3, 5, and so on). For the left eye view, only the left eye can see the image, and vice versa. In the 3D Vision Kit, this is achieved by the active shutter glasses.

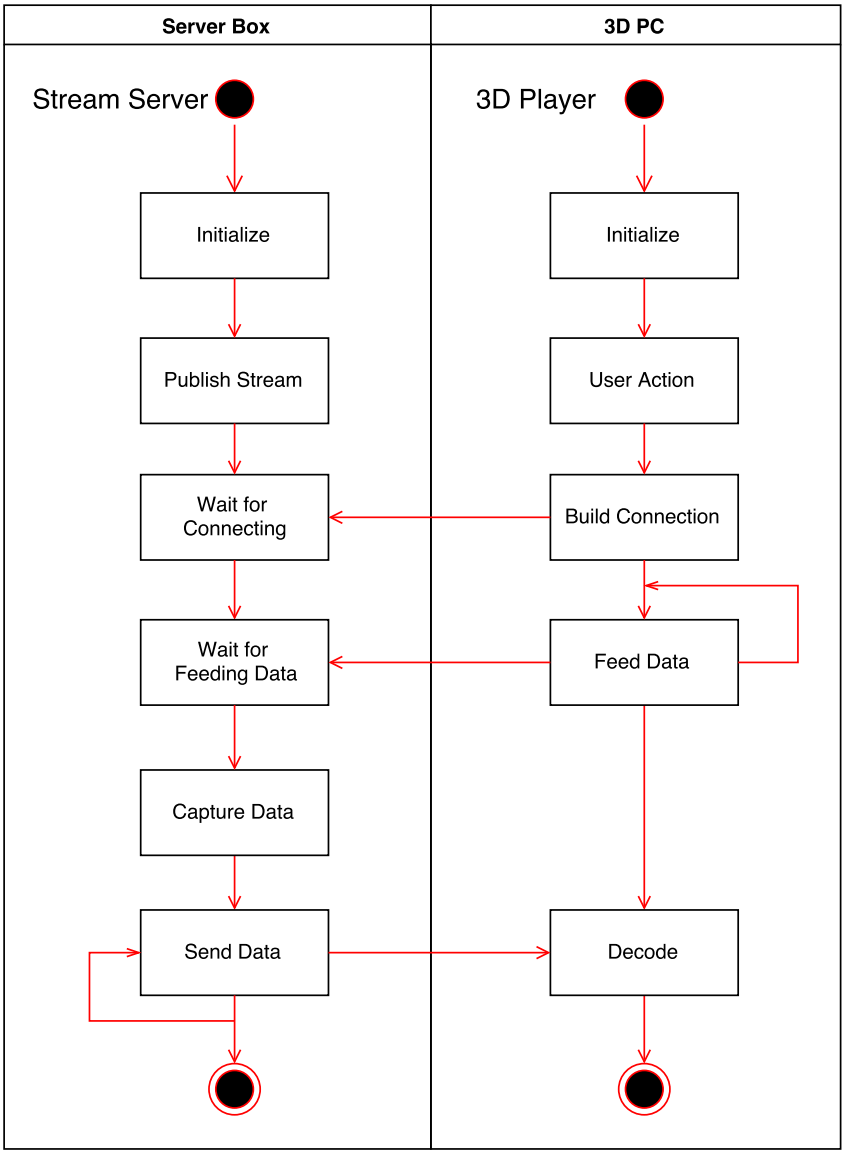
The active shutter glasses blacken out the right lens when the left eye view is shown on the display and blacken out the left lens when the right eye view is shown on the display. The synchronization between the monitor and the glasses is controlled by the GPU and stereoscopic 3D driver and transmitted by the IR emitter. The IR emitter of the 3D Vision Kit is connected to the PC through USB 2.0 mini-B. The IR emitter transmits an IR signal to the wireless glasses between 1.5 and 15 feet. The active shutter glasses receive the signals and blacken out the relative lens.

The refresh rate of the display is effectively cut in half for each eye (e.g., a display running at 120 fps is 60 fps per eye). When the refresh rate of the 3D monitor and glasses is considerably higher than the human perception ability (30 fps per eye), the human feels that the vision signals are captured synchronously by the left and right eyes. The resulting image for the end user is a combined image that appears to have depth in front of and behind the stereoscopic 3D display.

Chapter 4

Functional Mechanism

There are two subsystems that are responsible for the mechanism running on the server box and 3D PC, which are the stream server and the 3D player. These two subsystems communicate with each other by passing the data over the Internet. The server box captures the compressed stereo video from the paramedic headset and sends the video data to the 3D PC, while the 3D PC feeds the data and displays it on the 3D monitor. The flow chart of the data communication between these two subsystems is shown in Figure 4.1. The two outside boxes represent the hardware platform of each subsystem. The black points represent the start and end of each subsystem. The small boxes are the general functional processes of the subsystem, and the red arrows present the flow of the program.



a

Figure 4.1: The flow chart of the data communication showing the data communication between two individual subsystems when they work independently.

In Figure 4.1, the server box begins by the stream server running on the PCB board, while the 3D player is executed and initialized on the 3D PC. The stream server checks the input parameters and the sensor devices. If the sensors are successfully connected, then the stream server publishes the live stream to the Internet through wireless modules such as Wi-Fi. When the emergency doctor inputs the stream address into the 3D player, these two subsystems are connected. After the 3D player asks for the data from the server, the server starts to capture the compression data from the binocular camera and send it to the 3D player. The 3D player then decodes the compressed data and displays it on the 3D monitor.

4.1 Stream Server

The stream server is used to send the video data from the paramedic headset to the hospital. It is responsible for publishing the 3D video live stream and transmitting the video data to the 3D player when the 3D player connects to the live stream. This subsystem works on the PCB board contained in the server box.

Thus, the main task of the live stream server is to control the process of establishing the connection and the status of playing the live stream. For this purpose, a real-time streaming protocol (RTSP) is adopted as the live streaming protocol to establish and control connection sessions between the stream server and 3D player. The concurrent session of RTSP live streaming is tracked and controlled by the control sequences and the identifier. The workflow of the stream server and the process of data transfer between the stream server and 3D player is shown in Figure 4.2, where the black circles are the start and end points of each part, the white boxes are the progress of each part, the red arrows with the solid line are the transmission of the video data and predefined commands of RTSP live streaming between subsystems, and the red arrows with the dotted line are the sleeping progress in the subsystem.

In Figure 4.2, when the stream server is launched, the 3D player sends the "DESCRIBE" request to the server. The "DESCRIBE" request is a predefined command of RTSP live streaming. This request includes an RTSP URL and the type of reply

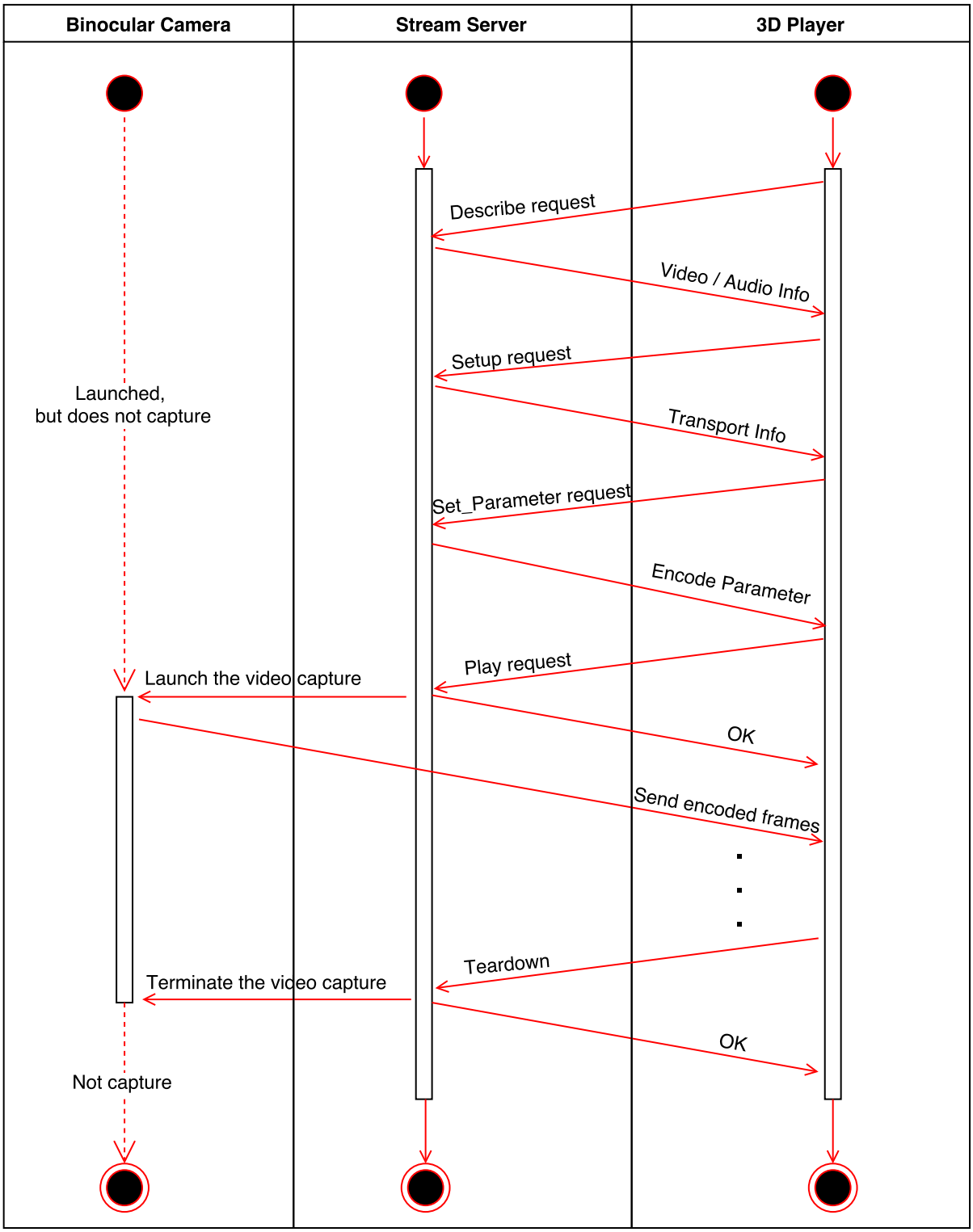


Figure 4.2: The flow chart of data transfer functions with RTSP live streaming between the stream server and the 3D player.

data that can be handled. Then, the stream server replies to the 3D player with the presentation description. The presentation description lists the media streams controlled by the aggregate URL. Since there is one media stream each for video, the stream server sends information of 2 streams back to the 3D player. This response must contain all media initialization information. For example, the 3D player sends a "DESCRIBE" request such as "rtsp://example.com/video RTSP/1.0", and the stream server replies back to the 3D player with the stream information such as streamid (0/1) and mime-type (video/MP4V-ES). The "rtsp://example.com/video" is the aggregate URL of the concurrent sessions.

The 3D player then uses the "SETUP" request to setup each stream based on the video and audio information. The "SETUP" request is a predefined command of RTSP live streaming. This request specifies the parameter for transporting live streams. The 3D player only needs to define the chosen parameters, while the other default parts can be missed in the request. The stream server replies typically confirm the chosen parameters and fill in the missing parts as their default values. Each live stream must be configured using "SETUP" before sending the aggregate play. For example, the "SETUP" request from the 3D player only contains the URL (rtsp://example.com/video/streamid=0) and client port (13000). The fulfilled transport information is sent back to the 3D player, including the missing parts, such as setting the server port as the default (14000) to the transport information.

The "SET_PARAMETER" request is then sent to the stream server. This request is a predefined command of RTSP live streaming, and it is used to set the video capture parameters of concurrent live streaming in the stream server. The stream server checks the parameters and returns validation of the parameters. The parameters of video capture include the resolution and frame rate of the captured video. This request also defines the encoding format of the captured video data.

When all the prepared parameters are setup ideally, the "PLAY" request should be sent to the stream server to play all live streams. The "PLAY" request is a predefined command of RTSP live streaming. This request allows the stream server to transmit the video data until the live streaming video is paused or ended. When the stream

server receives the command, it starts to capture the video from the binocular camera as using the defined parameters. Note that the binocular camera is launched when the stream server is executed, while it continues sleeping until the server receives the "PLAY" request. The stream server replies to the 3D player with the confirmation signal. The encoded video data from the camera are then transmitted to the 3D player by the stream server. The 3D player decodes and displays the video on the 3D laptop.

When the emergency doctor stops the video, the 3D player sends the "TEARDOWN" request to terminate the session. The "TEARDOWN" request is a predefined command of RTSP live streaming. This request stops all the live streams and frees the related data of the concurrent session on the stream server. The stream server also terminates the video capture process and replies to the 3D player with the status. Unless all the transport parameters are defined by the session description, a "SETUP" request must be issued before the session can be played again. Then, the stream server waits for the next connection from the 3D player.

4.2 3D player

The 3D player is a subsystem that runs in the 3D PC. The emergency doctor in the hospital uses the 3D player to play the stereo video by accessing the video address. The 3D player feeds the stereo video from the video address, decodes the compressed video data, analyzes the video and displays it on the 3D monitor. Based on its functions, the 3D player contains 5 main functional packages: controller, decoder, analyst, viewer and UI. The relation and modules of these functional packages are shown in Figure 4.3, where the colorful folders are the packages of the player, the pink bookmarks are the modules in each package, and the arrows indicate the relationship and data flow between each package.

As shown in Figure 4.3, the 3D player uses the controller to mount or free the decoder, viewer and user interface (UI) packages. When the package is mounted, the modules in each package are executed based on the status in the controller package. These modules also feed their status back to the controller and update the status of the controller.

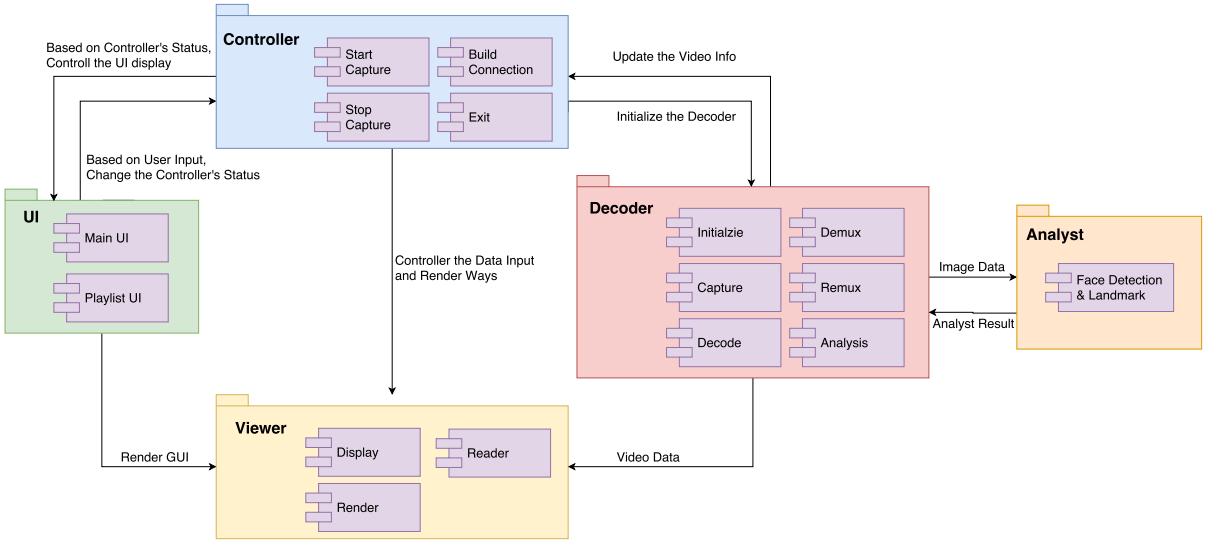


Figure 4.3: The package diagram of the 3D player.

