

DEVELOPMENT OF A TELEPRESENCE WHEELCHAIR USING ADVANCED WIRELESS VIDEO STREAMING

By

Van Kha Ly Ha

A thesis submitted to the Faculty of Engineering & Information Technology

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the University of Technology Sydney

Sydney, August 2017

CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as part of the collaborative doctoral degree and/or fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Van Kha Ly Ha

Sydney, August 2017

Acknowledgements

Foremost, I would like to take this opportunity to express my deepest gratitude and appreciation to my principal supervisor, Professor Hung Tan Nguyen, for his enthusiastic supervision, guidance, support and patience throughout the period of my doctoral research.

I would like to thank my co-supervisors, Dr. Nghia Nguyen and Dr. Rifai Chai, for all their help, especially for their valuable guidance and great support from the beginning of my research career at the University of Technology Sydney. I greatly appreciate Ms. Caroline Reed and Mr. John Hazelton for their great assistance in proofreading my papers and thesis.

I would also like to thank all of the staff and student members in the Centre for Health Technologies for their discussions, comments and advice during the helpful mixer CHT seminars, and staff of the University of Technology Sydney for their kind support.

I gratefully acknowledge the 911 VIED-UTS scholarship from the Vietnam Government; without this fund my overseas study dream would not come to reality.

Finally, I sincerely thank all of my friends and my family in Vietnam for their encouragement and strong support throughout my Ph.D. study.

Van Kha Ly Ha

Sydney, August 2017

TABLE OF CONTENTS

List of Figures	viii
List of Tables.	xvi
Abstract.....	xvii
Chapter 1 Introduction.....	1
1.1 Problem Statement	1
1.2 Objectives of the Thesis	4
1.3 Contributions of the Thesis.....	6
1.4 Structure of the Thesis.....	8
1.5 Publications	10
Chapter 2 Literature Review.....	11
2.1 Introduction.....	11
2.2 Intelligent Wheelchairs	12
2.3 Telepresence Systems	19
2.3.1 Telepresence Hardware System.....	19
2.3.2 Telepresence System Software.....	23

2.3.3	Telepresence System Applications.....	24
2.4	Advanced Communication Technologies	27
2.4.1	Wireless Technologies for Robot Control	27
2.4.2	Virtual Reality.....	30
2.4.3	360-Degree Video Communication	31
2.5	Discussion.....	33
Chapter 3	Real-Time Video Streaming for a Telepresence Wheelchair	36
3.1	Introduction.....	36
3.2	System Architecture.....	38
3.3	Wireless Video Streaming.....	41
3.3.1	Wireless Networking.....	41
3.3.2	Video Streaming over Wireless Networks.....	44
3.4	JPEG-Based Video Encoding and Quality Assessment.....	49
3.4.1	JPEG Coding.....	49
3.4.2	Video Quality Assessment	53
3.5	Experiments and Results.....	55
3.5.1	Experiment 1: Images Acquisition.....	55
3.5.2	Experiment 2: Images Streaming.....	59
3.5.3	Results.....	61

3.6	Discussion.....	71
-----	-----------------	----

Chapter 4 A Telepresence Wheelchair Based on Ubiquitous Platforms and Advanced Wireless Mobile Networks 73

4.1	Introduction.....	73
4.2	Streaming Speed Enhancement Based on Ubiquitous Platforms.....	76
4.3	Advanced Wireless Mobile Networks	78
4.4	Telepresence with Multiple Cameras.....	80
4.5	Telepresence Wheelchair and Virtual Reality.....	85
4.6	Experiments and Results.....	89
4.6.1	Experiment 1: Real-time Video Streaming with Multi-Camera for a Telepresence Wheelchair	89
4.6.2	Experiment 2: A Telepresence Wheelchair System Using Cellular Network Infrastructure in Outdoor Environments.....	94
4.6.3	Experiment 3: Telepresence Wheelchair and Virtual Reality.....	99
4.7	Discussion.....	108

Chapter 5 An Advanced Telepresence Wheelchair Based on WebRTC and 360-degree Video Communication..... 110

5.1	Introduction.....	110
5.2	Real-time Communication.....	112
5.2.1	Advanced Video Streaming over WebRTC	112

5.2.2	Prototype System Implementation and Deployment.....	117
5.3	Integration of Emerging Technologies for an Advanced Telepresence Wheelchair.....	120
5.3.1	Telepresence Wheelchair with 360-degree Vision.....	120
5.3.2	Remote Control Telepresence Wheelchair via WebRTC.....	128
5.4	QoE Estimation of Wireless Video Streaming Using Neural Networks.....	128
5.5	Experiments and Results.....	135
5.5.1	Experiment 1: Telepresence Wheelchair based on WebRTC.....	135
5.5.2	Experiment 2: An Advanced Telepresence Wheelchair with Wide Field View Using a Dual-Fisheye Camera	139
5.5.3	Experiment 3: QoE Estimation of Wireless Video Streaming Using Neural Networks.....	142
5.5.4	Experiment 4: Evaluation of Remote Control Telepresence Wheelchair over WebRTC and 360-degree Vision.....	146
5.6	Discussion.....	150
Chapter 6	Conclusion and Future Work	153
6.1	Conclusion	153
6.2	Future Work.....	158

Appendix A Power Wheelchair Components	159
Appendix B Software Code.....	164
Bibliography.....	214

List of Figures

2.1	The typical architecture of an intelligent wheelchair (Nguyen 2013).	13
2.2	Semi-autonomous wheelchair platform, Nagaoka University of Technology, Japan, (Katsura 2006).....	13
2.3	Smart wheelchair, University of Technology Sydney (Nguyen 2013).....	14
2.4	Intelligent wheelchair system (SAM), University of Technology Sydney, Australia, (Nguyen 2012).....	15
2.5	Intelligent wheelchair, Shanghai Jiao Tong University, China (Liu 2015).....	15
2.6	A computer vision based co-robot wheelchair, USA, (Li et al. 2016).....	16
2.7	EEG-based intelligent wheelchair, Korea University, Korea (Kim 2016).....	17
2.8	A BCI-based wheelchair, South China University of Technology, China, (Zhang 2016).....	18
2.9	Brain-machine interface (BMI) wheelchair, (Li 2017).....	18
2.10	MeBot platform of MIT Media Laboratory, (Adalgeirsson 2010).....	19
2.11	Principal components of Avatar robot system, (Martinez-Garcia 2012).....	20
2.12	Telepresence robot, the University of Tsukuba, Japan, (Hasegawa 2013).....	21
2.13	Telepresence Robot, Athabasca University, Canada, (Denojean 2014).....	21
2.14	Telepresence robot, Meijo University, Japan, (Hasegawa 2015).....	22
2.15	Remote patient monitoring telepresence robot, Canada, (Lepage 2016).....	22
2.16	Commercially-available telepresence robots,(Herring 2013)	24
2.17	The Giraff telepresence robot, (Gonzalez 2012).....	26
2.18	The network connection of the Beam telepresence robot, (Ackerman 2014)..	29

2.19	Oculus Rift, (Oculus 2016).....	30
2.20	The panoramic scene, National Chiao Tung University, Taiwan, (Huang 2014).	31
2.21	The panoramic stereo video, International Institute of Information Technology, India, (Aggarwal 2016).....	31
2.22	Driver Assistance System, the University of Texas, USA, (Pan 2016).....	32
3.1	The telepresence wheelchair system topology.....	38
3.2	The detail of the telepresence wheelchair hardware.....	39
3.3	The relation of the IEEE 802.11 standard to the OSI reference model.....	42
3.4	The bandwidth of the IEEE 802.11 standard.....	43
3.5	2.4 GHz band for Wi-Fi channels.....	43
3.6	5 GHz spectrum for Wi-Fi (Intel 2014).....	44
3.7	Video transmission over the wireless channel.....	45
3.8	Block diagram of the JPEG Encoding.....	51
3.9	Block diagram of the JPEG Decoding.....	51
3.10	Framework for video streaming over the wireless network.....	55
3.11	Video stream from Ladybug3.....	56
3.12	The configuration and side views of the Ladybug3 camera.....	57
3.13	Server and client topology.....	59
3.14	System performance test at the Centre for Health Technologies.....	61

3.15	Six images were taken from six camera units of Ladybug3.....	62
3.16	A JPEG panoramic frame generated after processing.....	62
3.17	Bitmap and JPEG file size comparison.....	62
3.18	Packet loss during transmission.....	63
3.19	The original image at the server.....	63
3.20	The histogram of the original image.....	64
3.21	The image received at the client.....	64
3.22	Histogram of the obtained images during the streaming process with MSE = 1.8651e+03 and PSNR = 15.4239 dB.....	64
3.23	An original panoramic image extracted from the wheelchair.....	65
3.24	The image obtained over wireless with errors.....	65
3.25	A sequence of six frames is extracted from the camera attached to the wheelchair.....	66
3.26	The image obtained at client side over wireless with our scheme.....	67
3.27	Images quality measured by mean squared error.....	67
3.28	PSNR measured received image quality.....	68
3.29	Wireless Network spectrums trace view.....	68
3.30	Streaming rates (Bytes/s) vs. Time (s).....	69
3.31	Client receives JPEG images from Server.....	70