The viewer package is the first package that the controller needs to mount. The viewer is used to render and display the data captured from the UI and decoder packages. The UI package is then mounted on the player by the controller. This package defines and displays the graphical user interface (GUI), and it allows the user to control the parameters in the controller to control the entire player. When the user inputs the video address through the GUI and starts to play the video, the decoder package is mounted by the controller. The decoder visits the video address and captures the compressed video data from the stream server. The video is then de-muxed and decoded as raw RGB data. The raw data are read and displayed by the viewer. The video can also be analyzed through the analyst package. This package is mounted to the stereo player by the decoder. The analyst package is not necessary to play the stereo video in this system; however, it can analyze the video to extend the intelligent functions of the player based on computer vision algorithms. This package receives the video data from the decoder package and returns the analyzed result to the view package.

The detailed mechanism of each package is introduced in the following subsections.

4.2.1 Controller

The controller package is the core mechanism of the 3D player. This package operates the 3D player through controlling the package mounting process of the other packages.

In Figure 4.3, the blue folder represents the controller package. Its modules, except for its constructor, are contained in the folder, which is "Start Capture", "Build Connection", "Stop Capture" and "Exit". The constructor of the controller is used to initialize the parameters of the controller. It also mounts the viewer package and "Main UI" module on the system. "Start Capture" mounts the "Playlist UI" module on the system. "Build Connection" updates the parameter from the "Playlist UI" and frees this module. The decoder package is also mounted by the "Build Connection" module based on the input parameters. The "Stop Capture" module frees the decoder package, and "Exit" frees the viewer package and "Main UI".

The detailed package mounting process of the controller is shown in Figure 4.4, where the four large boxes represent the controller, UI, decoder and viewer packages; the black points represent the start point of each package or module, while black points with the red circle represent the end point of each procedure; the white boxes are the external activities of this subsystem; and the pink bookmarks present the modules of each package shown in Figure 4.3.

Specifically, as shown in Figure 4.4, when the 3D player is executed, the controller is initialized and mounted on the system. The controller then mounts the viewer package and the "Main UI" module on the system. The viewer reads the background data and renders it on the screen. When the emergency doctor prefers to watch the stereo video, the controller calls the "Start Capture" module. The "Playlist UI" module is mounted by this module. When the emergency doctor inputs the video address and display mode, these data are sent to the controller and call the "Build Connection" module. The "Build Connection" module releases the "Playlist UI" module and mounts the decoder package on the system. When the decoder package is initialized, it connects the stream server and captures the data from it based on the stream address stored in the controller. The decoder then de-muxs the video package of the live stream and obtains the compressed

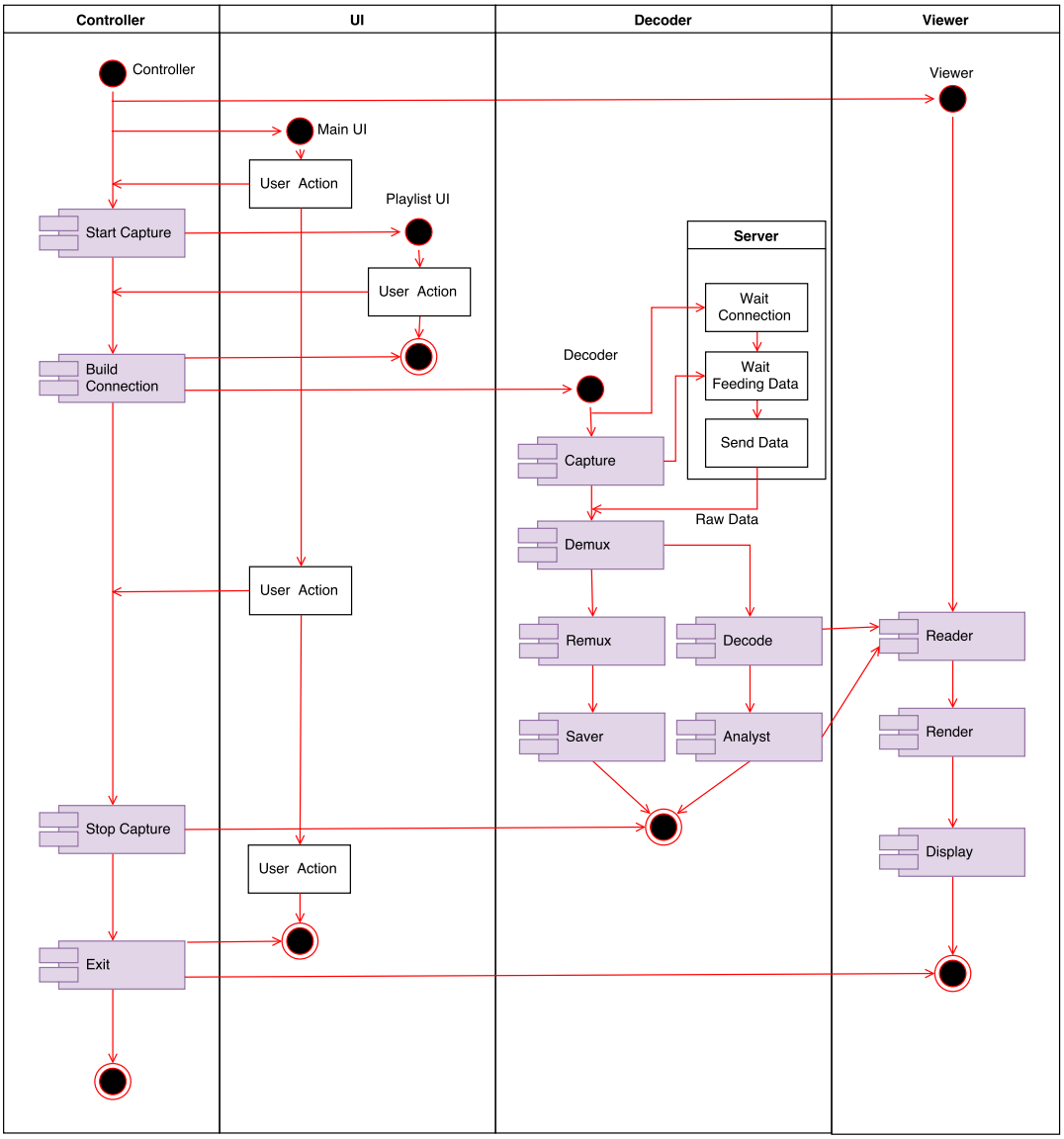


Figure 4.4: The flow chart shows the package mounting process of the controller.

data of the video. The compressed data can be decoded as RGB image data. The viewer reads the RGB data and renders it on the screen based on the render mode contained in the controller. The RGB data can also be sent to the "Analyst" module. The "Analyst" module is used to mount the analyst package. This package reads the data and sends the analyst result back, and the analyst result can be shown in the viewer. The compressed data in the "Demux" module can be re-muxed as files and saved on the PC by the "Remux" and "Saver" modules. When the emergency doctor stops playing the video, the "Stop Capture" module is called to free the decoder package. When the decoder package is completely free, the emergency doctor can play another video or simply exit the program. When the "Exit" module is called, it frees the "Main UI" module and the viewer package. Subsequently, the controller releases itself and exits the 3D player program.

This procedure shows the package mounting process of the controller. It also presents the procedure of the 3D player. In the next few subsections, the detailed implementation principles of the viewer, UI and decoder are introduced.

4.2.2 Viewer

The viewer package is responsible for rendering the stereo video into 3D mode and displaying it on the 3D monitor. Since the NVIDIA 3D Vision Kit is applied as the hardware of the 3D display in this system, it requires DirectX to render two times for each 3D scene - once for the left eye and once for the right eye.[1] This procedure is shown in Figure 4.5, where the pink bookmarks are the modules shown in Figure 4.3, the black arrows represent the data flow, and the boxes present the image data in different steps.

Specifically, DirectX is a 3D graphics engine for displaying 3D objects in a virtual environment. Therefore, the video data need to be stored as a surface object in DirectX. In Figure 4.5, the "Reader" module reads the RGB video data from the "Decoder" module into an off-screen surface. The off-screen surface not only contains the left and right images but also has a stereo tag to inform the driver to render this scene as a stereo scene. In this case, if the size of the left and right images is $w \times h$, then the size of the

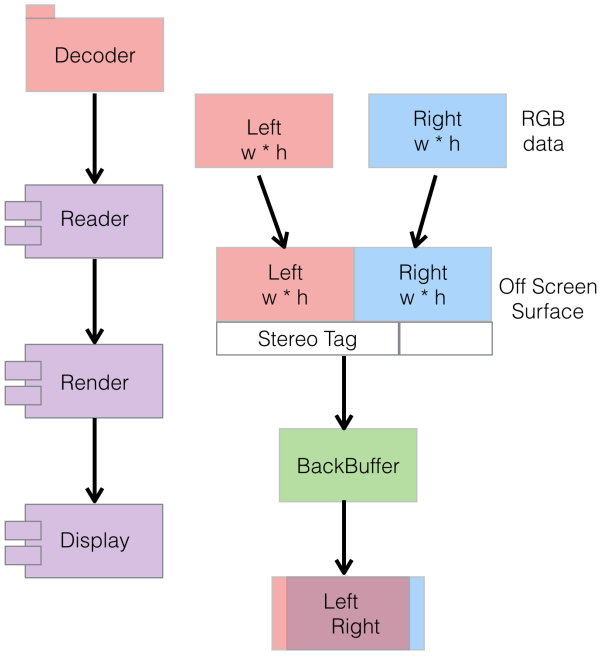


Figure 4.5: The procedure of displaying the stereo video on the NVIDIA 3D Vision platform.

off-screen surface should be $2w \times (h + 1)$. The stereo tag is stored in the last extra row. The "Render" module renders the video surface into the back buffer through StretchRect in DirectX, while the "Display" module presents the back buffer to the driver. Inside the driver, it checks the stereo tag and renders each 3D scene two times. Subsequently, the stereo frame is displayed on the 3D monitor.

4.2.3 UI

The UI package implements a GUI function of the 3D player, which allows emergency doctors to interact with the 3D player through graphical icons and visual indicators.

One critical problem in the NVIDIA 3D Vision Kit is that it requires the 3D player to run in full-screen mode to display the stereo video. If the 3D player runs as a windowed program, it only plays the regular side-by-side video without a 3D effect. Therefore, the 3D player needs the GUI parts embedded into DirectX. In this case, the immediate mode graphical user interface (ImGui [19]) is adopted in the 3D player as the UI package.

Compared with windowed UI methods such as MFC or Qt, ImGui is particularly

suited for embedding into 3D full-screen applications. Through outputting the vertex buffer, ImGui can be rendered and displayed as a 3D object in DirectX. ImGui visualizes a floating windowed UI contained in the whole video screen in the 3D player. The emergency doctors can hide the UI to make the screen clean and call it out when they need it. Since the floating windowed UI is transparent, it does not interrupt the video effect for human perception.

An example workflow of the UI package in the 3D player when the emergency doctor prefers to display a video is shown in Figure 4.6, where the red circles represent the label of the subfigures in Figure 4.6, and the red boxes and red arrows mean the relative actions and procedure of clicking the buttons, respectively.

As with the indications in Figure 4.4, when the "Main UI" module is mounted on the 3D player by the controller, the UI of the 3D player is as shown in subfigure 1. The UI has 3 buttons: "Home Page", "Start Capture" and "Exit". The "Home Page" button is able to let the controller reinitialize itself, while the "Exit" button tells the controller to close this program. The "Start Capture" button informs the controller to mount the "Playlist UI" module, which is shown in subfigure 2.

The "Playlist UI" module is used to initialize the parameters for playing the video. The 3D player is able to visit the live streaming video and the recorded files. The live streaming video can be visited using the "Network Address" button, and the recorded video is able to be read using the "Saved Files" button. The "Add" button is prepared for visiting the external live stream. When the "Add" button is clicked, the "Playlist" module allows the emergency doctor to add the stream address to the list (shown as subfigure 3).

When one video address is selected, it is shown at the top of the "Playlist" (shown as subfigure 4). Then, this address is able to be connected ("Connect" button), deleted from the list ("Delete" button) or does nothing ("Back" button). When this address is deleted or does nothing, the "Playlist" returns to the status shown in subfigure 2 and lets the user re-select the video address again. When the user decides to play the video at this address, the doctor needs to set the play mode, decode core, and video mode. The "Write File" clickbox in play mode is used to tell the controller to save the video

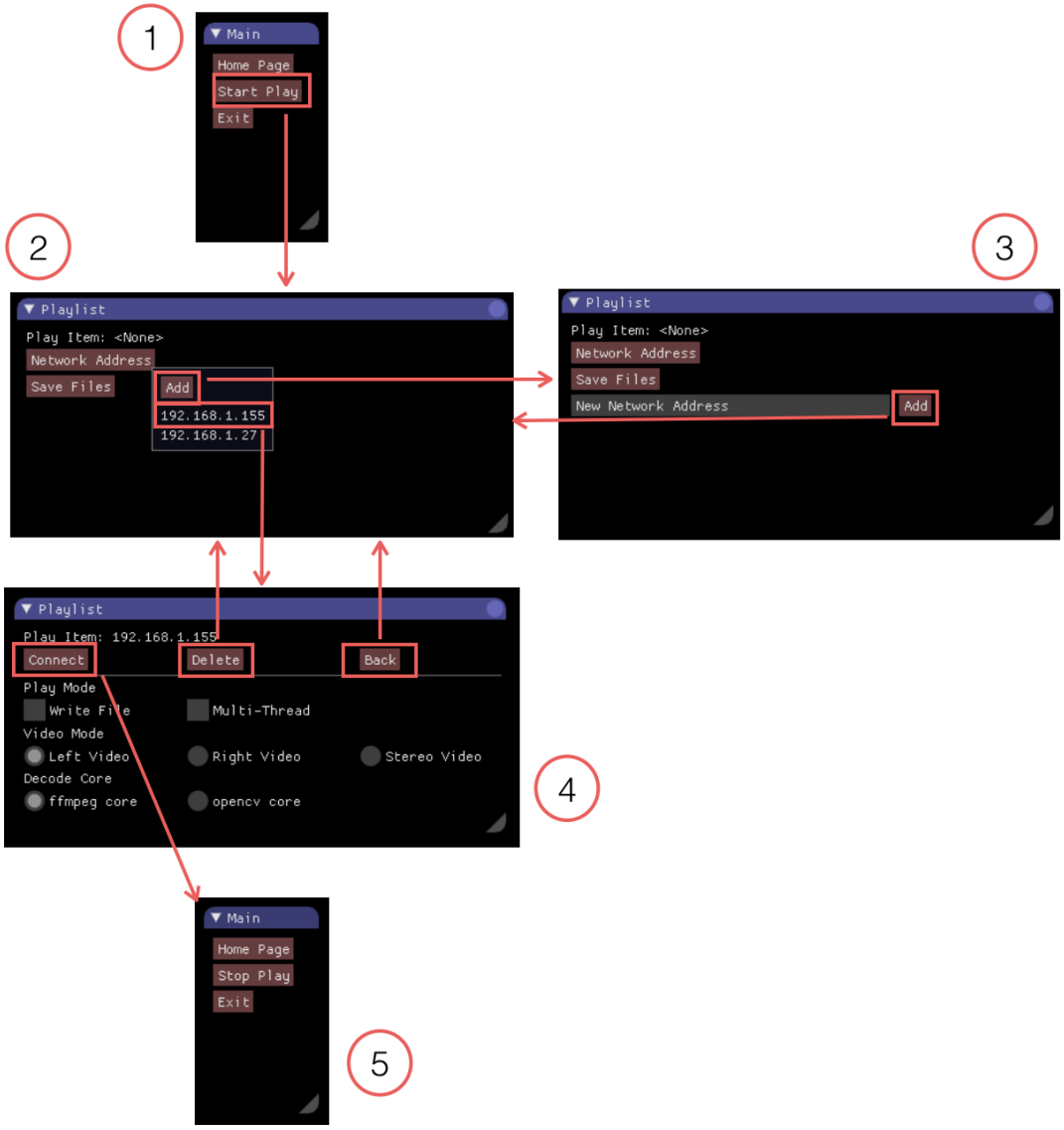


Figure 4.6: The procedure of controlling the GUI to display a video.

into the file. If the file is successfully saved, the file is shown in the "Save File" list. The "Multi-thread" clickbox lets the viewer package read each RGB image by multi-thread. The decode core determines the video decoder in the 3D player. There are two video decoders in the system, which are FFMpeg and OpenCV. FFmpeg core is faster, and OpenCV core allows the video to be analyzed. The video mode determines the capturing and rendering approaches in the 3D player. The "Left Video" and "Right Video" options let the 3D player capture the data from the mono camera and display it as regular 2D video, whereas the "Stereo Video" option lets the 3D player capture video from the binocular camera and display it as 3D video. The mono channel mode can reduce the burden of the communication channel for poor bandwidth conditions.