4.1	A typical Skype architecture.....	76
4.2	A telepresence wheelchair based on Skype architecture.....	77
4.3	A simplified LTE cellular network architecture.....	78
4.4	The back and front view of the telepresence wheelchair.....	80
4.5	The diagram of a multi-video streaming based on Skype framework.....	81
4.6	The connections of the wheelchair control components.....	83
4.7	The front and the back view of the Oculus Rift and Gear VR.....	85
4.8	A remote user is wearing virtual reality device and control the wheelchair...	86
4.9	System experiments at the University of Technology Sydney.....	89
4.10	The average streaming rate of ten trials in the experiments.....	91
4.11	The average round-trip time of ten trials in the experiments.....	92
4.12	Multi-view video obtained from a telepresence wheelchair and the control interface at the remote site.....	93
4.13	(a) Wheelchair in an outdoor environment at Jones Street – Sydney. (b) A connection from a remote user in Vietnam to the wheelchair user in Sydney.....	95
4.14	Remote user controlled wheelchair from a long distance.....	96
4.15	The RSRP measurements from the wheelchair of experiments.....	96
4.16	Round trip time vs. jitter over a period of 4500 milliseconds of same country links (Australia – Australia).....	97
4.17	Round trip time vs. jitter over a period of 4500 milliseconds of different country connections (Vietnam – Australia).....	98

4.18	CPU usage of wheelchair computer and remote user's computer.....	98
4.19	The diagram of the telepresence wheelchair with virtual reality in wireless connections.....	99
4.20	A remote user was wearing Oculus Rift and controlling the wheelchair in outdoor environments.....	100
4.21	Participant (FS1) wore a virtual reality headset to observe the surrounding of the wheelchair and controlled the wheelchair to the desired locations from another room.....	101
4.22	Ten participants in the experiments.....	102
4.23	The signal strength of the wireless networks.....	103
4.24	A comparison of the latency of different connections.....	103
4.25	The measurement of PSNR of 1000 video frames.....	104
4.26	A remote user (MS2) was wearing Oculus Rift and controlling the wheelchair in indoor environments.....	105
4.27	The trajectories of the telepresence and joystick control wheelchair in the experiments.....	106
4.28	Time spent wheeling comparison between the manual control (Joystick) and remote control using telepresence (Tel).....	107
5.1	The MediaStream API component.....	113
5.2	WebRTC architecture (WebRTC 2017)	114
5.3	WebRTC Protocol Layer (Loreto & Romano 2012).....	115
5.4	WebRTC using STUN, TURN, and Signalling (WebRTC 2017)	115

5.5	The signal of A and B connections (Mozilla 2017)	116
5.6	The WebRTC based telepresence wheelchair design.....	117
5.7	A telepresence wheelchair equipped with a dual-fisheye camera.....	121
5.8	A 360-degree view telepresence using WebRTC architecture.....	121
5.9	An equirectangular projection.....	122
5.10	The block diagram of feature-based panoramic images stitching.....	123
5.11	Remote control telepresence wheelchair based on WebRTC.....	126
5.12	The connection of the control components.....	127
5.13	Neural Network.....	130
5.14	One layer of Neurons.....	130
5.15	Typical transfer functions.....	131
5.16	Multiple layers of Neural Networks.....	131
5.17	QoE Estimation model based on the Neural Network.....	132
5.18	Block diagram of QoE estimation based on Neural Network.....	133
5.19	The user interface and the video obtained by the remote user.....	136
5.20	Bandwidth and bitrate of the system.....	137
5.21	Round-trip time of telepresence based on WebRTC.....	137
5.22	Frame rate per second (fps) of WebRTC vs. Skype.....	138
5.23	System performance test at the Centre for Health Technologies.....	139
5.24	Dual-fisheye video without stitching processing.....	140

5.25	Spherical panoramic video obtained from a telepresence wheelchair and the control interface at the remote site.....	141
5.26	(a) The measurement of wireless network speed, (b) the average processing time per frame, (c) the frame rate and (d) the peak signal noise ratio of 10 experiments.....	141
5.27	The proposed neural network for QoE estimation.....	143
5.28	The performance progress over epochs.....	144
5.29	Regression plots for the training, validation, testing and all data.....	145
5.30	Estimated MOS based on ANN versus measured MOS.....	145
5.31	System performance test at the Centre for Health Technologies, UTS.....	146
5.32	Trajectory of the wheelchair when the remote user controlled the wheelchair with straight ahead directions.....	147
5.33	Test path setup with the figure eight curve.....	147
5.34	The full field of view obtained at the remote user.....	148
5.35	The full top-down view obtained at the remote user.....	148
5.36	The trajectory of the remote control telepresence wheelchair and the desired path.....	149
5.37	The telepresence wheelchair experiments in an outdoor environment.....	149
5.38	The trajectory of the remote control telepresence wheelchair in an outdoor environment at the campus of the University of Technology Sydney and the desired path.....	150

A.1	Wheelchair gear-motors	159
A.2	DX System	159
A.3	Keypad of the wheelchair	160
A.4	Battery charging	160
A.5	Signal Label Application Diagram and pin out	161
A.6	NI USB-6008/6009 Block Diagram.	163
A.7	NI USB-6008/6009 Analog Input Circuitry.	163

List of Tables

3.1	TASO Picture Rating.....	53
3.2	CCIR Five-Grate Scale	53
3.3	Camera Specifications.....	57
3.4	Network settings for experiments.....	60
3.5	Average PSNR Results for Various Distances.....	69
4.1	The Test Positions of the Experiments.....	90
4.2	Command buttons available during experiments.....	91
4.3	The average accuracy of command tasks.....	93
4.4	Details of the computers used for implementation.....	94
4.5	Performance of the System with Ten Participants.....	106
5.1	ITU 5-Point Quality Scale Measurement of MOS.....	133
5.2	Mapping the PSNR, SSIM to the MOS.....	134
5.3	The average performance results of the 30-trials.....	136
A.1	NI USB-6008 and NI USB-6009 Comparison	161
A.2	Signal Descriptions	162

Abstract

In recent years, assistive technologies, especially intelligent wheelchairs have played a crucial role in supporting the elderly and people with disabilities. Current smart wheelchairs are usually controlled by individuals sitting in the wheelchairs. There has been limited research undertaken on the ability to control the mobility of wheelchairs while people are not inside the wheelchairs. Along with intelligent wheelchairs, telepresence robots have been developed for various applications. However, there has been limited work in developing telepresence systems based on wheelchair platforms for assistive technologies. Thus, an effective integration of telepresence functions into a wheelchair would create an innovative health care support solution.

This thesis aims at developing a telepresence wheelchair using an assistive technology which is capable of providing two-way video communication and interaction. The telepresence wheelchair developed in this thesis can support independent mobility of the wheelchair users, especially in the scenarios where people with disabilities might use the telepresence function to move the remote wheelchair towards their positions without the help of caregivers. In addition, the system can allow users to look at the environment surrounding the wheelchair and to control the wheelchair to move around different places to communicate and meet other people. Thus, the designed system would be significantly meaningful and helpful for individuals with mobility impairment. However, designing and developing such an Internet-enabled telepresence wheelchair remain significant technical challenges in terms of implementing in real-time, providing remote control, and realizing a wide field of view.

The key contribution of this thesis is to develop a unique telepresence wheelchair system that provides high-quality video communication with a full field of view and efficient remote control over the wireless Internet in real-time. The current telepresence systems which are coping with one view direction at one time are not flexible and are cumbersome while operating in the reverse direction. To address these issues, this thesis investigates the advanced technologies to develop a telepresence wheelchair with 360-degree vision. To develop a complete telepresence wheelchair, this thesis has exploited technological

advancement in image processing, wireless communication networks, and healthcare systems. There are three core tasks to be considered: Firstly, to investigate the data communication techniques to stream the views surrounding the wheelchair wirelessly from the unknown environment to the remote users to find the appropriate protocol for the real-time application. Secondly, to develop real-time communication and remote interaction with depth information for immersive experiences in telepresence. Thirdly, to implement the 360-degree field of view for a telepresence wheelchair and to develop an independent application based on emerging technologies.

Accordingly, in the first stage, the Internet protocol for transmitting video based on client-server topology over a wireless network has been developed and implemented as the first attempt. The experimental results with the video transmission based on client-server topology over a wireless network showed that the proposed method successfully streamed the *panoramic images* with the average peak signal to noise ratio (PSNR) of 39.19 dB.

In the second stage, the real-time video streaming based on the cross-platform, namely Skype framework has been explored for *multiple-video transmission* to improve the system performance. The experimental results also showed that the streaming rate was between 25 and 30 frames per second (fps) and the round-trip time varies from 3 to 271 milliseconds. In addition, the wheelchair can successfully navigate by remote control at different distances with the average accuracy of 97.72 %.

In the final stage, this thesis has performed the development and implementation of *360-degree video streaming* and remote controlling based on *advanced real time communication*. The experimental results showed that the round-trip time significantly decreased and fluctuated from 3 to 20 ms. Moreover, the proposed system is able to stream *a full field of view video* surrounding the wheelchair smoothly in real-time with the average frame rate of 25.83 fps, and the average PSNR of 29.06 dB.

The experimental results demonstrated that the proposed telepresence wheelchair is able to stream *a full field of view video* surrounding the wheelchair effectively and smoothly in real-time. Furthermore, the results showed the designed telepresence wheelchair could be controlled remotely via the wireless Internet with high accuracy.

Chapter 1

Introduction

1.1 Problem statement

Assistive technologies have played a vital role in supporting the elderly and people with disabilities. According to the global survey on assistive technologies from the World Health Organization, there will be 1.5 billion people with disabilities and the elderly, who will need useful assistive technologies such as hearing aids and wheelchairs by 2030 around the world (WHO 2016). In Australia, the Survey of Disability, Ageing and Carers (SDAC) from the Australian Bureau of Statistics (ABS) in 2015 showed that over 4.3 million people in Australia have some forms of disabilities. The disability rate in Australia has remained relatively stable over time, with about 18.3% and 18.5% of Australia's population in 2015 and 2012, respectively. Moreover, Australia has an aging population. There were approximately 3.5 million older Australians in 2015, and 50.7% of seniors were living with disabilities, representing one in every seven people or 15.1% of the Australia's population (ABS 2016). The information from the survey is valuable in providing evidence-based resources that motivate researchers to develop assistive technologies to meet the needs of seniors and people with disabilities to improve their quality of life in their everyday lives.

Over the last few decades, there have been significant achievements in the assistive technology design and development. Among the assistive technologies, wheelchairs are the most commonly used assistive devices for supporting the personal mobility of disabled people (Ktistakis & Bourbakis 2017). Traditional wheelchairs are typically designed to operate in the manual or electric mode which has been widely used for the elderly and people with disabilities moving around in daily life at home or in hospitals.

Apart from traditional wheelchairs, researchers have developed smart wheelchairs or intelligent wheelchairs to improve the functionalities of the wheelchairs to meet the needs of the disabled people. It is practically desirable to develop smart wheelchairs which can be navigated safely for those who have difficulties or are not able to control the wheelchairs independently. Examples are smart wheelchairs which can avoid surrounding obstacles (Nguyen et al. 2012; Nguyen et al. 2009), and wheelchairs which can be controlled by using brain waves with the brain-computer interface (BCI) (Rifai et al. 2013; Zhang et al. 2016).

While the large proportion of previously existing studies about intelligent wheelchairs have primarily focused on developing a wheelchair controlled by a person sitting on it, there is little attention given to developing a smart wheelchair which supports remote interaction and operation from long distance in a remote location. The communication between the wheelchairs and the remote users provides significant benefits for the disabled people in flexibility, mobility, and independence in healthcare support. Thus, the research on developing intelligent wheelchairs which support the communication interactions with the remote users is of practical interest. However, there has been limited research undertaken on the ability to control the mobility of wheelchairs from a remote location.

In recent years, along with the explosive growth of telecommunications technologies, the field of telepresence has attracted very considerable interest in both academic and industrial fields. Telepresence systems involve a combination of advanced technologies such as mobile robotics, computer, and telecommunication networks which enable users to be virtually present and to interact in a remote location between the remote user and the local participants. The large proportion of telepresence robots were developed through mobile robot platforms (Kristoffersson, Coradeschi & Loutfi 2013). for video and audio communication. Apart from telepresence robots for meetings, telepresence robots for education have also been explored (Oh-Hun et al. 2010). Meanwhile, there has been limited work in telepresence healthcare, such as for remote visiting, for assistance and healthcare purposes (Tatsuno et al. 2010b). Moreover, there has been little work done in developing telepresence systems based on wheelchair platforms for mobility aids and healthcare purposes.

Developing a telepresence wheelchair as assistive technology would be meaningful and would have great benefit for individuals who have mobility challenges, especially the elderly and people with disabilities. The telepresence wheelchair is a mobile wheelchair platform which is capable of providing two-way video communication and interaction over the wireless Internet. The use of telepresence wheelchairs could provide mobility assistance to people with disabilities when distance separates the participants (Ha, Nguyen & Nguyen 2015).

The telepresence wheelchair system would have potential in a variety of healthcare applications and helpfulness for people with disabilities. One of the important applications is to provide support for independent mobility of the wheelchair user, especially in the scenarios when the wheelchair is located at a distance from the people with disabilities. The telepresence wheelchair system allows users to interact and control the mobility of the wheelchair at a long distance. For instance, people with disabilities or the elderly are on the bed or somewhere that is far away from the wheelchair, and they might use the telepresence function to move the wheelchair towards their positions without the help of caretakers. Thus, it is of practical interest to develop a telepresence wheelchair providing assistance for people with disabilities and allowing them to have an independent life.

Moreover, the telepresence wheelchair can provide opportunities for social interaction. The people with disabilities, the elderly, and individuals who have difficulty with mobilities may be restricted to participate in communities in person due to lack of transportation. To aid those populations to reduce these barriers and to facilitate involvement in activities and community engagement, the development of a telepresence wheelchair can allow the disabled people to control the wheelchair to move around in different places to meet people without actually sitting in the wheelchair. The system can enable disabled people to communicate with other disabled people and the community via two-way video communication in real-time. Such a system is also very helpful during the recovery and rehabilitation process. In these contexts, people with disabilities can avoid the feelings of isolation and loneliness and, thus, their quality of life and health issues can be improved.

Furthermore, remote monitoring and telehealth from the wheelchairs are common challenges for people with disabilities. Telepresence wheelchair systems are designed not only to provide assistance to the wheelchair users but also to act as a bridge between the participants. By way of using two-way video communication, the systems may be used for the wheelchair users to interact with remote users such as healthcare professionals, nurses, as well as doctors for healthcare instruction. In addition, telepresence wheelchairs can transmit the wheelchair user's biological signals such as information of heart rate, blood pressure, and body temperature for advanced monitoring in assistive healthcare applications.

It is obvious that individuals with disabilities would greatly benefit from these technologies. However, designing and developing such a telepresence wheelchair to meet the needs of disabled people remains technical challenges of real-time communication, remote control, and wide field of view.

1.2 Objectives of the thesis

The primary objective of this thesis is to develop an advanced telepresence wheelchair system to provide assistive mobility for people with disabilities and older adults. The telepresence wheelchair system is designed to allow the remote users to be present at the wheelchair place and interact with the environment (Ha, Nguyen & Nguyen 2015, 2016b). The aim of this study is to investigate various approaches and different levels of telepresence experiences. The hardware system development and software implementation are explored to identify the suitable and the best affordable approaches. In order to achieve these purposes, the research in this thesis has the following objectives.

- The first objective of the research is to study the integration of technological advancements in assistive technologies, wireless communication networks, and images processing for healthcare applications. The aim is to develop an intelligent wheelchair system in which the images surrounding the wheelchair from the unknown environment are transferred wirelessly to the remote users. The challenges

of video codec and the suitable physical layer of the wireless transmission are also explored for video streaming in real-time.

- The second objective is to develop real-time communication and remote interaction at a long distance. This research focuses on developing a telepresence wheelchair based on *ubiquitous frameworks, mobile wireless data communication infrastructure*, and technologies that are able to stream video and data in real-time. In addition, *multiple-camera* implementation and development strategies are designed to achieve a multi-view telepresence wheelchair system. Moreover, an integration of virtual reality and telepresence wheelchair is conducted to obtain depth information and immersive experiences in telepresence.
- The third objective of this research is to develop an advanced telepresence wheelchair system with *a 360-degree field of view*. A novel approach to emerging information technology, web real-time communication to improve the system performance and develop an independent application is explored. This objective is to investigate the feasibility of 360-degree field of view for a telepresence system. The key idea of this unique telepresence wheelchair is to provide efficient and safe wheelchair navigation.
- Finally, an advanced telepresence wheelchair with a complete telepresence system with a wide field of view and *remote collaborate-control* is designed. Several techniques are involved in the process of completing the design in real-time applying real hardware. The aim is to not only provide a prototype for a telepresence wheelchair for healthcare assistants but also adopt the advancement of emerging information technology in the field of information technology, robotics, and biomedical engineering.

During the system development and implementation in this project, a number of design factors are taken into consideration, including:

- User-friendly and easily accessible: this telepresence wheelchair design is towards people with disabilities. Thus, the design of the user interface must provide easy usage and quick access from anywhere and anytime.
- Real-world and real-time: the telepresence wheelchair is required to interact with the changes in the surrounding environments quickly. Thus, the delay and processing time are important factors.
- High-quality video: The quality of video streaming is required to meet the standards of resolutions and frame rates. The high-quality video makes the users feel more natural in interaction with the remote users.
- Wide Field of view: it is desired that the telepresence wheelchair is capable of covering the whole scene with the wider field of view.

The above-mentioned factors will contribute to the quality of the designed telepresence wheelchairs and will be measured and analyzed in the subsequent chapters of the thesis.