When the "Connect" button is clicked, the "Build Connection" module in the controller is called. The "Playlist UI" module is released by the controller, and the decoder package is mounted in the system. Only the "Main UI" module still exists on the video screen (shown as subfigure 5). The "Stop Capture" button replaces the "Start Capture" button in subfigure 1. The "Stop Capture" button is the trigger of the "Stop Capture" module. It releases the decoder package and makes the "Main UI" return to the initial status, which is shown in subfigure 1. In this case, the emergency doctor could play another video by repeating the above steps.

4.2.4 Decoder

The decoder package is applied to feed the video data from the stream server and decode compressed video into raw RGB data that can be read by the viewer.

The encoding and decoding processes in this system are shown in Figure 4.7, where the red boxes represent the data format in the different statuses, the blue box presents the objects to process the data, and the arrows show the flow of the data.

In Figure 4.7, the raw video data are first captured by the binocular camera. The sensor encodes the original analog signal to the YUV420P format, which is standard pixel format. YUV is a color space that is typically used as part of a color image pipeline with one luma (Y) and two chrominance (UV) components, and the chrominance components can reduce bandwidth for communication. YUV signals are typically created from an

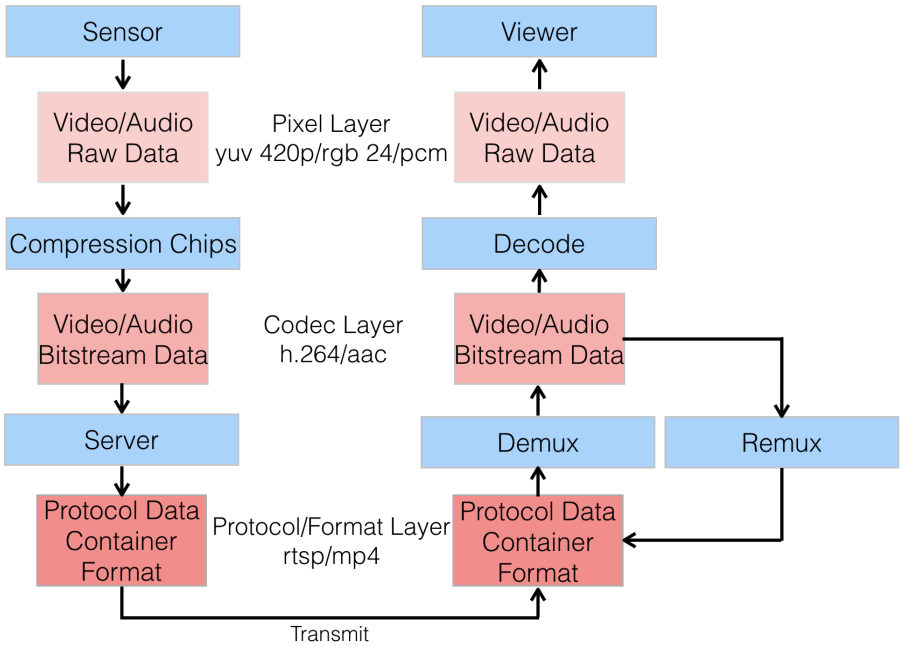


Figure 4.7: The encoding and decoding processes in this system.

RGB (red, green and blue) source. The conversion between YUV and RGB is shown in Equation 4.1.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.1}$$

The YUV signals are then compressed into the h.264 bitstream by the compression chips on the PCB board of the binocular camera. The h.264 conforms with the standard special encoding format for bitstream data. Since the RTSP is applied as the video live streaming protocol in this system, the stream server then adds the standard live streaming header and trailer (the protocol data) for the bitstream data. The protocol data of the RTSP live streaming added to the bitstream are shown in Figure 4.8, where the white small boxes represent the protocols in the live streaming, and the arrows show the data flow between different protocols.

In Figure 4.8, there are three protocols applied in the RTSP live streaming: RTP, RTCP, and RTSP. RTSP is used for establishing and controlling media sessions between end points. It always contains the predefined commands of RTSP live streaming. Based

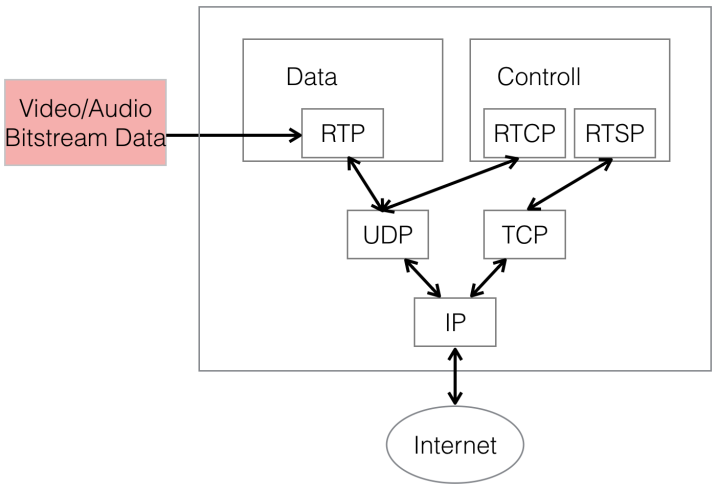


Figure 4.8: The protocol data of the RTSP live streaming added into the bitstream.

on the RTSP, the client can control the live streaming and perform actions such as play, pause, fast forward, record, and stop play. When the live streaming is established by the RTSP, the bitstream data start to be sent to the stream server through the real-time transport protocol (RTP). RTP is primarily used to carry real-time transmission of streaming media data, including audio and video data, as well as the time load and the serial number of the video stream. When the video stream is received by the 3D player, the RTP control protocol (RTCP) is used to monitor the quality of the video stream. RTCP brings the 3D player's feedback of packet counts, packet loss, packet delay variation, and round-trip delay time to the stream server. In this case, the server uses this information to show the quality of the service parameters. The data packaged by the RTP and RTCP are sent by user datagram protocol (UDP) to reduce the delay.

When the decoder in the 3D player receives the protocol data from the server, the "Demux" module in the decoder decodes the data into h.264, while the "Decode" module decodes the h.264 bitstream as raw RGB24 data. Since the h.264 bitstream has high compression rates for video data, it is applied to reduce the total amount of transferred data in the system. To achieve higher compression rates, h.264 attempts to take advantage of temporal redundancy between neighboring frames. It contains I-frame (intra-coded frame), P-frame (predicted frame) and B-frame (bi-predictive frame). I-frame is a fully specified picture, whereas P-frames and B-frames rely on the neighbor frame to encode

and hold only part of the image information. Therefore, when the P-frames or B-frames are lost during the transmission, it needs to wait for the next I-frame to decode in the "Decode" module. This situation is due to the delay in the system. If the bitstream data are successfully decoded into raw RGB data, then viewer would read the data and display it on the screen.

Moreover, the bitstream data can also be saved as a file by the "Remux" module in the decoder package. Figure 4.7 shows that the "Remux" module adds the standard format header and trailer for the bitstream data and encodes the h.264 bitstream into an mp4 file stored on the PC. Note that when the "Remux" module is exited or abnormally interrupted, the trailer of the mp4 format cannot be added to the end of the file. In this case, the live streaming video cannot be successfully saved. Since the "Remux" module uses the captured video data, it does not utilize additional network bandwidth to record the video.

The decoder package is implemented using the fast forward MPEG (FFmpeg) library [4] and open source computer vision (OpenCV) library [6], which are the two decode cores in the decoder package. FFmpeg is a video/audio codec library. In the 3D player, it can de-mux and decode the video quickly. It can also control the video to play and pause or change the frame rate by controlling the time base of the video. In addition, it can provide detailed information for the video. In this case, it is the best choice for playing and controlling videos. The other decode core in the 3D player is OpenCV. When OpenCV is set as the video decoder, the "Analysis" module in the decode package is called for mounting the analyst package in the 3D player. This module only works for the OpenCV core. OpenCV is a real-time computer vision. It utilizes the FFmpeg library to encode and decode videos, and it packages the raw data into its own data type. As a widespread computer vision library, its own data type cannot only be analyzed using its own analysis algorithms but is also compatible with other libraries, such as the convolutional architecture for fast feature embedding (Caffe) library [22]. Although OpenCV is not good enough to control the video, it is necessary for the 3D player when the video needs to be analyzed. Therefore, it greatly expands the player from a regular video player to an intelligent video player.

4.2.5 Analyst

The analyst package is used to extract the useful information from the video for the emergency doctor through computer vision algorithms.

This package can be mounted to the 3D player through the "Analysis" module in the decode package. It receives the video frame as an OpenCV image format and outputs an image that is marked with the analysis result. To display the video normally, this package only adopts the real-time analyst algorithm. This package is designed based on the abstract factory pattern, which provides a way to encapsulate a group of individual analysis algorithms that have a common interface without specifying their concrete classes. These analysis algorithms share the same base class and contain the same interface function called the "analyst". The factory pattern is beneficial for adding a new analysis algorithm. The new algorithm only needs to inherit the base class and make a simple registration in the base class. The analysis algorithm is chosen by the parameter of the controller package in the 3D player. When the analysis algorithm is chosen, the parameters of the analysis algorithm are loaded by its constructor, and each frame is then sent to the "analyst" function as input.

For now, we only applied a self-developed face detector in the analyst package. The face detector can extract the patient face directly from the stereo video. The extract patient face area is able to be resized as a larger image, which can let the emergency doctor take a closer look at the patient's facial information. In addition, the face detection is the necessary steps for facial analysis. With the patient face video, it is possible to detect patient heart rate [3] and extract the patient previous treatment information in the dataset through face recognition [39]. The face detector is implemented with Caffe library with deep learning technology, while the detailed principle of the face detector is introduced in Chapter 5.

Chapter 5

Multitask CNN (MTCNN) For Face Detection and Alignment

The video analyst algorithms aim to enhance the information of the video and help the doctor have a better and intuitive understanding of the first aid scene. In this chapter, we present a state-of-the-art face detection algorithm that is contained in the 3D player and prepares for enlarging the patient's face, identifying patients and further extracting patient information.

The algorithm is proposed based on MTCNN, which can detect faces and localize face landmarks simultaneously through cascaded CNNs and multitask learning, and the "feature pooling" technique is applied to the MTCNN. In the following sections, the

overall cascaded framework and its CNN architectures are introduced.

5.1 Overall Cascaded Framework

The strategy of the cascaded framework is to use the shallow neural network to reject most candidates quickly and the complex neural network to predict limited candidates precisely. The small neural networks can greatly save time, while the complex neural network is able to provide face classification with high accuracy. The pipeline of the cascaded framework is the same as MTCNN, which is shown in Figure 5.1.

When the test image is input into the network, it is first resized to a different scale as an image pyramid. The scale factor in this step is related to the acceptable minimum face size, which is shown in Equation 5.1.

$$S_i = \frac{s^i}{F} \tag{5.1}$$

where S_i is the scale factor used to resize the image, s is the scale factor between the image pyramid, i is the series of the image pyramid and begins with 0, and F is the acceptable minimum face size. s is greater than 1.0, while the scaled image should not be smaller than the acceptable minimum face size. For example, when the acceptable minimum face size is 40×40 pixels, the image size is 800×600, s equals 2, and S_i should not be larger than $\frac{\min(800,600)}{40} = 15$. Therefore, i is an integer from 0 to 3.

The scaled image is first input into the proposal network (P-Net). P-Net obtains the candidate windows and their bounding box regression vectors through a very shallow CNN. P-Net crops the candidate windows and resizes the candidate windows to 12×12 as the input image. For example, if the size of the scaled image is $W \times H$ and the stripe between the candidate windows is 4 pixels, it would obtain $(\frac{W-12}{4} + 1) \times (\frac{H-12}{4} + 1)$ candidate windows through P-Net. The candidate windows are calibrated based on the bounding box regression vectors. Then, non-maximum suppression (NMS) is employed to merge the highly overlapped candidates. After P-net, more than 90% of the detection windows are rejected, and the remaining candidates are fed to refine network (R-Net).

NMS is applied to merge the highly overlapped candidates and is calculated by the

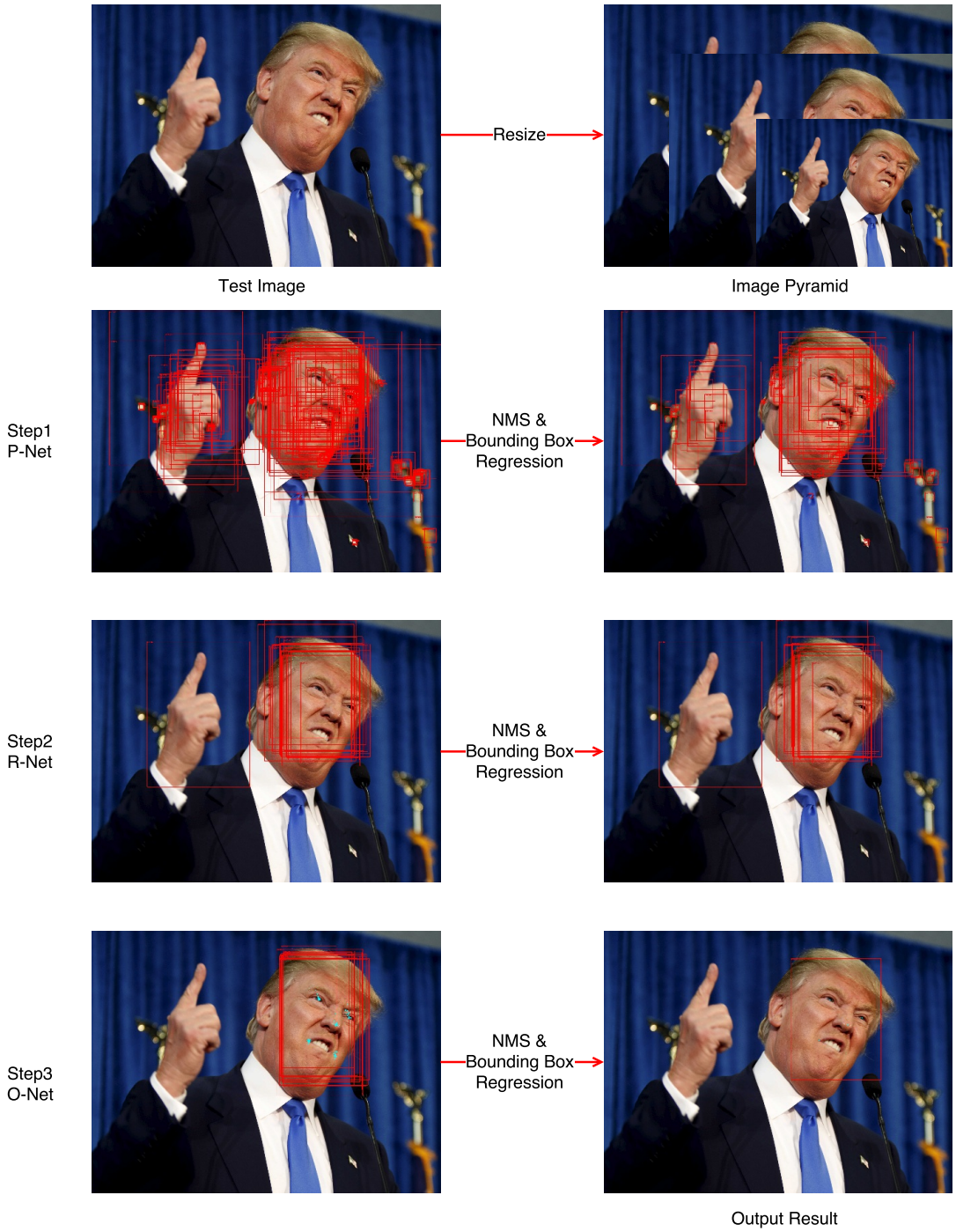


Figure 5.1: The pipeline of the cascaded framework.

intersection over union (IoU) and intersection over minimum (IoM). The IoU calculates the overlap region between two candidates through the ratio of intersected areas to joined areas, whereas the IoM calculates the overlap region between two candidates through the ratio of intersected areas to minimum areas. The equations of the IoU and IoM are shown in Equation 5.2.

$$\begin{cases} IoU(c_i, c_j) = \frac{area(c_i) \cap area(c_j)}{area(c_i) \cup area(c_j)} \\ IoM(c_i, c_j) = \frac{area(c_i) \cap area(c_j)}{\min(area(c_i), area(c_j))} \end{cases} \tag{5.2}$$

where c_i, c_j represent two different candidate windows. When the IoU and IoM are higher than a pre-defined threshold, the compared windows are considered to be overlapped and need to be merged as a large candidate window.

Refine network (R-Net) further rejects a large number of false candidates and reduces the number of detection windows. The candidate windows from P-Net are cropped and resized into 24×24 pixels as input images for the CNN. Similar to P-Net, R-Net performs calibration with bounding box regression, and it utilizes NMS to merge overlapping candidates.

Output network (O-Net) is the last CNN to produce a final bounding box and facial landmark positions. At this stage of the cascade, it is feasible to apply a more powerful but slower CNN. O-Net receives 48×48 pixels as input images and outputs the positions of five facial landmarks to describe the face with more details. Similar to the previous two networks, R-Net also calibrates the candidate windows with bounding box regression, and it merges overlapping candidates with NMS.

5.2 CNN Architectures

In Zhang's work, MTCNN was used to save computational resources for face detection and face regression and to utilize the inherit correlation between the face detection and alignment to improve the detection performance. [50] However, the real-time performance might be limited by the large-scale convolution computation. Therefore, the "feature pooling" technique is adopted to optimize the computational resources and improve the

performance through combining features.

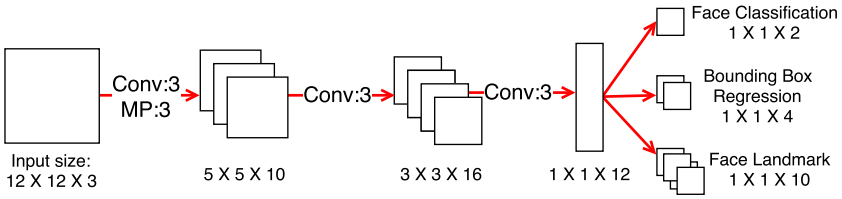
The "feature pooling" technique adds a 1×1 convolution layer between the original convolution layers. The 1×1 convolution was proposed by Lin *et al.*. [28] The 1×1 convolutions are used to spatially combine features across features maps after convolution; thus, they effectively use very few parameters, which are shared across all pixels of these features. The power of the 1×1 convolution can greatly increase the effectiveness of individual convolutional features by combining them into more complex groups. Moreover, it suffers from less over-fitting due to the smaller kernel size (1×1). This idea is widely used in most recent architectures, such as ResNet [14], Inception [20] and their derivatives.

Based on the "feature pooling" technique, the last powerful but slower CNN is slightly changed, and all the CNN architectures are shown in Figure 5.2, where Figure 5.2a, Figure 5.2b and Figure 5.2c show the architectures of P-Net, R-Net, and O-Net, respectively. In these figures, "Conv" means the convolutional layer, and "MP" means the max pooling layer. The numbers following "Conv" and "MP" are the kernel sizes of the convolutional layer and the max pooling layer. The numbers for each step are the feature map sizes after the convolutional layer and the max pooling layer. The upper red dotted box in Figure 5.2c is changed by the lower red box.

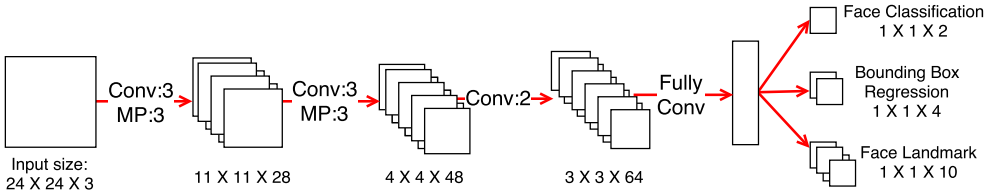
The "feature pooling" is applied in the red dotted box shown in Figure 5.2c. Although the outputs from the upper red box and the lower red box are the same, the lower box, which utilizes the 1×1 convolutional layer, can greatly reduce the computations. Due to the exponential increase in the number of the features, the feature pooling can reduce multiple times before passing data to the expensive convolutional layer by reducing the number of features. For example, in the red dotted box shown in Figure 5.2c, it has 128 features coming in and 256 features coming out. Before the "feature pooling" strategy, the 2×2 convolutional layer costs 131072 times multiply operations based on Equation 5.3.

$$O = IN_{feature} \times OUT_{feature} \times K^2 \tag{5.3}$$

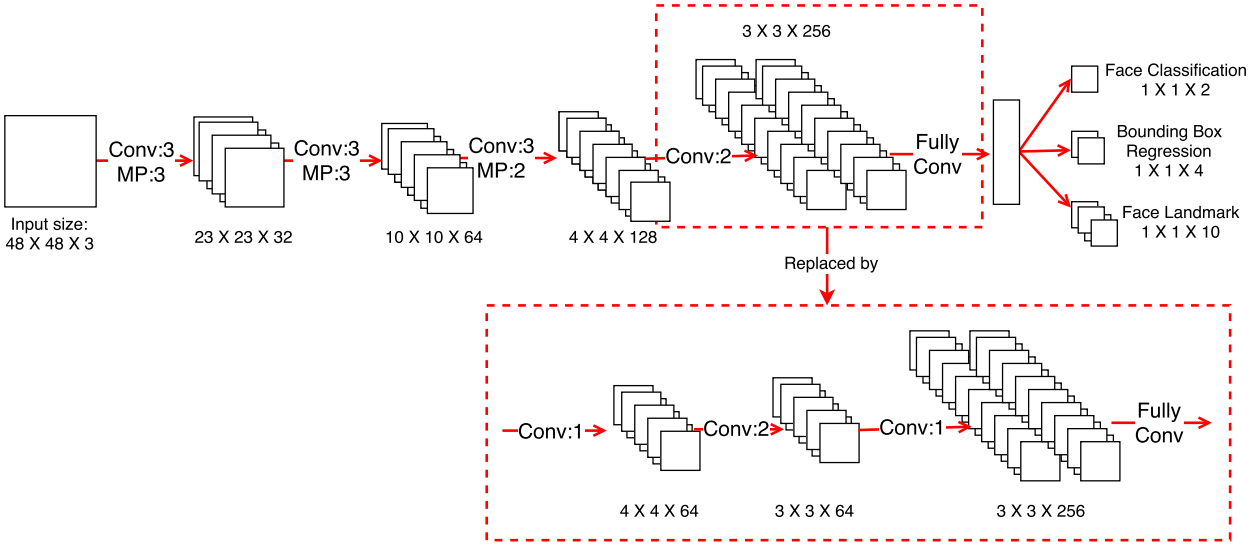
where O represents the operation times, and $IN_{feature}$, $OUT_{feature}$ and K represent



(a)



(b)



(c)

Figure 5.2: The CNN architectures in the cascaded framework. (a) P-Net, (b) R-Net, and (c) O-Net.

the numbers of input feature, output feature and kernel size, respectively.

When the 1×1 convolutional layer is applied before and after the expensive convolutional layer, the operation times decrease to $8192 + 16384 + 16384$, where these three numbers are the operation costs for the 1×1 convolutional layer, the 2×2 convolutional layer and the 1×1 convolutional layer in the lower red dotted box, respectively. Here, the first 1×1 convolutional layer is used to reduce the number of features to 64 features before the expensive 2×2 convolutional layers, while the second 1×1 convolutional layer is applied to reconstruct the output to 256 features. Through feature pooling, the computational resources are saved 8 times.

In addition, the loss function for face classification in the three frameworks uses cross-entropy loss, which is shown in Equation 5.4.

$$L_i = -(y_i \log(P_i)) + ((1 - y_i)(1 - \log(P_i))) \tag{5.4}$$

where L_i is the cross-entropy loss value of the input image, y_i represents the ground-truth label, and p_i donates the probability of a face. The loss functions of bounding box regression and face landmarks are calculated by the Euclidean loss for each sample.

5.3 Training Strategy

The CNN models are trained based on the large-scale CelebFaces Attributes (CelebA) dataset [29]. The images used to train the CNN are divided into 3 groups: positive face, negative face and part of the face. When the intersection-over-union (IoU) ratio of the cropped image is higher than 0.7, lower than 0.3, or between 0.3 and 0.7 to any ground truth provided by the dataset, it belongs to the positive face, the negative face or part of the face, respectively. The general training process is shown in Figure 5.3, where the boxes present the neural networks in the cascade framework, and the red arrows represent the input data for the networks. The training process consists of the following steps.

- Cropping the images from the dataset randomly and dividing them into positive face, negative face and part of the face based on the IoU between the ground truth and cropped image;

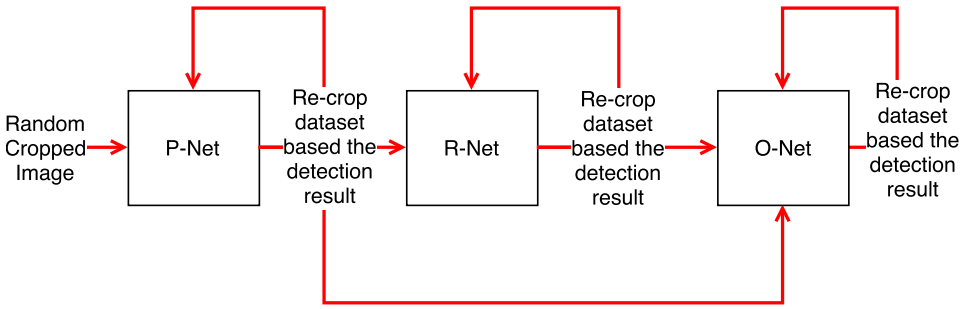


Figure 5.3: The training process for the cascaded framework and each CNN architecture.

- Training the P-Net based on the randomly cropped images;
- Cropping the image from the dataset based on the detected bounding box of the P-Net, dividing it and utilizing it to fine-tune the P-Net;
- Training the R-Net based on the data used to fine-tune the P-Net;
- Cropping the image from the dataset based on the detected bounding box of the R-Net, dividing it and utilizing it to fine-tune the R-Net;
- Training the O-Net based on the data used to fine-tune the P-Net and R-Net;
- Cropping the image from the dataset based on the detected bounding box of the O-Net, dividing it and utilizing it to fine-tune the O-Net;

Note that only parts of the detected bounding box are used to fine-tune each CNN model. Hard sample mining is employed, where its effectiveness was proven in Zhang's work [50]. The basic idea of hard sample mining is that the face detection model is learned with this training set and subsequently applied to all negatives to harvest false positives. The false positives are then added to the training set, and then the model is trained again. This process is iterated several times until satisfaction. In this case, the definitely positive sample and the negative sample would not repeat to fine-tune the CNN model. Moreover, to avoid over-fitting and increase the diversity of the data, the classified data are randomly sampled to fine-tune the CNN model.

Chapter 6

Evaluation and Results

In this chapter, the network environment for the video communication of the stereo monitoring system is first tested, and the quality of the stereo monitoring system is evaluated through multiple objective video quality assessments. The efficiency and accuracy of the face detection algorithm are also tested. In the following sections, each part is introduced separately.

6.1 The network environment for the video transmission

The stereo monitoring system captures the video from paramedic helmet and transfers the video to the emergency doctor through the wireless network. The network environment determines the speed of doctor receive information and give the response. This section evaluates the network environment for the video transmission between the paramedic and emergency doctor.

6.1.1 Experiment Design and Setup

The network environment test for the video transmission check the bandwidth and loss during the video transmission with a simulated network. The network simulates the real network environment in the satellite communication and is setup by Teleset.

The experiment is performed in a room, and all the hardware components mentioned in Chapter 3 are adopted to evaluate the 3D video communication of the system. The server box and the 3D laptop are connected to a router through 2.4 GHz Wi-Fi. The logical network diagram is shown in 6.1, where each icon presents a transfer station and the related name is labeled under the icon; the left computer presents the server box, while the right computer represents the 3D PC.

In this experiment, we first placed the helmet and the server box placed at one place stationary and activated the video communication between the server box and the 3D PC. Then we recorded the TCP dump in the server box for a hour to calculate the delay between the sending package and receiving package.

This system is tested in the lowest resolution (640 * 360) and highest resolution (1920 * 1080), with different network environment (different delay and jittery). Since the system is able to transit the stereo and mono video in different network bandwidth, the network delay is tested separately for stereo and mono video.