1.3 Contributions of the thesis

This thesis presents a development of a telepresence wheelchair system as an intelligent wheelchair to provide high-quality video communication and efficient remote control over the Internet in real-time. The most significant contribution of this thesis is the development of an Internet-enabled intelligent wheelchair to help people with disabilities. In addition, throughout the system development, implementation and experiments, this study also makes significant contributions to the fields of robotics, computer vision, human-machine interaction, and assistive technology in the area of biomedical engineering. The specific contributions of this thesis are summarized as follows:

- A prototype of a telepresence wheelchair with wireless video streaming is developed. The outcome of this study provides a good understanding of images processing and video transmission over wireless networks in real-time. The design and implementation of the telepresence wheelchair system provide an architecture and implementable framework for the research. Successful implementation of real-time communication from the wheelchair to remote locations has significant meaning for developing the concept of telepresence wheelchairs for remote

assistance in distance support for the people with disabilities. In addition, the evaluation of the system performance and the findings contribute to the wireless streaming, telepresence system, and real-time application development. Further details are given in Chapter 3 and the author's conference paper (Ha, Nguyen & Nguyen 2015).

- A telepresence wheelchair is implemented based on a *ubiquitous platform*, advanced mobile wireless network and multiple cameras. This study demonstrates the feasibility of using such a cross-platform for telepresence systems. The investigation in this study provides theoretical concepts and prototypes for video communication frameworks. The greatest advantage of the proposed approach is to enhance the streaming speed and video quality. The technique explored in this research is also useful and feasible for many mobile applications. In addition, a telepresence wheelchair system with multi-view using the multiple cameras is developed. This work not only provides safe wheelchair navigation but also provides an immersive experience of virtual reality. These detailed contributions are presented in Chapter 4 and have been published in the author's conference papers (Ha, Nguyen & Nguyen 2016a, 2016b).
- An advanced telepresence wheelchair system by using the *web real-time communication* and *dual-fisheye camera* to achieve a full field of view is presented. Using the dual-fisheye camera with a wide field of view allows the remote user to easily control the wheelchair. The result of this research has a significant contribution to the area of telepresence. The demonstration of developing a wide field of vision for a telepresence wheelchair is feasible. Moreover, the resulting real-time data transferring using emerging information technology, namely web real-time communication, also is significant in the field of information technology and telecommunications. In addition, the thesis has developed a *video quality of experience (QoE) estimation model* based on the *neural networks*. The QoE estimation provides the useful information to evaluate the quality of video streaming and to allow the remote users to control the wheelchairs towards reliable wireless signal coverages. The novel approach developed in the thesis is useful and contributes to the area of computer vision, robotics, and human-machine interaction

applications. Further details of these contributions are presented in Chapter 5 and the author's conference papers (Ha, Chai & Nguyen 2017).

1.4 Structure of the thesis

The thesis is divided into six chapters with references and appendices. The chapters in this thesis are organized as follows:

- **Chapter 1** - Introduction
- **Chapter 2** - Literature Review
- **Chapter 3** - Real-Time Video Streaming for a Telepresence Wheelchair Prototype
- **Chapter 4** - A Telepresence Wheelchair Based on Ubiquitous Platform and Advanced Wireless Mobile Networks.
- **Chapter 5** - Development of an Advanced Telepresence Wheelchair Based on WebRTC and 360-degree Video Communication.
- **Chapter 6** Conclusion and future work
- References and appendices are provided at the end of this thesis.

Chapter 1 – This first chapter discusses the motivation of the thesis which covers the problem statement, objective, contributions, and outline of the thesis. The current challenges and the research questions in the area are raised. It also clarifies the specific objectives and provides an overview of the contributions of this doctoral research. Finally, the contents of the chapters are briefly outlined.

Chapter 2 – is a review of the literature. This chapter presents the background research including a review of intelligent wheelchairs, telepresence systems, and related technologies. Firstly, it introduces the various intelligent wheelchair systems which have been introduced by researchers around the world. Secondly, telepresence systems are explored in terms of system platform developments and their applications in the research communities and commercially. Then, reviews of relevant research in emerging technologies which provide a possible integration for a telepresence wheelchair are presented. Last but not least, a discussion about the motivation to develop a telepresence wheelchair to find the gap for this research will be presented.

Chapter 3 - In chapter 3, the introduction begins by considering the approach to the telepresence in the field of telepresence. A prototype to build a real-time video streaming for the telepresence wheelchair using *local wireless networks* is introduced. The system architecture is also described. Then, a prototype for understanding and designing a telepresence wheelchair is presented. Data communication technique and video transmission are applied in order to transmit the video information. Finally, experiments were conducted to assess the system performance. Experimental results and discussion are presented at the end of the chapter to demonstrate the effectiveness of the proposed approach.

Chapter 4 - starts with a brief introduction of the research targets. The next section provides relevant technologies to enhance the telepresence system. Firstly, in order to improve the performance of such a telepresence wheelchair in real-time, the *ubiquitous platform* and *advanced wireless mobile networks* are explored. Secondly, to enhance the field of view of the telepresence wheelchair, multiple cameras are implemented. Then, the virtual reality technologies are integrated into the system for immersive experiences. This chapter also focuses on a number of experiments which were carried out to test the hypotheses. Finally, the results of real-time implementations are presented to provide evidence for the feasibility of the system.

Chapter 5 - presents the development of an advanced telepresence wheelchair based on *WebRTC with the wide field of vision*. After the introduction section, a new approach for real-time communication based on *WebRTC* is introduced. It begins with the emerging information technology which provides real-time communication. The next section presents *the wide field of vision* and development for a telepresence wheelchair. This is followed by the integration of emerging technologies for an advanced telepresence wheelchair. The last section of the chapter shows real-time and real-world experiences to demonstrate the effectiveness of the proposed methods.

Chapter 6 - provides a conclusion to this thesis and, then, sketches the directions for future research.

At the end of the thesis, the bibliography and appendices related to the previous chapters are included for the reader's reference.

1.5 Publications

There are four international conference papers of the Institute of Electrical and Electronics Engineers (IEEE) (ERA Rank A) related to this thesis. Details about these papers are listed as follows:

- **Van Kha Ly Ha**, T. N. Nguyen, and H. T. Nguyen, "Real-time transmission of panoramic images for a telepresence wheelchair," in 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, Italy, 2015, pp. 3565-3568.
- **Van Kha Ly Ha**, T. N. Nguyen, and H. T. Nguyen, "A Telepresence Wheelchair Using Cellular Network Infrastructure in Outdoor Environments," in 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, Florida, USA, 2016, pp. 5352-5355.
- **Van Kha Ly Ha**, T. N. Nguyen, and H. T. Nguyen, "Real-time Video Streaming with Multi-Camera for a Telepresence Wheelchair," in The 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016, Phuket, Thailand, 2016, pp. 1-5.
- **Van Kha Ly Ha**, Rifai Chai, and H. T. Nguyen, "Real-Time WebRTC-Based Design for a Telepresence Wheelchair," in 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Jeju, Korea, 2017, pp. 2676-2679.

Chapter 2

Literature Review

2.1 Introduction

Assistive Technology has offered individuals with various types of disabilities practical solutions to their daily life activities. There has been significant research undertaken in the field of assistive technology over the last few decades. One of the remarkable assistive technologies is in wheelchair development. A wheelchair is an assistive device which can be considered a universal technical aid for all the cases of people with disabilities leading to restricted mobility due to illness, injury or disability. There exists a wide variety of types of wheelchairs to meet different requirements of people who have different levels of difficulties with mobility. The first invention of an electrically powered wheelchair was in 1903 in the US, and the studies on the development of wheelchairs have increased attention since the early 1950s (Woods & Watson 2003).

Year after year, as electric powered wheelchairs have become more popular, intelligent wheelchairs have also been developed to meet the special needs of disabled people. The goal of smart wheelchairs is to provide assistance and improve the independent mobility for people with disabilities. Most of the existing wheelchair studies have focused on developing intelligent wheelchairs from electric power wheelchairs. In such researches, the smart wheelchairs have been considered in autonomous and semi-autonomous control frameworks by upgrading the conventional wheelchairs with the additional components such as sensor systems, computers, and advanced user interfaces for human-machine interactions.

The relevant literature review shows that emerging information technology is becoming increasingly important in modern society. Some of telepresence systems which allow people to communicate with each other from long distance in various ways have been introduced (Kristoffersson, Coradeschi & Loutfi 2013). Recently, researchers have also approached the development of intelligent wheelchairs by integrating advanced technologies and devices for wheelchair navigation aids (Shen et al. 2016).

This chapter will provide the literature review about the existing wheelchair systems with emphasis on the telepresence systems which lays a foundation for the focus of the thesis. The chapter is organized as follows. Firstly, Section 2.2 presents a brief survey of the popular techniques used for intelligent wheelchairs; secondly, Section 2.3 introduces telepresence systems and analyses the systems in terms of hardware, software, and applications. This is followed by a review of relevant research in advanced technologies that are related to our research which is presented in Section 2.4. Finally, a discussion section is given in Section 2.5 at the end of this chapter.

2.2 Intelligent Wheelchairs

The literature shows that different approaches have been developed for smart wheelchairs over the past few decades (Horn 2012). An intelligent wheelchair, also known as a smart wheelchair, can be defined as an electric-powered wheelchair which can operate in different modes such as manual, autonomous or semi-autonomous by modifying with the addition of sensors, user control interfaces, and a central processing unit. These external devices are incorporated into the wheelchairs in order to provide and collect the environmental data, and user commands to assist the wheelchair users more efficiently. The general architecture of a smart wheelchair is illustrated in Figure 2.1.