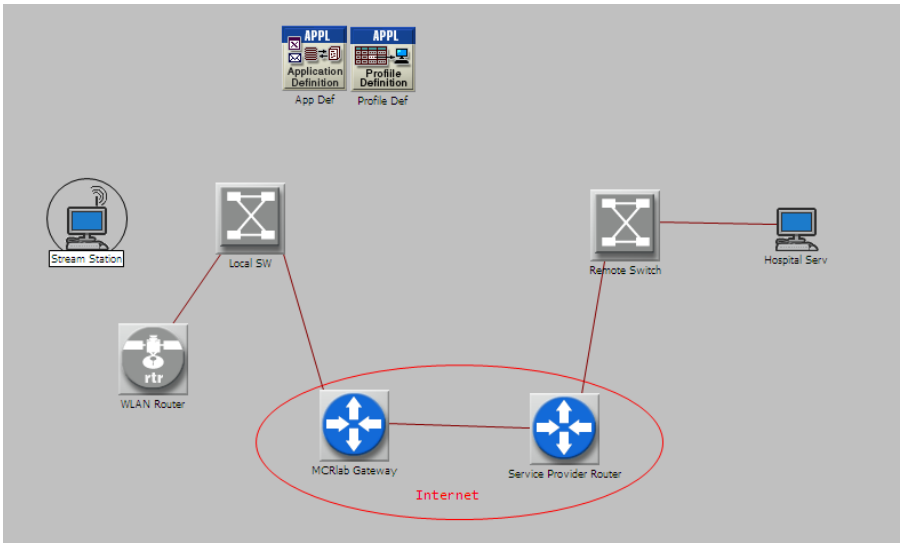


Figure 6.1: The logical network diagram to test the video communication for the system.

6.1.2 Evaluation Result

The evaluation result of this test is shown in Table 6.2, where the mono camera and stereo cameras means that this system; the resolution is the resolution of the transmitted video; the delay and jittery are the delay and jittery happened in the simulated network environment; the forward loss and the bandwidth are the package loss and bandwidth during the video transmission.

	Mono Camera						Stereo Camera					
Resolution (pixels)	640*360		1920 * 1080				640*360		1920 * 1080			
Delay (ms)	0	508	0		508		0	508	0		508	
Jittery (ms)	0	0	0	10	0	10	0	0	0	10	0	10
Forward Lost (%)	0.18	0.42	0.90	2.30	0.64	2.63	1.36	0.21	1.82	2.87	1.92	2.95
Bandwidth (Mbits/s)	4.69	4.91	4.93	4.06	5.23	4.57	8.95	10	10	10	12	8.52

Figure 6.2: The network test for the video transmission.

In this test, the system is tested for low resolution video (640 * 360) and high resolution video (1920 * 1080), while in Table 6.2, the bandwidth for different resolution video

is similar. This is caused by the compression strategy of the system: the current system uses the compression units in the binocular camera directly to compress video without delay, which is optimized for high resolution video. It could achieve high compression rate in high resolution, while the compression for the low resolution video is limited to perform as good as well as the high resolution video. However, since the 4G network has an improved speed of internet connection with a download rate ranging from 17 to 71 Mbit/s, an upload rate ranging from 5 to 43 Mbit/s[9], the bandwidth of high resolution stereo is able to transmit in the 4G network.

In addition, when there are more jittery and delay appeared in the network, it appears higher package lost shown in Table 6.2. The highest package lost is 2.95% appeared when the high resolution stereo video transmits in the 508 ms delay and 10 ms jittery. The package lost influence the quality of video which is saw by the emergency doctor.

This system is also compared with the CAmbulance system [9] shown in Table 6.3, where resolution, FPS and bandwidth means the resolution, frequency and bandwidth in the video transmission.

	CAMbulance		Our			
Resolution (pixels)	240*190	960 * 540	640*360		1920 * 1080	
FPS (Frame/s)	12~19	18 ~ 25	30			
Bandwidth (Mbits/s)	15 ~ 21 (Mono)	21~43 (Mono)	4.5~5 (Mono)	8.5~10 (Stereo)	4.5~5.3 (Mono)	8.5~12 (Stereo)

Figure 6.3: The evaluation comparison between our system and previous system (CAmbulance system).

In Table 6.3, the resolution, frequency and bandwidth in different mode are better that the previous system. In fact, our system has 4 times resolution than the CAmbulance system (1920 * 1080 verse 960 * 540), while the bandwidth our system used is 4 times less than the CAmbulance system.

6.2 Video quality evaluation for the encoded and transmitted video

Since the original video would be distorted as a result of the video transmission and processing, video quality is a factor that is used to evaluate the negative impact of the received video compared with the original video. Therefore, the quality of the video is evaluated by comparing the image sequences of the video with the sequences of the original video. In this section, the video quality in the stereo video system is evaluated.

6.2.1 Experiment Design and Setup

The experiment is performed in a room, and all the hardware components mentioned in Chapter 3 are adopted to evaluate the 3D video communication of the system. The server box and the 3D laptop are connected to a router through 2.4 GHz Wi-Fi, and the distance between the server box and the router is 2 to 10 m.

(a) (b)

Figure 6.4: Example of recorded video from helmet and 3D player. (a) The image recorded from the helmet (original). (b) The image recorded from the 3D laptop (distorted).

Prior to the experiment, an application is constructed to obtain the stereo video from the helmet and the 3D player synchronously. During this experiment, a participant was asked to stand, walk around, rotate their head and look at multiple humans with

the helmet while the application records the four groups of videos simultaneously. An example of the recorded video from the helmet and 3D player is shown in 6.4, where Figure 6.4a and Figure 6.4b present one frame of the original video recorded from the paramedic headset and the 3D PC.

Each video records 100 frames to be tested, and the recorded video resolution is 640 * 480. After the experiment, the four groups of original video and received video are used to measure the video quality of the stereo monitoring system through the objective quality assessment, and all of the assessment methods use YUV420P color space to measure the result.

6.2.2 Metric and Corresponding Result

6.2.2.1 Spatial (SI) and temporal perceptual information (TI) of the recorded video

To have detailed information for the four groups of videos, the TI and SI of the recorded video are calculated based on ITU-T, P.910 [35]. Since the TI and SI can provide the capacity of the video, these features are important to the size of compressed video and the level of impairment during the video compression and transmission. In this case, the TI and SI of the original video are calculated using the following equations.

The SI is computed based on the Sober filter, which is shown in Equation 6.1.

$$SI = \max(\sigma(Sobel(F_n))) \tag{6.1}$$

where F_n represents the frame of the video sequence, $Sobel$ means that each frame is filtered by the Sober filter, σ is the standard deviation over each pixel in the filtered frame, and max is the maximum value of the SI in each frame to represent the SI for the video sequence. When the SI is higher, the frames in the video contain more detailed texture where the video contains more data after compression and transmission. For example, a video containing a tree has a higher SI than a video containing an empty sky.

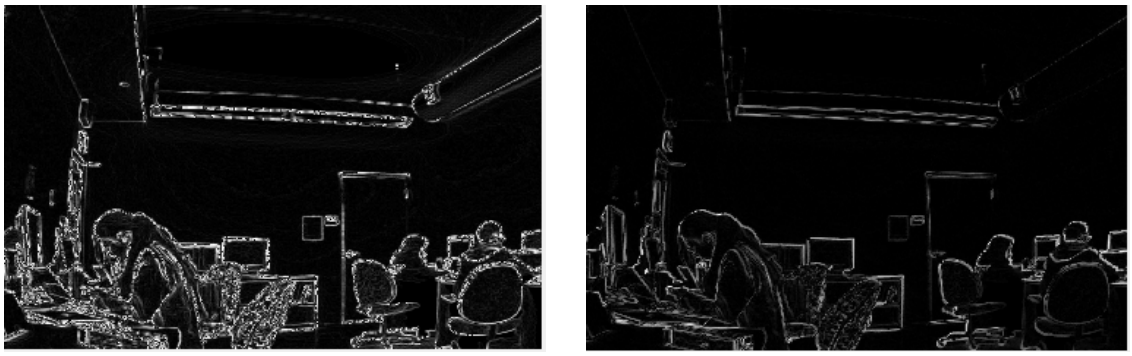
The TI is used to frame the difference between two adjacent frames to measure the

motion difference feature, which is shown in Equation 6.2.

$$TI = \max(\sigma(F_n(i,j) - F_{n-1}(i,j))) \tag{6.2}$$

where $F_n(i,j)$ represents the pixel in the frame of the video sequence, σ is the standard deviation over each pixel in the filtered frame, and max is the maximum value of the TI in each frame to represent the TI for the video sequence. When the TI is higher, the video sequence contains more motions and needs more data to compress and transmit the video. For example, a video that contains a human fighting has a higher TI than a video that contains a human standing.

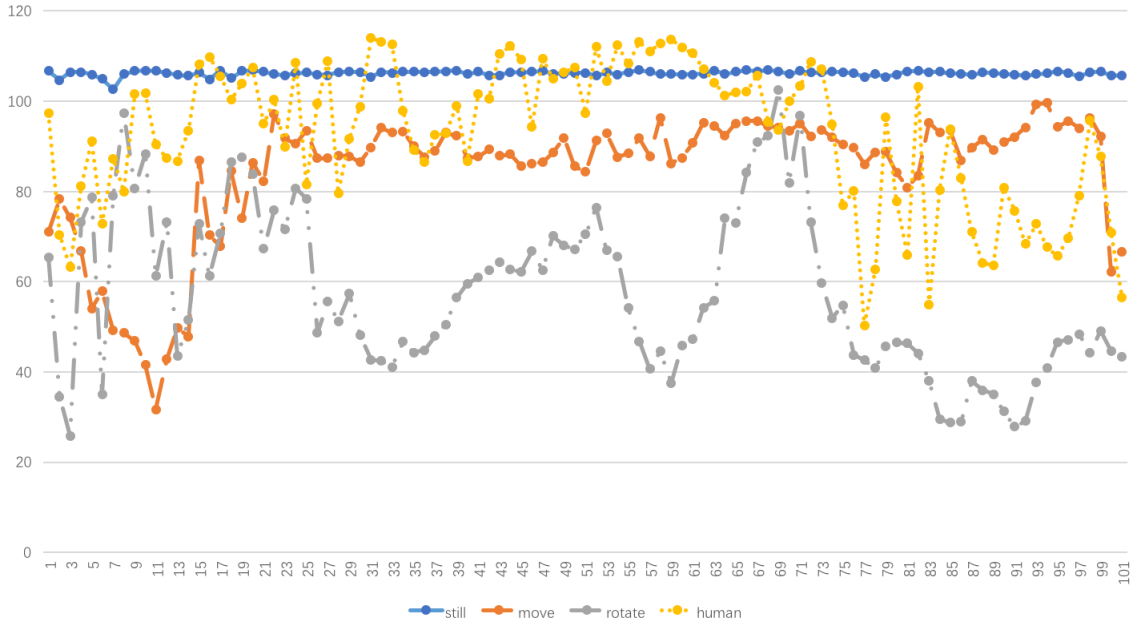
Examples of the SI and TI in the recorded video are shown in Figure 6.5, where Figure 6.5a and Figure 6.5b are the SI and TI calculation results in one frame, respectively. The grayscale values of each pixel in these two figures are used to calculated the values of SI and TI of the corresponding frame. Black means 0, while white means 255.



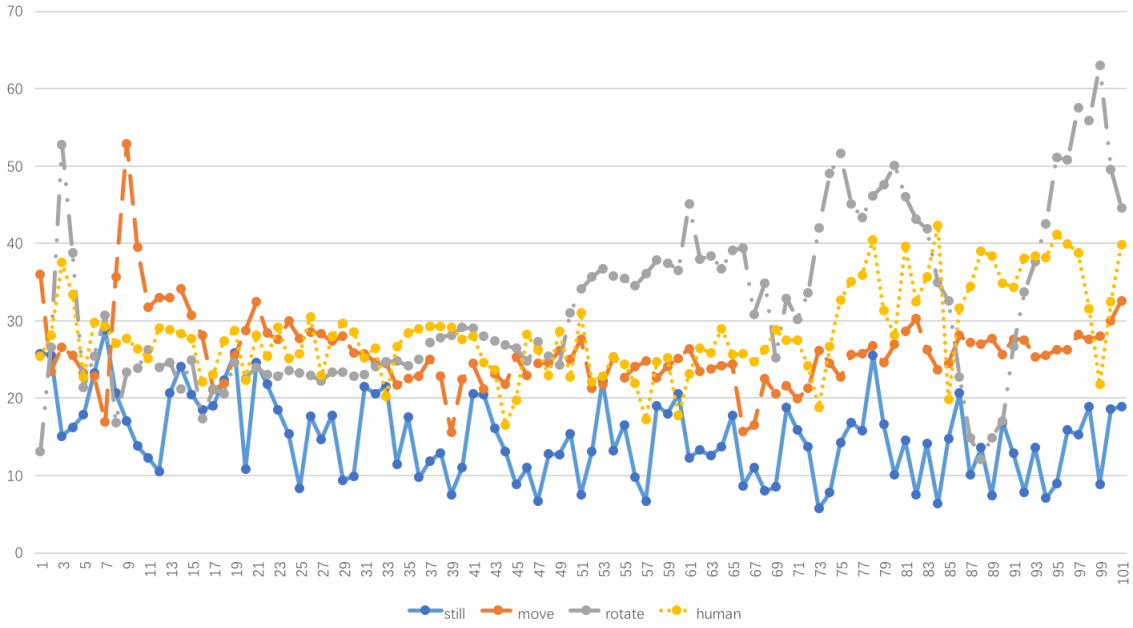
(a) (b)

Figure 6.5: Example SI and TI results in the recorded video. (a) SI. (b) TI.

The SI and TI results of each frame in the four groups of videos are shown in Figure 6.6, where Figure 6.6a and Figure 6.6b are the SI and TI calculation results in one frame, respectively; the solid blue line, dashed orange line, dash-dotted gray line and dotted yellow line represent the "still", "move", "rotate" and "human" recorded videos, respectively. The average SI in the four groups of videos are 106.17, 84.43, 57.29 and 92.90, while the average TI are 15.01, 25.91, 31.83 and 28.46, respectively. The standard deviation of SI in the four groups of videos are 0.58, 14.42, 18.08 and 16.09, while the



(a)



(b)

Figure 6.6: The SI and TI results of each frame in the four groups of videos. (a) SI. (b) TI.

standard deviation of TI are 5.35, 4.85, 10.74 and 5.75, respectively. The maximum of SI in the four groups of videos are 106.92, 99.61, 102.47 and 113.97, while the maximum of TI are 28.80, 52.92, 63.04 and 42.30, respectively.

The SI and TI results show the diversity of the data streaming. The "still" video keeps a high SI value and low TI value. The "move" video adds a small vibration in the SI and TI result curves, whereas the "rotate" video adds a more obvious vibration. In this case, the volume of the image data in the "still" video almost keeps the same size in the data compression and transmission procedure. The image data in the "move" and "rotate" videos would have a higher or lower data size during the compression and transmission procedure. The "human" video reflects the paramedic situation in which patients appear in the video.

With the basic information of the test videos, the detailed evaluations of the videos are analyzed in the following subsections.

6.2.2.2 Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR)

MSE and PSNR are the algorithms that have historically been adopted in image processing to evaluate the performance of the distortion of the frame in each pixel during the video compression and transmission processes.