An early research of an intelligent wheelchair based on an autonomous robot was introduced in (Bourhis et al. 1993). The authors in (Bourhis et al. 1993) improved the control of the powered wheelchair by proposing a hybrid software and hardware to indicate an obstacle and stop the mobility of the wheelchair in the case of emergency. However, the system was very complex. The applicability of the scheme was restricted since it only worked on a flat surface and short distance.

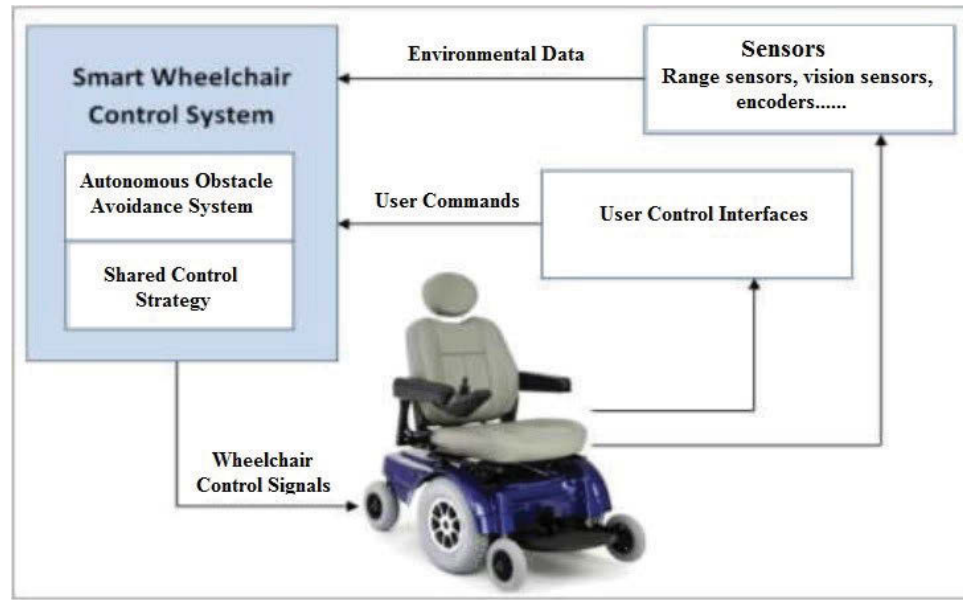


Figure 2.1: The typical architecture of an intelligent wheelchair (Nguyen 2013).

Since then, there have been many different approaches which have been developed for smart wheelchairs with similar objectives. A research group from Nagaoka University of Technology (Katsura & Ohnishi 2006) developed an intelligent wheelchair robot for adaptation to an unknown environment. The study presented a semi-autonomous wheelchair by developing a robot adapted to the environment to avoid obstacles. Also, position control and force control are integrated into the acceleration dimension, and the installation of a semi-autonomous operation method to improve human operation was carried out as illustrated in Figure 2.2.

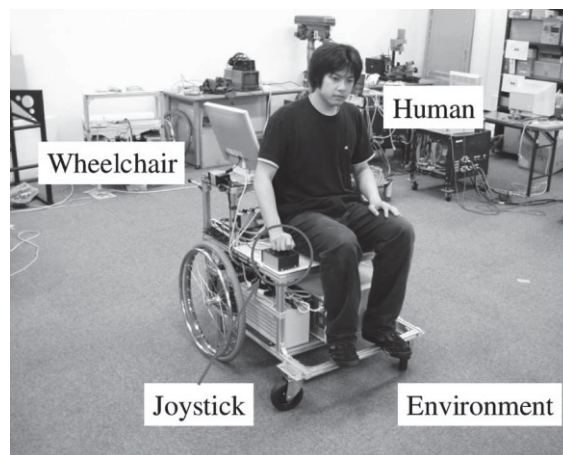


Figure 2.2: Semi-autonomous wheelchair platform, Nagaoka University of Technology, Japan, (Katsura 2006).

A series of smart wheelchair systems also were explored by the University of Technology Sydney (UTS). Researchers from the Centre for Health Technologies have developed intelligent wheelchairs using various approaches. Firstly, an intelligent wheelchair which dealt with human interaction based on head movement was introduced in (Nguyen et al. 2004) by using artificial intelligence and neural networks. Secondly, Nguyen, Su & Nguyen (2010) developed a smart wheelchair based on a power wheelchair equipped with a spherical camera to allow detection of surrounding objects during the navigation of a power wheelchair. Notably, a Point Grey Research Ladybug2 spherical vision camera system was attached to the power wheelchair for surrounding view as shown in Figure 2.3. Thirdly, another intelligent wheelchair, namely SAM (Semi-Autonomous Machine) was developed by Nguyen, Su & Nguyen (2011). In this work, a semi-autonomous wheelchair was designed by attaching some additional components onto a commercial powered wheelchair as illustrated in Figure 2.4. Moreover, (Nguyen et al. 2012) and (Nguyen, Su & Nguyen 2013) also presented a method to provide obstacle avoidance for the system. The combination of stereoscopic cameras and spherical vision cameras is implemented to detect surrounding obstacles (Nguyen, Su & Nguyen 2013).



Figure 2.3: Smart wheelchair, University of Technology Sydney, (Nguyen 2013).



Figure 2.4: Intelligent wheelchair system (SAM), University of Technology Sydney, Australia, (Nguyen 2012).

Similarly, research in (Liu, Chen & Wang 2015) presented a study on mapping and localization of an intelligent wheelchair in large and dynamic environments. The wheelchair was equipped with an onboard computer, a laser SICK LMS111, touchscreen, microphone and encoder-based odometer. The prototype of the wheelchair is presented in Figure 2.5. The wheelchair could provide effective localization in dynamic environments.



Figure 2.5: Intelligent wheelchair, Shanghai Jiao Tong University, China (Liu 2015).

Another recent research group from the Stevens Institute of Technology, Hoboken, USA also designed a smart wheelchair in different approaches to improve the quality of life for the elderly. The authors proposed an egocentric computer vision based on a power wheelchair (Li et al. 2016). The egocentric computer vision based control was developed by accessing the motion direction of the camera and collaborating with the robotic wheelchair by conveying the motion commands with head movements. As a result, the wheelchair could enable autonomous navigation towards a detected person and could search the correct direction for the target object from the camera as illustrated in Figure 2.6.

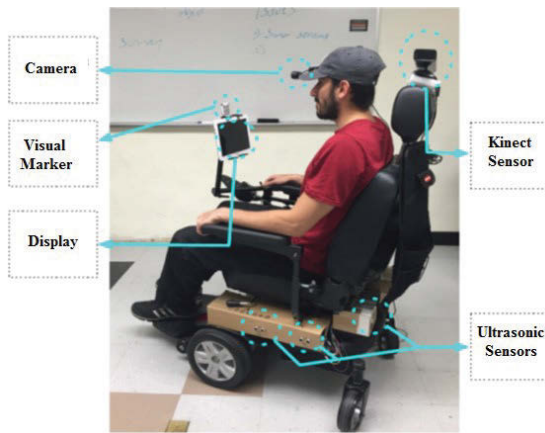


Figure 2.6: A computer vision based co-robot wheelchair, USA, (Li et al. 2016).

On the other hand, there have also been a significant number of studies aimed at integrating human inputs or physiological signals to a powered chair platform to propose an intelligent wheelchair. Researchers have been focusing on human-machine cooperation technology to help a user drive the wheelchair and the brain-computer interface (BCI) wheelchairs have attracted much attention from investigators. Several different methods have been approached to control the wheelchairs using electroencephalography (EEG). Typical examples of such studies are the approaches in (Craig, Nguyen & Burchey 2006), (Iturrate et al. 2009), and (Chai et al. 2014). In (Chai et al. 2014) the authors presented a brain-computer interface classifier for wheelchair commands using the neural network with fuzzy particle swarm optimization which used only two EEG channels classification to control the wheelchair. The researchers have demonstrated how to control a wheelchair using a person's thoughts. The system enables a wheelchair to turn left or right or to move forward simply by thinking the commands.

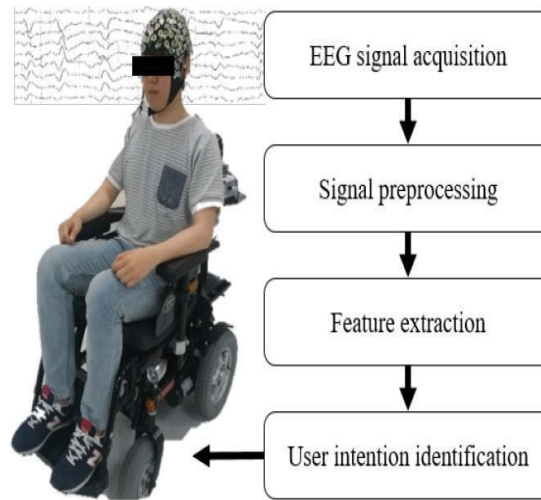


Figure 2.7: EEG-based intelligent wheelchair, Korea University, Korea (Kim 2016).

Similarly, a research group from the Department of Brain and Cognitive Engineering, Korea University, Seoul, Korea designed an EEG-based intelligent wheelchair (Kim & Lee 2016). In this study, the authors used a steady-state somatosensory evoked potential (SSSEP) paradigm, which elicits brain responses to vibrotactile stimulation of specific frequencies, for a user's intention identification to driving a wheelchair. The result of this work confirmed the effectiveness of an SSSEP-based wheelchair driving system (corresponding to turn left, turn right, and go forward). The framework of the EEG-based intelligent wheelchair system is illustrated in Figure 2.7.

In addition to BCI wheelchair approaches, another group of researchers from the South China University of Technology, developed a smart wheelchair combining a P300-based BCI and an autonomous navigation (Zhang et al. 2017). This study reported that the smart wheelchair system was developed by using a Kinect sensor for obstacle avoidance and using ERP-based BCI to adjust the direction of the wheelchair as illustrated in Figure 2.8. The wheelchair was also equipped with a proportional-integral-derivative (PID) controller and a laser range finder, two encoders. The laser range finder is used to determine the distances from the wheelchair to objects in a room. The proportional-integral-derivative controller is used to drive the wheelchair along a planned path from its initial position to a selected destination.



Figure 2.8: A BCI-based wheelchair, South China University of Technology, China, (Zhang 2016).

More recently, a human cooperative wheelchair with brain-machine interaction based on shared control strategy has also been presented in a recent study (Li et al. 2017). In this study, the authors proposed a human-machine shared control strategy for wheelchair navigation control, in both brain-machine control and autonomous control mode. The study demonstrated four direction control signals in BCI operation, and a mapping technique to guide the wheelchair movement among the obstacles. The design of the wheelchair is as shown in Figure 2.9.



Figure 2.9: Brain-machine interface (BMI) wheelchair, (Li 2017).

2.3 Telepresence Systems

Telepresence refers to the systems which provide geographically separated users a sense of being present in the same location. In particular, telepresence robots are systems having the capability of video conferencing developed from a set of technologies which allow the participants to feel as if they are present at a distant location by using telerobotics (Ha, Nguyen & Nguyen 2015). There exist many telepresence systems which have been designed based on different hardware platforms and software development schemes.

2.3.1 Telepresence System Hardware

The early concept and prototype of telepresence robots were presented in (Schultz, Nakajima & Nomura 1991), where the authors discussed the first design, construction, and interface for interactions. However, the field of telepresence has gained a remarkable interest from both academic and industrial areas since 2010. Literature has shown a range of different approaches which have been developed for telepresence robots over the past few years. Much research about the telepresence has started to deal with the issues of the interaction between humans in remote areas.

Research in (Adalgeirsson & Breazeal 2010) developed a Mebot telepresence robot. It introduced the design and evaluation of a telepresence robot for social expression by audio or video communication. A design of the MeBot platform is shown in Figure 2.10. The system simply was a robot which was equipped with a Nokia N810 device with one integrated camera, two arms for shaking, and a display screen played a role as a face of the robot and control interfaces.



Figure 2.10: MeBot platform of MIT Media Laboratory, (Adalgeirsson 2010).

Similarly, an Avatar telepresence robot was designed by Martinez-Garcia, Gallegos & Jaichandar (2012). The Avatar was developed based on a robotics platform as shown in Figure 2.11. For hardware design, the authors developed an on-board computer with an i86 processor and an FPGA board for sensor data processing. A 32bit system controller board with multiple input and output ports was also designed for the motors. The robot also has two cameras in the head of the avatar, and these pairs of images will be transmitted in real-time by using a wireless connection with a Wi-Fi device connected to the computer.

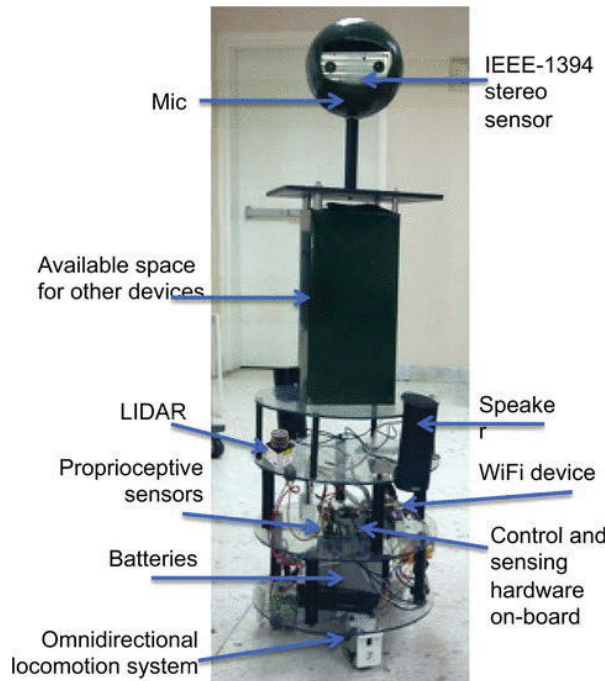


Figure 2.11: Principal components of Avatar robot system, (Martinez-Garcia 2012).

Another telepresence robot was developed by the University of Tsukuba in Japan (Hasegawa & Nakauchi 2013). The authors proposed a telepresence robot conveying pre-motions by using Kinect for gesture perception and a humanoid robot for conveying unconscious gestures. In their study, the system was designed with four arms which enabled various kinds of hand gestures motions. For expressing head motions and looking directions, it had three necks, looking directions, body directions, and facial expressions. A humanoid robot by Kondo Kagaku used a robot platform and servomotors. The display showed the operator's face video so that it conveyed the facial expressions, as illustrated in Figure 2.12.

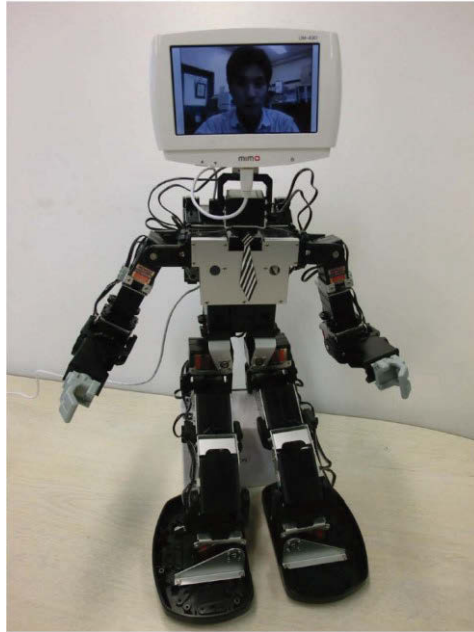


Figure 2.12: Telepresence robot, the University of Tsukuba, Japan, (Hasegawa 2013).

Also, a study in (Denojean-Mairet et al. 2014b) presented an approach to building an affordable telepresence robot based on a ubiquitous computing platform as shown in Figure 2.13. This paper mainly focused on the design and applications. The authors developed a telepresence robot that would be tailored to their research and implementation by adding two 12-volts geared motors paired and a rear caster wheel. An Android mobile device and a Raspberry Pi were implemented to control the robot. A Wi-Fi device was plugged in the Raspberry Pi to connect to the network. The system was also composed of an LCD, a webcam, and a microphone for video conferencing.



Figure 2.13: Telepresence Robot, Athabasca University, Canada, (Denojean 2014).

Furthermore, in a similar study (Tanaka et al. 2015), a group of researchers from the Meijo University from Japan developed a telepresence system that presents similarities to the hardware described in (Hasegawa & Nakauchi 2013). The authors developed the telepresence system by attaching a laser range finder that provides an obstacle avoidance function for safety movement. The telepresence robot was designed for families with older adults living in remote areas, where the robot is independently controlled. The telepresence robot's hardware configuration is shown in Figure 2.14.



Figure 2.14: Telepresence robot, Meijo University, Japan, (Hasegawa 2015).

More recently, a remote patient monitoring telepresence robot was presented by a group of researchers from the Natural Sciences and Engineering Research Council of Canada (Lepage et al. 2016). The system was developed by integrating the video conferencing and a robot platform for remote patient monitoring and visiting. A telecommunication framework was explored to provide bi-directional video and audio communication. This robot was equipped with an 8-microphone array, a camera, a Kinect sensor, a laser range finder, an expressive face, and two compliant arms with four degrees of freedom as illustrated in Figure 2.15.

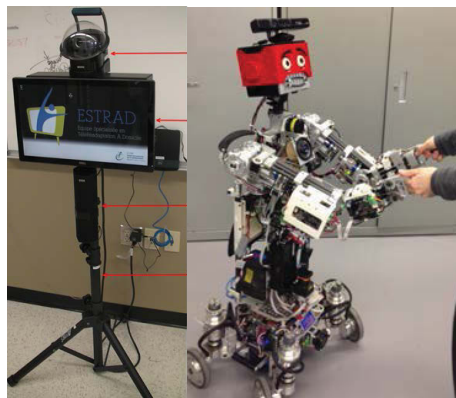


Figure 2.15: Remote patient monitoring telepresence robot, Canada, (Lepage 2016).