MSE calculates the deviation between the undistorted and distorted videos to approximate the expected distortion, as shown in Equation 6.3.

$$MSE = \frac{\sum_{i=0}^{W-1} \sum_{j=0}^{H-1} (F_{i,j} - F'_{i,j})}{W \times H} \tag{6.3}$$

where $F_{i,j}$ and $F'_{i,j}$ represent pixels in the comparison frame of the undistorted and distorted videos, respectively, and W and H are the width and height of the frame, respectively. In statistics, this equation represents the expectation value of the square of the error. The distortion during the data compression and transmission is the positive correlation with the MSE value. In this case, when the video after compression and transmission is close to the original video, the MSE value is small.

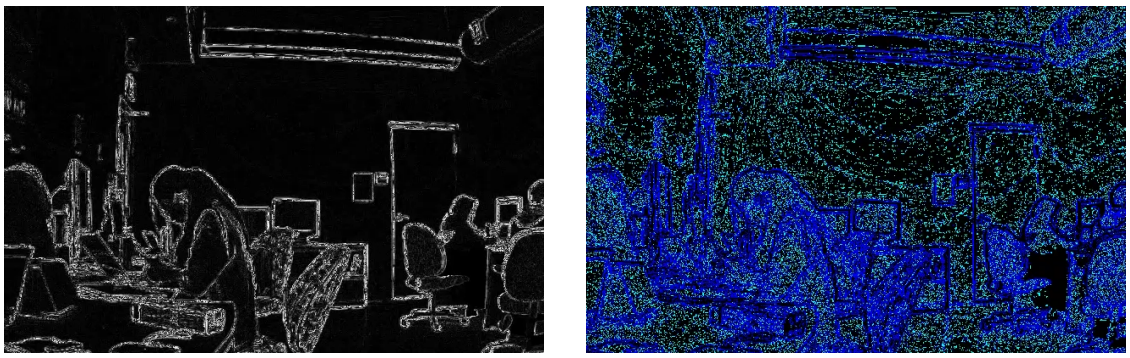
PSNR is the logarithmic form of the MSE, and its measurement is shown in Equation

6.4.

$$PSNR = 10 \log_{10} \frac{MAX_F^2}{MSE} \tag{6.4}$$

where MAX_F is the maximum possible pixel value of the frame in the video. Since this stereo system use 8 bits per sample to save the data for each pixel, $MAX_F = 2^8 - 1 = 255$. A small MSE value results in a high signal-to-noise ratio; if the MSE tends to zero, then the PSNR tends to infinity. A larger PSNR means a smaller difference between the distorted video and original video.

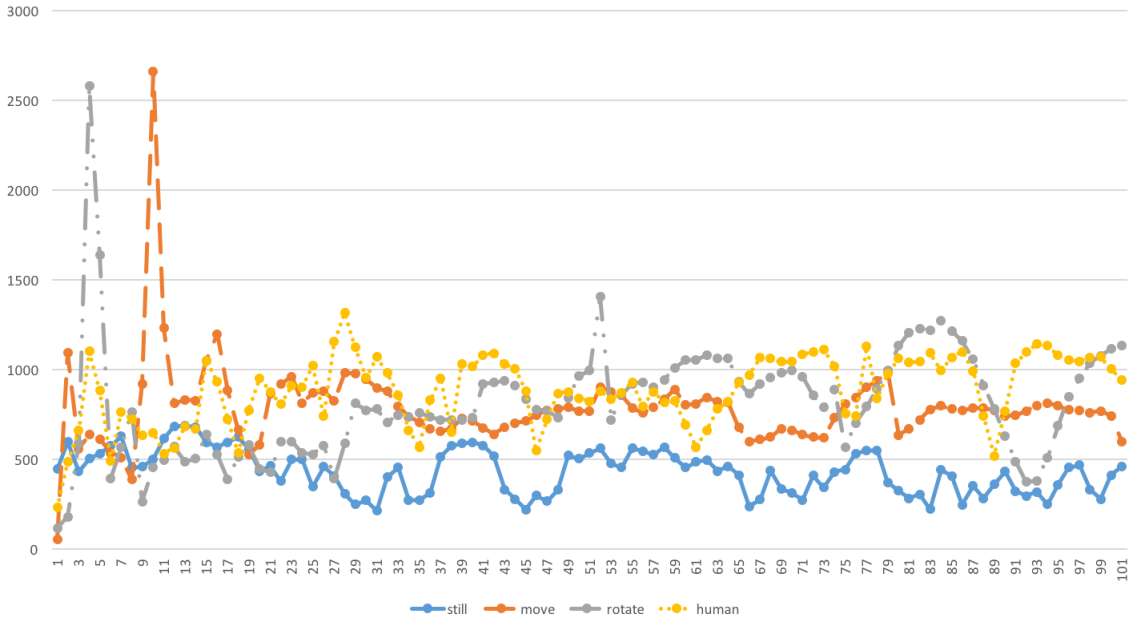
Examples of the MSE and PSNR in the recorded video are shown in Figure 6.7, where Figure 6.7a and Figure 6.7b are the MSE and PSNR calculation results in one frame, respectively. The grayscale value in Figure 6.7a is used to present the MSE value in the corresponding frame, which is increasing from black to white. The colors in Figure 6.7b represent the PSNR value, which is increasing based on the order red, yellow, green, blue, and black.



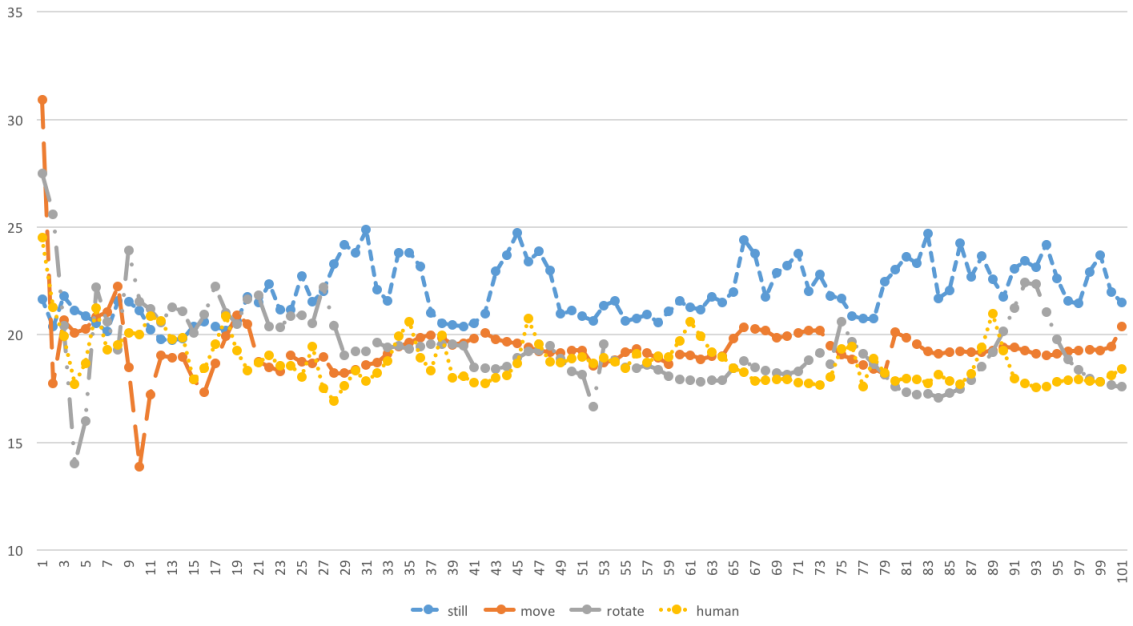
(a) (b)

Figure 6.7: Example MSE and PSNR results in the recorded video. (a) MSE. (b) PSNR.

The MSE and PSNR results of each frame in the four groups of videos are shown in Figure 6.8, where Figure 6.8a and Figure 6.8b are the MSE and PSNR calculation results in one frame, respectively; the solid blue line, dashed orange line, dash-dotted gray line and dotted yellow line represent the "still", "move", "rotate" and "human" recorded video, respectively. The average MSE in the four groups of videos are 430.94, 784.99, 810.63 and 855.27, while the average PSNR are 21.97, 19.38, 19.39 and 18.79,



(a)



(b)

Figure 6.8: The MSE and PSNR results of each frame in the four groups of videos. (a) MSE. (b) PSNR.

respectively. The standard deviation of MSE in the four groups of videos are 121.04, 242.53, 321.56 and 195.14, while the standard deviation of PSNR are 1.30, 1.49, 1.86 and 1.13, respectively. The maximum of MSE in the four groups of videos are 685.35, 2659.42, 2578.50 and 1315.26, while the maximum of PSNR are 24.88, 30.92, 27.50 and 24.50, respectively.

The average MSE and PSNR of the four groups are acceptable in the wireless transmission system, where acceptable values for wireless transmission quality loss are considered to be approximately 20 dB to 25 dB [40]. The low standard deviation indicates the stability of the system. Although the SI value is high in the "still" video, its performance is quite reliable. There are two bad frames in the "move" and "rotation" videos, where the content between these two frames changes considerably and has a high value in TI.

The MSE and PSNR have favorable properties. These methods have a clear physical and desirable meaning: these methods are the natural way to define the energy of the error signal and are a desirable measurement in the statistics and estimation framework. However, the MSE and PSNR evaluation methods cannot account for information communication framework, such as a video communication system that involves lossy compression, noise contamination, and packet loss. [44] In this case, more targeted evaluation methods are required to test the stereo video communication system.

6.2.2.3 Structural Similarity (SSIM) Index and Multiscale SSIM (MSSSIM) Index

Since the frames in the video are highly structured, the SSIM index [46] and MSSSIM index [45] are used to measure the structural information of the video, which is defined as the set of attributes that form the structure of the objects represented in the scene, regardless of the average luminance and contrast.

The SSIM index measures the video based on luminance similarity, contrast similarity and structural similarity and combines them into a result value. The SSIM index is calculated based on the comparison process of the SSIM index, which is shown in Figure 6.9, where F and F' represent the comparison frames of the undistorted and distorted

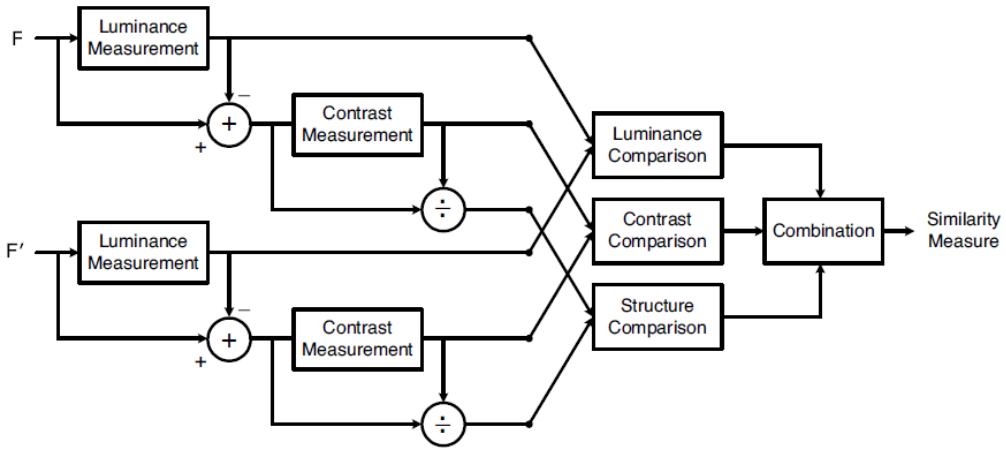


Figure 6.9: The comparison process of the SSIM index.

videos, respectively. The relative formula is shown in Equation 6.5.

$$SSIM = \frac{(2\mu_F\mu_{F'} + C_1)(2\sigma_{FF'} + C_2)}{(\mu_F^2 + \mu_{F'}^2 + C_1)(\sigma_F^2 + \sigma_{F'}^2 + C_2)} \tag{6.5}$$

where μ_F and $\mu_{F'}$ are the luminances of the original frame F and distorted frame F', respectively; σ_F and $\sigma_{F'}$ are the standard deviations of the two frames; the structure comparison function is based on the Pearson's correlation index, which is $\sigma_{FF'}$; and C_1 and C_2 are two coefficients that depend on the maximum value of the image color component MAX_F, where MAX_F is clarified in Equation 6.4. Here, $C_1 = 0.01^2 \times 255^2 = 6.5025$, and $C_1 = 0.03^2 \times 255^2 = 58.5225$. As SSIM approaches 1, the degree of fidelity becomes closer to the original.

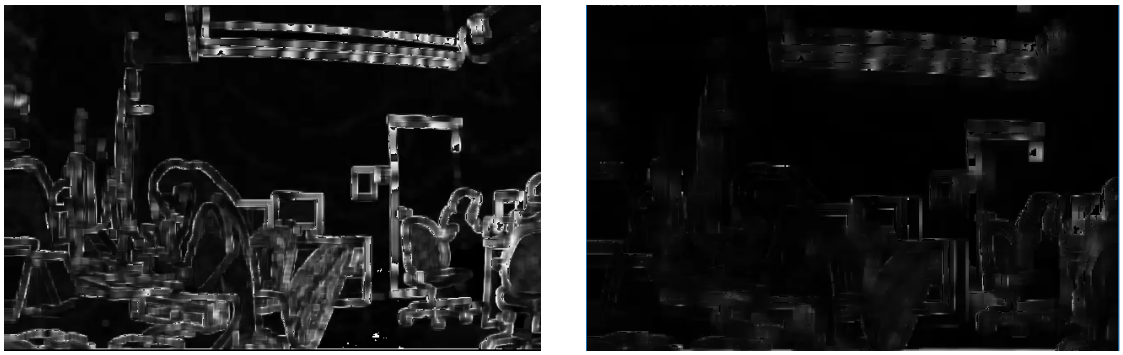
The MSSSIM index is based on the SSIM metric of several downscaled levels of original images. It moves multiple Gaussian windows on the entire image pixel by pixel, producing multiple SSIM results. MSSSIM averages these multiple SSIM results, as shown in Equation 6.6.

$$MSSSIM = \frac{\sum_{m=1}^{M} SSIM_m}{M} \tag{6.6}$$

where M is the number of Gaussian windows. Similar to SSIM, the distorted frame is close to the original one when MSSSIM is close to 1.

Examples of the SSIM and MSSSIM in the recorded video are shown in Figure 6.10,

where Figure 6.10a and Figure 6.10b are the MSE and PSNR calculation results in one frame, respectively, and brighter areas correspond to a greater difference in the grayscale image.



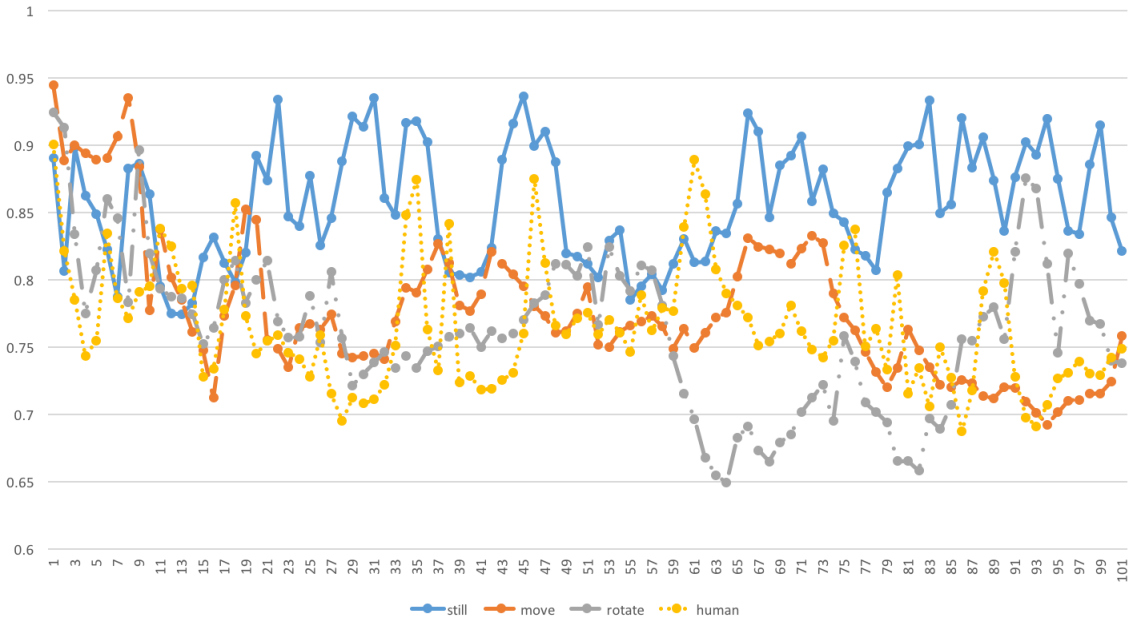
(a) (b)

Figure 6.10: Example SSIM and MSSSIM results in the recorded video. (a) SSIM. (b) MSSSIM.

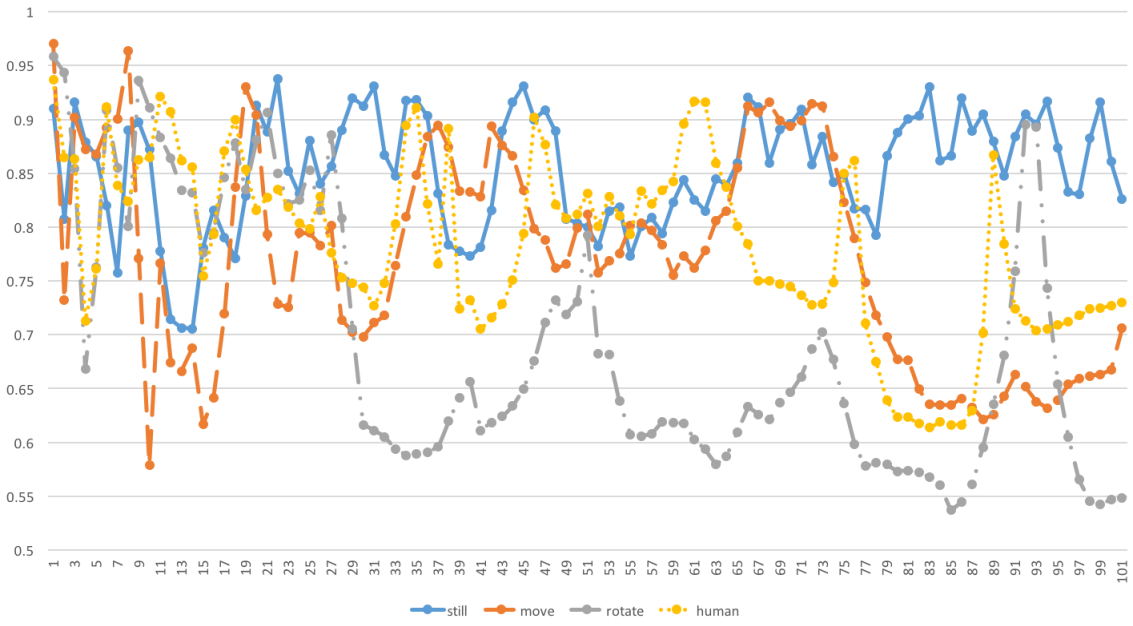
The SSIM and MSSSIM results of each frame in the four groups of videos are shown in Figure 6.11, where Figure 6.11a and Figure 6.11b are the SSIM and MSSSIM calculation results in one frame respectively; the solid blue line, dashed orange line, dash-dotted gray line and dotted yellow line represent the "still", "move", "rotate" and "human" recorded video respectively. The average SSIM in the four groups of videos are 0.86, 0.78, 0.76 and 0.77, while the average MSSSIM are 0.85, 0.77, 0.69, 0.78 respectively. The standard deviation of SSIM in the four groups of videos are 0.04, 0.05, 0.06 and 0.05, while the standard deviation of MSSSIM are 0.05, 0.10, 0.12 and 0.08 respectively. The maximum of SSIM in the four groups of videos are 0.94, 0.94, 0.92 and 0.90, while the maximum of MSSSIM are 0.94, 0.97, 0.96 and 0.94, respectively.

The average SSIM of the four groups of videos are acceptable. Compared to the SSIM result, the MSSSIM result has a more obvious oscillation indicating the stability of the system. The "rotate" video has a quite high standard deviation and low average MSSSIM result, particularly for the frame that has a relatively high TI value. An example bad frame for the SSIM is shown in Figure 6.12.

Hore's study has revealed that the PSNR is more sensitive to additive Gaussian noise



(a)



(b)

Figure 6.11: The SSIM and MSSSIM results of each frame in the four groups of videos. (a) SSIM. (b) MSSSIM.

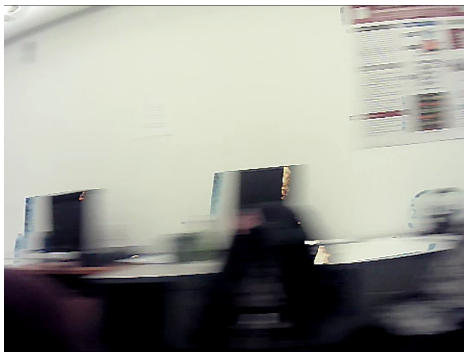


Figure 6.12: Example frame for the SSIM.

than the SSIM, whereas the opposite is observed for jpeg compression. [16] For this 3D helmet monitoring system, although the burden of the transmission is quite high, the impact of wireless transmission on video quality is small and stable due to the MSE and PSNR results. This is also related to the relatively close transmission distance. Because of the high compression of H.264, the distance between the I frame in H.264 is quite far. When the contents of the video change too fast, it would cause obvious distortion in the video. For this situation, the moving object would leave white points.

In addition to the video quality algorithm, such as PSNR and SSIM, this system is also evaluated based on human perception.

6.2.2.4 DCT-based Video Quality Metric (VQM)

To evaluate the stereo video system based on human perception, DCT-based VQM with human spatial-temporal contrast (SCS) sensitivity model is used. [48]

The general procedure of the VQM algorithm is shown in Figure 6.13, where F and F' represent the comparison frames of the undistorted and distorted videos, respectively.

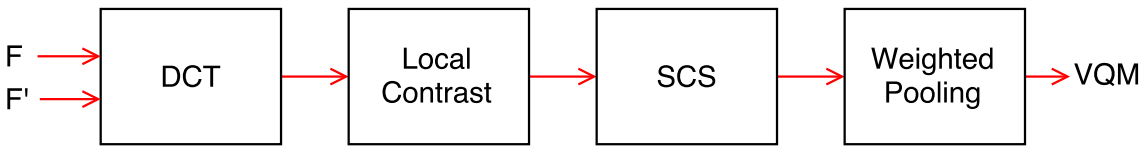


Figure 6.13: The procedure of the VQM algorithm.

The incoming frames are separated into different spatial frequency components through a discrete cosine transform (DCT). The DCT coefficients are converted to just-noticeable

differences by multiplying each DCT coefficient by local contrast and its corresponding entry in the SCS matrix. The MPEG default quantization matrix is applied to the SCS matrix as the contrast threshold matrix, which is the inverse of the contrast sensitivity matrix and shown in Equation 6.7. The final VQM is weighted pooling of the mean and maximum distortion for the differences calculated by the SCS matrix.

$$SCS\ matrix = \begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \tag{6.7}$$

Since the VQM is calculated based on the SCS matrix, it is more sensitive to the block-based distortions, which is close to human perception. The value of VQM is the positive correlation with the difference between the distorted frame and the original frame. When the VQM is smaller, the distorted frame is closer to the original frame.

Examples of the VQM in the recorded video are shown in Figure 6.14, where brighter blocks correspond to a greater difference.

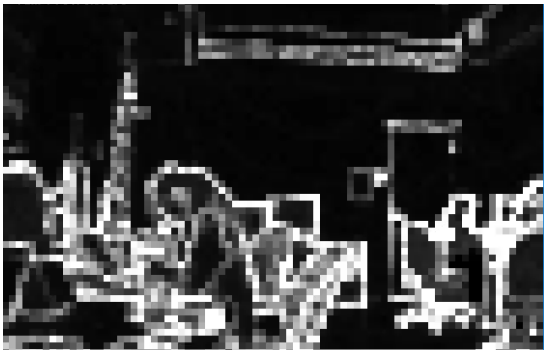


Figure 6.14: Example VQM result in the recorded video.

The VQM results of each frame in the four groups of videos are shown in Figure 6.15, where the solid blue line, dashed orange line, dash-dotted gray line and dotted yellow line

represent the "still", "move", "rotate" and "human" recorded videos respectively. The average VQM in the four groups of videos are 3.97, 5.28, 5.87 and 5.81, its relative standard deviation in the four groups of videos are 0.84, 0.86, 0.81 and 0.81 respectively. The maximum of VQM in the four groups of videos are 6.15, 6.75, 7.29 and 7.27, respectively.

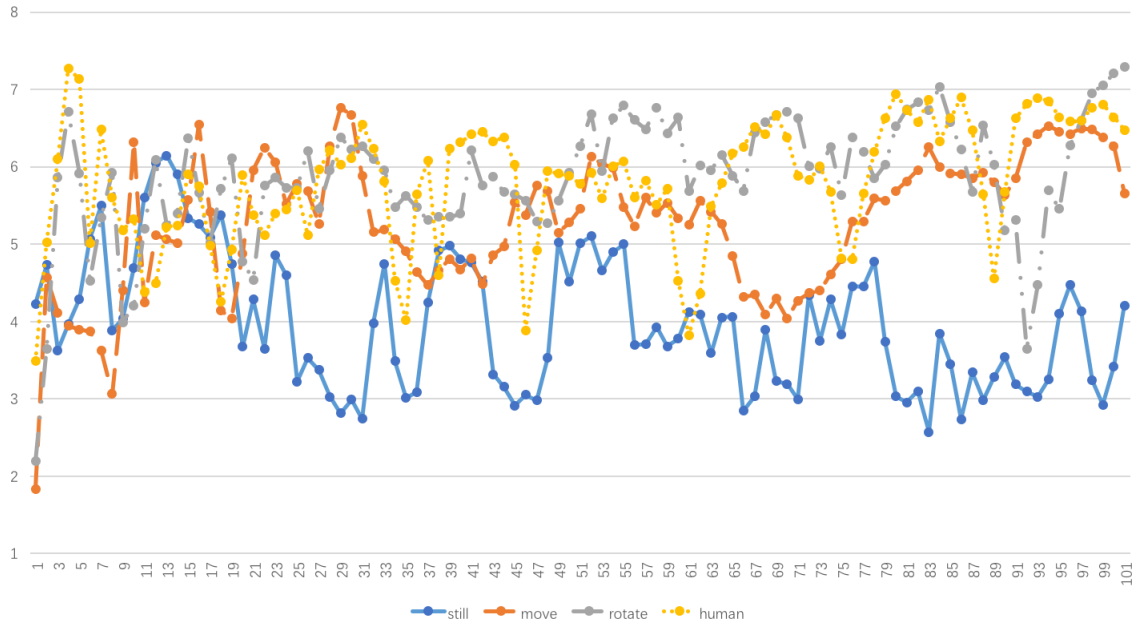


Figure 6.15: The VQM results of each frame in the four groups of videos. (a) MSE. (b) PSNR.

The average VQM of the four groups are acceptable. Compared to the MSSSIM result, the VQM result has more stability. An example of an obvious bad frame in VQM is shown in Figure 6.16, where several blurred squares have appeared in the picture.

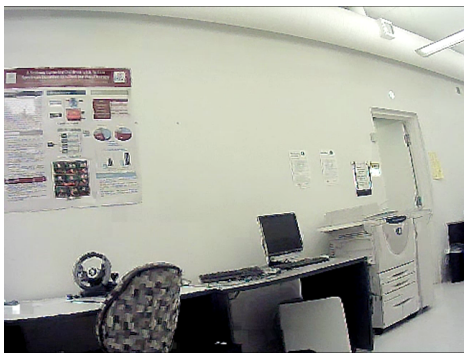


Figure 6.16: Example frame for the VQM.

The general performance of the system is evaluated using multiple evaluation methods. Based on the previous evaluation results, even though there are a few bad frames for the 4 groups of videos, the quality and stability of the video in the stereo monitoring system are acceptable for the wireless transmission system.

6.3 Evaluation for Face Detection

In this section, the proposed face detection algorithm is evaluated. The face detector is compared with the state-of-the-art methods in the Face Detection Data Set and Benchmark (FDDB) [21], which contains the annotations for 5,171 faces in a set of 2,845 images. The efficiency of the face detector is also evaluated.

6.3.1 Accuracy for Face Detection

To evaluate the performance of our face detection method, the face detector is compared with the Viola-Jones face detector [42], CascasdeCNN [27] and Find Tiny Face [15]. The detection result is evaluated by the discrete score provided by FDDB, which is shown in Equation 6.8.

$$\begin{cases} y_i = \delta_{S(d_i,v_i)>0.5} \\ S(d_i,l_i) = \frac{area(d_i) \cap area(l_i)}{area(d_i) \cup area(l_i)} \end{cases} \tag{6.8}$$

where l_i and d_i represent the labeled box and its corresponding detected box, respectively, and $S(d_i, l_i)$ is the overlap region between l_i and d_i through the ratio of intersected areas to joined areas. When the overlap region between l_i and d_i is greater than 0.5, the face is known as a detected face.

The performance of this algorithm in FDDB is shown in Figure 6.17, where the solid blue line, dashed orange line, dash-dotted green line and dotted red line are the evaluation results of the Viola-Jones face detector, Find Tiny Face, CascasdeCNN and MTCNN, respectively. The Viola-Jones face detector is a famous face detector that is implemented by OpenCV. CascasdeCNN is the method that MTCNN is based on. Find Tiny Face is the state-of-the-art algorithm and has achieved the best performance in face detection.

Moreover, except for the Viola-Jones face detector, the other three algorithms are all developed using a convolutional neural network. In addition, since the accuracy of the original MTCNN algorithm is not published in the evaluation website of fddb dataset [21], currently it is not compared with our own algorithm.