2.3.2 Telepresence System Software

Many successful telepresence robots with various designs can be found in the literature. Telepresence has also been an intensive field of research. However, it should be noted that the telepresence robot systems were developed in many different prototypes and software. (Peter Corke 2012), for example, has developed a telepresence robot by integrating a third party application for remote control and video conferencing. An App2app (application to application) was designed for robotic applications by using the Skype communication channel to connect with remote users and clients. The sender sends the control signals to a receiver over Skype channels, and the authors developed App2App.

On the other hand, the TRiCmini telepresence robot for interpersonal communication with older adults was developed by integrating with social network services such as Google Talk to provide a wide range of communication and information sharing easily and conveniently (Chen, Lu & Hsu 2013). The robot control software is an Android App on tablets. The robot control signals received by the tablet connected to the robotic vehicle via Bluetooth. Also, 3G and Wi-Fi were used for wireless communication and data transmission. Two-way audio and video communication are achieved by using social network Google Talk (Chen, Lu & Hsu 2013). The robot was controlled by text messages which were sent and received from Google Talk. Moreover, there are other possible protocols and software designed for the robot. Typical examples are remote control telepresence robots from the remote client by developing the video and data communication based on Web Real-time communication and Google App Engine (Pavón-Pulido et al. 2015).

More recently, an industrial telepresence robot, also known as Beam RPS (Remote Presence System) was available on the market. The Beam RPS system consisted of three components, including the remote presence device, the beam docking station for charging and the client software (Suitable 2014). Inside the Beam, the mobile telepresence robot runs Ubuntu-based Linux OS, while video/audio communication runs over Skype (Calin 2014). To control the Beam from a distance, the Beam is connected to the Internet via Wi-Fi connection. A user-friendly application programming interface was also developed for the best two-way communication with a reliable Internet connection.

2.3.3 Telepresence System Applications

In recent years, there has been significant development in the field of telepresence. The telepresence systems have been developed in many different forms, and the applications of telepresence robots have changed dramatically. A typical application of telepresence is a video conference. However, different from a traditional video conference, the telepresence system is developed based on a robot platform with the technical advancements to enable the remote mobile control. Telepresence may be used as a tool for the meeting, education, and healthcare. Some other types of telepresence applications are under-explored.

According to (Kristoffersson, Coradeschi & Loutfi 2013), there are a variety of uses for telepresence technology in the business domain such as meetings and related works. Telepresence provides significant benefits to organizations since it offers the ability to hold professional meetings across quality communication channels, reduces traveling and improves the effectiveness of distance working. Recently, there has been a considerable increase in companies designing telepresence robots for meetings as illustrated in Figure 2.16.



Figure 2.16: Commercially-available telepresence robots, (Herring 2013).

While many of the above-mentioned systems have been developed for meetings, some other researchers have also explored the use of telepresence for education. For example, the idea of telepresence robot long distance learning was presented in (Oh-Hun et al. 2010) and (Tanaka et al. 2013). The projects explored the use of child-operated telepresence robot systems for the purpose of education. A remote controlled robot that provided video conferencing enabled children to interact with teachers. These works provide an efficient way to learn English for children in Japan from Australia. Also, telepresence could be utilized in education in different ways. In (Cain, Bell & Cheng 2016), the authors used 12 robotic telepresence devices to build collaborative learning experiences for face-to-face and online participants. It could enable long-distance education with real-time video communication and allow students to study from home.

Healthcare and the medical industry are other fields where telepresence is now becoming very prominent. Some telepresence systems developed for the elderly and healthcare support have also been studied. There are many different forms and terms used to healthcare applications, such as remote visiting people, telemedicine, and telehealth. An example of remote visiting people was reported in (Tatsuno et al. 2010a). The authors developed a remote visitor robot. This study aimed at developing a robot that allows the user to join a conference from a remote place through the Internet in order to visit people. Telepresence for healthcare helps connect people across regional or global locations.

However, remote visiting and social communication for older adults are still ongoing research. A telepresence robot for visiting elderly people was developed by Gonzalez-Jimenez, Galindo & Ruiz-Sarmiento (2012). It opens a new channel for friends, relatives, and caretakers to communicate from a distance to assist living of the patients, elders, and people with disabilities. The authors presented the robot mobility and teleoperation named Giraff for the visitors as shown in Figure 2.17. Giraff is a mobile robotic base equipped with a video conferencing set, including web camera, microphone, speakers, and screen. Giraff can be a useful tool to enhance social interaction suited to particular groups of users such as the elderly. For example, the interactions between the remote embodied person and the locally embodied person by the social telepresence robot system were presented by Coradeschi et al. (2011).



Figure 2.17: The Giraff telepresence robot, (Gonzalez 2012).

Furthermore, telepresence robots can be used for assisting patients. Telepresence robots can become a beneficial tool in patient assistance since such robots can help the patient or disabled people interact with the surrounding environment, therefore supporting independent living. For example, Lyons & Joshi (2013) demonstrated the use of a brain-computer interface to control a mobile robot in a remote location. In Taiwan, Sebastian, Jun-Ming & Yen-Liang (2013) have been developing a TRIC (Telepresence Robot for Interpersonal Communication). In this study, a telepresence robot can be applied to dementia care. Through the use of a touch screen, the audiovisual experience that can be delivered to families remotely, physicians and patients are connected.

In the study by Leeb et al. (2015), a BCI telepresence robot for people with severe motor disabilities was introduced. This paper presented a method using brain-computer interfaces to control a telepresence robot to interact with real devices in the real world. Telepresence robot was equipped with an infrared sensor system for obstacle detection and a Skype connection on top for video communication.

Other examples are the studies of Lepage et al. (2016) and Borvorntanajanya et al. (2016), in which a remote patient monitoring telepresence robot was explored for assistance in daily living activities from a distant location. There is also the concept of using telepresence robots as healthcare support tools, for example, to allow the doctor to talk and examine patients from a distance, or from hospital to assess patients in real-time remotely. It allows patients to speak to doctors from thousands of miles away rolled out to seven hospitals without having to be physically present.

2.4 Advanced Communication Technologies

2.4.1 Wireless Technologies for Robot Control

In recent years, there are rapid developments of wireless communication networks. The wireless communication revolution and its emerging applications are becoming more and more sophisticated. In this section, we review the state-of-the-art wireless networking and highlight several promising uses of wireless technology for robot and wheelchair control. However, the details of these wireless networks were not considered in this thesis.

With the advancement of wireless communication technology, many mobile robot platforms used wireless technology to communicate with computing resources, human machine interfaces or others robots. Many mobile robots have been equipped with wireless technology such as personal communication networks, ZigBee, Bluetooth, Wireless Local Area Network, and wireless mobile cellular networks.

According to Mulla et al. (2015), although there are various wireless communication standards for establishing reliable, real-time monitoring and controlling, each technology has advantages and disadvantages. For example, Zigbee provides low complexity, mobility, and an easy implementation and configuration with low-cost. However, it was found that interference lays constraints on Zigbee. On the other hand, wireless local area network (WLAN) allows multiple users to access the service without interfering with the existing technologies. However, the widespread coverage area is a challenging task in WLAN. In addition, the wireless mobile cellular networks, such as 4G and 5G standards have been improving their data rates and coverage to meet real-time communication.

An intelligent wheelchair based on Zigbee wireless sensor networks was introduced by Yi, Yu-chao & Yuan (2010). In this study, the authors introduced a design of sensor nodes, network construction, and communication in the wireless sensor network for remote monitoring of a wheelchair. However, the data transfer rate was 250 kbps indoors with a short distance of 10 m (Yi, Yu-chao & Yuan 2010). In addition, a similar research in (N, Rosman & Sarmawi 2011) presented a design and analysis of a wireless control panel using an RF module and microcontroller PIC16F877 for a robotic wheelchair. The results showed

that the wheelchair could be controlled by using Zigbee wireless communication. However, the RF module had the limitations of one-way data transfer and in a short distance.

Another wireless technology usage has been developed for a teleoperation robotic wheelchair as in (Wu, Jen & Huang 2011). This project proposed a radio based robot localization using a received wireless signal from WLAN. These WLAN signals are responsible for providing an accurate positioning. For the robot navigation, the authors designed a visual feature based path planning, which constructs the topology and linkages of the task environment map.

Gao et al. (2016) also presented a development of a robot communication system. In this evolution, two Wi-Fi models were introduced to assist the navigation of relay robots. The mobile robots with Wi-Fi routers can be optimally and automatically deployed to build and maintain a wireless connection between remote clients and the base station. The signal distribution of the base station is modelled based on a Gaussian process and is used to find optimal locations for relay robots.

On the other hand, Gonzalez et al. (2012) proposed a remote control helicopter system based on the cellular data network for video streaming. The system modules contain a MoFi 35600-3G router for data transfer. The system enabled the downloading of still photos from the helicopter to the ground station. To that end, the smartphone was embedded on the helicopter board.

Other examples as in (Lian, Zhang & Jiang 2012) presented the architecture of the remote control system oriented to 4G networks. By connecting terminals through 4G networks, and using a server, a system can send information and receive commands from the server. Furthermore, (Tsui et al. 2013) designed and developed two generation of semi-autonomous social telepresence robots. A VGO telepresence robot can be connected by using mobile cellular network 4G LTE. It also has onboard 802.11 Wi-Fi for wireless connections to allow the local user to choose what type of internet connection should be used.