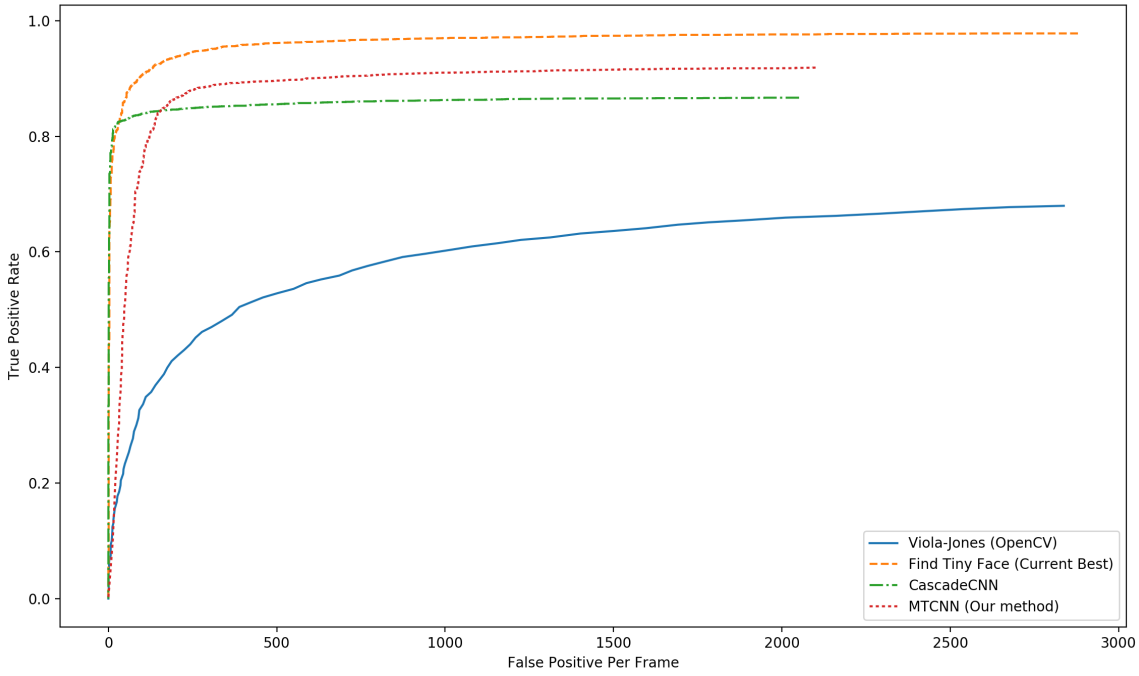


Figure 6.17: The evaluation result on FDDB for the accuracy of face detection.

In Figure 6.17, the face detection accuracies of the four algorithms are 67.98%, 97.78%, 86.68%, and 91.78%. This result shows that the MTCNN has achieved good performance in the FDDB dataset. Note that the face detection accuracy of MTCNN can still be improved through more training and fine-tuning processes because of the relatively low slope at the beginning time.

6.3.2 Efficiency

Based on the given cascade structure, MTCNN can achieve high speed in joint face detection and alignment. Since the CNN architecture has been slightly improved, the speed of MTCNN with feature pooling is also improved. The comparison between the MTCNN without and with feature pooling is shown in Figure 6.18, where the dash-dotted green line and dotted red line are the time cost of the MTCNN without feature

pooling and of the MTCNN with feature pooling, respectively. This result is generated by a machine running Ubuntu 16.04 operating system and a GTX 1080 GPU.

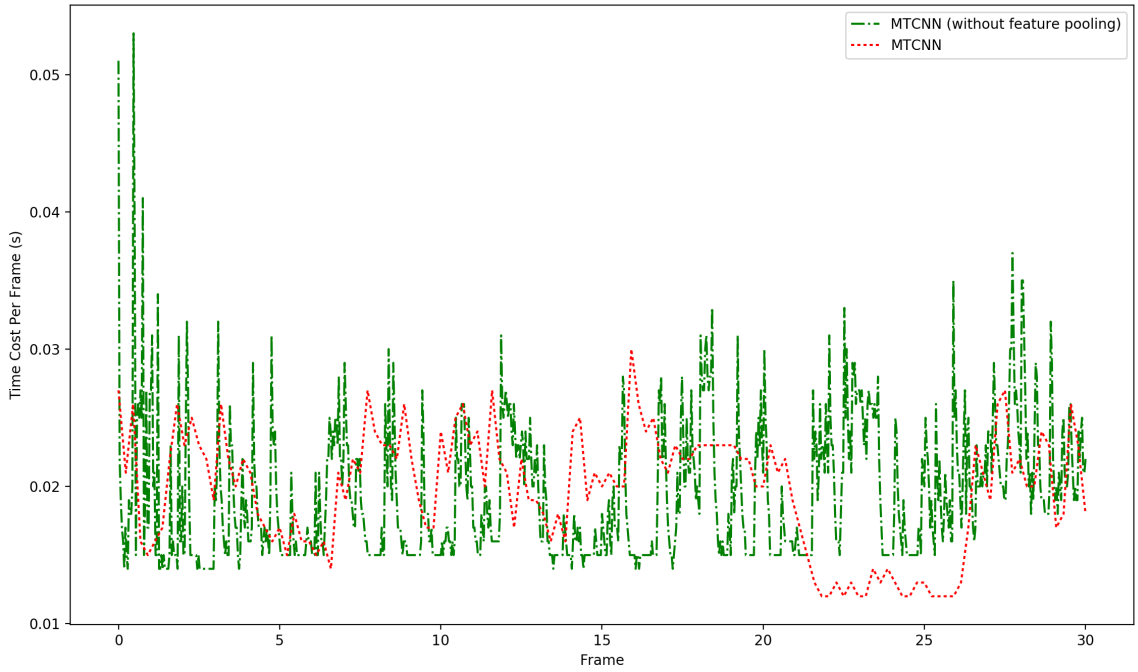


Figure 6.18: The evaluation result for the efficiency of face detection.

In Figure 6.18, the average time costs of these two algorithms are 20.03 and 19.71 ms. This result shows that both of these algorithms have achieved the real-time requirement. In addition, there is a slight improvement for MTCNN with the feature pooling algorithm. Moreover, since the system is developed using C++ with multiple threads and CUDA, the face detector and the 3D player run in parallel, which means that the face detection does not affect the video effects.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis describes an end-to-end stereo monitoring system for a paramedic headset to enhance the treatment procedure for telemedicine in the pre-hospital stage.

This system establishes 3D video communication between paramedics and emergency doctors. The 3D video communication consists of three self-developed hardware components: paramedic helmet, server box and 3D PC. The 3D video is captured and transmitted as a live stream by the stream server subsystem running on the portable server box, while it is displayed as a 3D effect by the 3D player subsystem. Through comprehensive objective video quality assessments for the video communication, it is proven that this

system has a relatively acceptable performance for 3D wireless video communication. We also propose an improved MTCNN face detector that is applied to enlarge patient's face and a more intelligent telemedicine assistance system. Through comparisons with other state-of-the-art algorithms, the experimental results demonstrated that our method has achieved good accuracy and real-time results.

7.2 Future Work

Although this system has implemented the basic requirements for 3D video communication between paramedics and emergency doctors, there are still several works that could be performed to pursue a more robust and intelligent 3D telemedicine assistance system, which are listed as follows:

- The paramedic headset needs to add hardware components that allow audio and haptic communication for paramedics and emergency doctors. Interactive communication between the paramedics and emergency doctors needs to be embedded into the telemedicine system for more convenient use. Meanwhile, the server box also needs to adopt a faster wireless transmission device with larger bandwidth to achieve more stable 3D video communication.

- The stream server should try different real-time communication protocols such as RTMP to test the stability and delay for 3D video communication. The server should also employ a more efficient video codec format to reduce the transmission burden and improve the stability of the video communication.

- More video analysis algorithms need to be applied to the 3D player. For example, a face recognition algorithm could connect the patient to the hospital database, while detecting the heart rate from the face can determine the patient's physical condition. These video analysis algorithms can be adopted by the system to extract patient information to help doctors make the proper decision.

- Objective video quality assessments for 3D live video need to be developed. During our experiments, there is not yet a mature objective video quality analysis method

for 3D live video. However, with the development of VR, AR and 3D devices, 3D live video is gradually becoming possible. The related objective video quality analysis methods must be developed for testing the objective video quality of the 3D live video.

References

[1] *3D Vision Develop Website*. `https://developer.nvidia.com/3d-vision-and-surround-technology`. [Online]. 2017.

[2] *ADLINK Ambulance-based Telemedicine System*. `http://www.adlinktech.com/solution/Ambulance-based_Telemedicine_Saving-Real-Time.php?utm_source=`. [Online]. 2017.

[3] Karim Alghoul et al. "Heart Rate Variability Extraction From Videos Signals: ICA vs. EVM Comparison". In: *IEEE Access* 5 (2017), pp. 4711–4719.

[4] Fabrice Bellard, M Niedermayer, et al. "FFmpeg". In: *Availabel from: http://ffmpeg.org* (2017).

[5] Alexandre Benoit et al. "Quality assessment of stereoscopic images". In: *EURASIP Journal on Image and Video Processing* 2008.1 (2009), pp. 1–13.

[6] Gary Bradski. "The OpenCV library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] Rich Caruana. "Multitask learning". In: *Learning to learn*. Springer, 1998, pp. 95–133.

[8] Henry Chen et al. "3D collaboration method over HoloLens™ and Skype™ end points". In: *Proceedings of the 3rd International Workshop on Immersive Media Experiences*. ACM. 2015, pp. 27–30.

[9] Andrea Corradini and Constantin Alexandru Gheorghiasa. "CAMbulance: A live video streaming system for ambulance services". In: *International Conference on Information and Communication Technology Research (ICTRC)*. IEEE. 2015, pp. 100–103.

[10] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639 (2017), pp. 115–118.

[11] Holly French et al. “Real time video QoE analysis of RTMP streams”. In: *International Conference on Performance Computing and Communications Conference (IPCCC)*. IEEE. 2011, pp. 1–2.

[12] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[13] Dan Grois et al. “Performance comparison of h. 265/mpeg-hevc, vp9, and h. 264/mpeg-avc encoders”. In: *Picture Coding Symposium (PCS), 2013*. IEEE. 2013, pp. 394–397.

[14] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[15] Kaiming He et al. “Finding Tiny Faces”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[16] Alain Hore and Djemel Ziou. “Image quality metrics: PSNR vs. SSIM”. In: *International Conference on Pattern Recognition (ICPR)*. IEEE. 2010, pp. 2366–2369.

[17] Quan Huynh-Thu and Mohammed Ghanbari. “Scope of validity of PSNR in image/video quality assessment”. In: *Electronics letters* 44.13 (2008), pp. 800–801.

[18] Sangheum Hwang and Hyo-Eun Kim. “Self-Transfer Learning for Fully Weakly Supervised Object Localization”. In: *arXiv preprint arXiv:1602.01625* (2016).

[19] *ImGui Website*. `https://github.com/ocornut/imgui`. [Online]. 2017.

[20] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[21] Vidit Jain and Erik G Learned-Miller. “FDDB: A benchmark for face detection in unconstrained settings”. In: *UMass Amherst Technical Report* (2010).

[22] Yangqing Jia et al. “Caffe: Convolutional architecture for fast feature embedding”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678.

[23] Morten Kjelso, Mark Gooch, and Simon Jones. “Design and performance of a main memory hardware data compressor”. In: *Proceedings of Hardware and Software Design Strategies*. IEEE. 1996, pp. 423–430.

[24] Sanna Laine and Ismo Hakala. “H. 264 QoS and application performance with different streaming protocols”. In: *Proceedings of International Conference on Mobile Multimedia Communications*. ICST. 2015, pp. 32–38.

[25] Eric C Larson and Damon M Chandler. “Most apparent distortion: full-reference image quality assessment and the role of strategy”. In: *Journal of Electronic Imaging* 19.1 (2010), pp. 011006–011006.

[26] E Brooke Lerner and Ronald M Moscati. “The golden hour: scientific fact or medical “urban legend”?” In: *Academic Emergency Medicine* 8.7 (2001), pp. 758–760.

[27] Haoxiang Li et al. “A convolutional neural network cascade for face detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 5325–5334.

[28] Min Lin, Qiang Chen, and Shuicheng Yan. “Network in network”. In: *arXiv preprint arXiv:1312.4400* (2013).

[29] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[30] Stephane Meystre. “The current state of telemonitoring: a comment on the literature”. In: *Telemedicine Journal & e-Health* 11.1 (2005), pp. 63–69.

[31] Ricky KP Mok, Edmond WW Chan, and Rocky KC Chang. “Measuring the quality of experience of HTTP video streaming”. In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE. 2011, pp. 485–492.

[32] Onur Mudanyali et al. “Compact, light-weight and cost-effective microscope based on lensless incoherent holography for telemedicine applications”. In: *Lab Chip* 10 (11 2010), pp. 1417–1428.

[33] *NVIDIA 3D System Requirements*. `http://www.nvidia.ca/object/3d-vision-system-requirements.html`. [Online]. 2017.

[34] J-R Ohm et al. “Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC)”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (2012), pp. 1669–1684.

[35] ITU-T RECOMMENDATION P. “Subjective video quality assessment methods for multimedia applications”. In: (1999).

[36] Lamberto Piron et al. “Exercises for paretic upper limb after stroke: a combined virtual-reality and telemedicine approach”. In: *Journal of Rehabilitation Medicine* 41.12 (2009), pp. 1016–1020.

[37] Viorel G Popescu et al. “A virtual-reality-based telerehabilitation system with force feedback”. In: *IEEE Transactions on Information Technology in Biomedicine* 4.1 (2000), pp. 45–51.

[38] Mukund Srinivasan. “VP9 Encoder and Decoders for Next Generation Online Video Platforms and Services”. In: *SMPTE Annual Technical Conference and Exhibition*. SMPTE. 2016, pp. 1–14.

[39] Yaniv Taigman et al. “Deepface: Closing the gap to human-level performance in face verification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1701–1708.

[40] Nikolaos Thomos, Nikolaos V Boulgouris, and Michael G Strintzis. “Optimized transmission of JPEG2000 streams over wireless channels”. In: *IEEE Transactions on Image Processing* 15.1 (2006), pp. 54–67.

[41] Ann Thuring et al. “Remote fetal medicine consultation using video streaming”. In: *Ultrasound in Obstetrics and Gynecology* 24.3 (2004), pp. 318–318.

[42] Paul Viola and Michael J Jones. "Robust real-time face detection". In: *International Journal of Computer Vision* 57.2 (2004), pp. 137–154.

[43] Yubing Wang and Mark Claypool. "RealTracer—Tools for measuring the performance of RealVideo on the Internet". In: *Multimedia Tools and Applications* 27.3 (2005), pp. 411–430.

[44] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: *IEEE Signal Processing Magazine* 26.1 (2009), pp. 98–117.

[45] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment". In: *Conference Record of the 37th Asilomar Conference on Signals, Systems and Computers*. Vol. 2. IEEE. 2003, pp. 1398–1402.

[46] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612.

[47] John Watkinson. *The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4*. Taylor and Francis, 2004.

[48] Andrew B Watson, James Hu, and John F McGowan. "Digital video quality metric based on human vision". In: *Journal of Electronic Imaging* 10.1 (2001), pp. 20–29.

[49] Stefan Winkler and Praveen Mohandas. "The evolution of video quality measurement: From PSNR to hybrid metrics". In: *IEEE Transactions on Broadcasting* 54.3 (2008), pp. 660–668.

[50] Kaipeng Zhang et al. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.

[51] Lin Zhang et al. "FSIM: A feature similarity index for image quality assessment". In: *IEEE Transactions on Image Processing* 20.8 (2011), pp. 2378–2386.