Note that, most of the telepresence robots are controlled via wireless networks or the fourth generation cellular network through the Internet due to the video streaming and communication in real-time. However, the system architectures have been approached in different ways. Some of the telepresence robots can be remote controlled and video conference using direct Internet connection. Others are developed based on their network. For example, a telepresence robot which was developed by the Athabasca University in Canada (Denojean-Mairet et al. 2014a) considered using a direct Wi-Fi connection to the Internet based on a ubiquitous computing platform and existing network.

In contrast, Ackerman (2014) introduced a commercial telepresence robot, Beam, in which remote control and two-way video communication were based on their network. The key modifications to the robot involved adding a web-based user interface, a smart tablet to enable two-way audio/video calls via Skype application, and an external speaker to amplify the sound output. For security reasons, the connection must allow outgoing User Datagram Protocol and return traffic on a small range of network ports. The network ports were defined from 6868 to 6871 in both directions through the system firewall as illustrated in Figure 2.18.

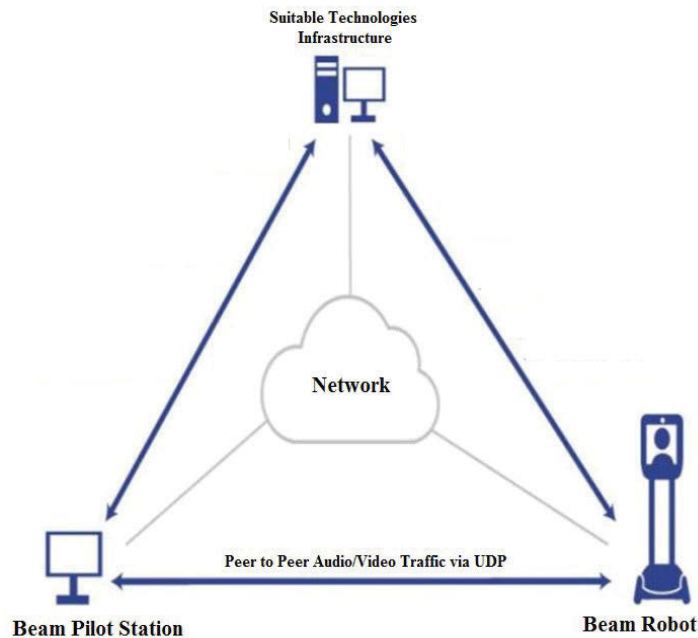


Figure 2.18: The network connection of the Beam telepresence robot, (Ackerman 2014).

2.4.2 Virtual Reality

Virtual reality (VR) technology has revolutionized the way people interact and communicate. One of the most remarkable aspects of virtual reality is that it provides the user an experience of being present in that environment and feels realistic. VR headset allows a user to step into the immersive environment and look in any direction for interaction (Bolas et al. 2013; Mathur 2015; Sharma, Rajeev & Devearux 2015). To date, several companies are developing a virtual reality headset. The most notable virtual reality devices are the Oculus Rift headset (Oculus 2016) and Samsung Gear VR (Samsung 2016). In this thesis, we shall use Oculus Rift headset in Figure 2.19 as a test platform.



Figure 2.19: Oculus Rift, (Oculus 2016).

The explosive growth of virtual reality (VR) technology has attracted considerable attention from both academia and industry. Apart from the development of virtual reality for gaming, several researchers have attempted to develop methodologies for virtual interaction applications. For instance, the studies of immersive virtual tours for campus safety (Sharma, Rajeev & Devearux 2015), and the low-cost open virtual reality devices have also been introduced by (2013).

Interestingly, there has been an increased interest in implementing virtual reality in the field of biomedical engineering (Mathur 2015). According to (Mathur 2015), there is some utility of virtual reality in rehabilitation, treatment, medical training. An example of a VR application for medical education was presented. The study used virtual reality devices such as the Oculus Rift and Razer Hydra for an immersive training environment, including hand interactivity.

2.4.3 360-Degree Video Communication

Many types of research on a wide field of view image acquisition have been carried out on different applications. The concept of a 360-degree video system design was introduced in (Huang et al. 2014). The authors provided a new blending method in video stitching. The results revealed that the system could stitch 30 fps in real-time as shown in Figure 2.20. Moreover, Aggarwal, Vohra & Namboodiri (2016) presented a practical solution to generate 360-degree stereo panoramic video using a single camera by capturing all the light rays required for stereo panoramas in a single frame as shown in Figure 2.21.



Figure 2.20: The panoramic scene, National Chiao Tung University, Taiwan (Huang 2014).



Figure 2.21: The panoramic stereo video, International Institute of Information Technology, India, (Aggarwal 2016).

The Iwate University, Japan introduced an idea of using the full-view spherical image camera for monitoring the whole surrounding environment of the vehicle in (Shigang 2006). In that project, the author introduced a prototype of the spherical image sensor to capture spherical images. The author showed that full field of view is also necessary for real-time tasks of mobile robots, vehicles, and immersive virtual environment construction. For example, providing real-time visual information to car drivers is very helpful for safe car driving and autonomous intelligent vehicles.

Researchers are also working to create a panoramic 360-degree view of the vehicle's environment by merging different images from sides, rear and front of the car using multiple cameras. For example, Pan, Appia & Bovik (2016) considered using a wide field of view camera for parking assistance as illustrated in Figure 2.22. In this study, the researchers used four cameras to capture images from the front, side, rear and top-view images. Fisheye images are captured by four cameras placed with orthogonal principal axes on the front, sides, and rear of a vehicle. As a result, the cameras can capture the entire surrounding 360-degree view, and the captured images can be stitched together to make an image as if it had been taken from a virtual camera placed overhead.

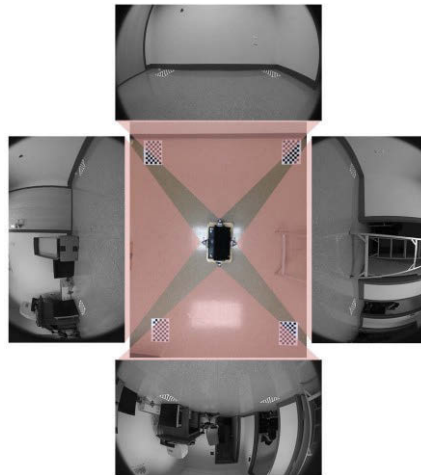


Figure 2.22: Driver Assistance System, the University of Texas, USA, (Pan 2016).

Recent technological developments in Electronics and Signal Processing led to the development of the Spherical camera. 360-degree videos have gained the attention of the research community (Argyriou et al. 2016). For example, Kim, Campos & Hilton (2016) proposed a pipeline for 3D room layout estimation by using a spherical stereo image pair. A spherical stereo alignment algorithm was proposed to align two spherical images. Finally, a simplified room design was reconstructed.

The idea of safe and smart vehicles with 360-degree vision has been thoroughly researched over the past several years to ensure the safety of vehicles movement from possibly dangerous situations. However, 360-degree vision for such system remains challenging due to the complexity of stitching and real-time processing.

2.5 Discussion

This chapter has provided an overview of the development of intelligent wheelchairs, telepresence robots, and relevant technologies used for mobile robot control and robot vision. It has also reported a comprehensive literature review of the concepts and applications of telepresence systems. The literature review demonstrates very considerable potential applications of the telepresence systems in daily life activities, especially in the field of healthcare applications. The literature review ascertains the current research trends into the telepresence systems, which provide research motivation, direction, and development of the assistive technology.

In the field of telepresence, a range of telepresence robots has been reviewed. Although various telepresence robot schemes have been previously investigated, a standardized scheme for hardware and software designs of telepresence systems has not been introduced. The literature review presented in this chapter also shows that the video communication techniques developed for these systems are typically based on existing video conference applications in which traditional cameras or webcams were installed for video streaming. In such, these systems had a limited field of view. There has been little work in the area of telepresence focusing on producing immersive video experiences. Although the user perception of the remote environment around the telepresence robots is critical for the remote users to navigate the robots safely, there has not been any investigation of telepresence systems dealing with a wide field of view of a real-world environment.

Regarding intelligent wheelchairs, it can be seen that there is a wide variety of smart wheelchairs that have been developed to assist people with disabilities. The majority of smart wheelchairs were based on powered wheelchairs in which the additional devices such as a computer, sensors of perception and a control device are attached. Such designs have achieved certain degrees of success in proving the functionalities of the powered wheelchairs. However, almost all of the intelligent wheelchairs have the same typical application contexts in which the users are sitting in the wheelchairs. There has been little research about the flexibility of controlling the mobility of wheelchairs while wheelchair

users are far from the wheelchairs and in the contexts that the wheelchair users want to drive the wheelchairs towards their positions without the availability of carers.

It is also important to highlight that the majority of the telepresence systems have been developed from robot platforms, while research on telepresence systems based on wheelchair platforms is still limited. In addition, a comprehensive benefit study on telepresence as an assistive system for disabled people has not been much discussed in the literature. Individuals with disabilities would benefit from the telepresence technology. Although there are considerable benefits of telepresence applications for people with disabilities, telepresence wheelchairs have not been designed to meet the special needs of people with disabilities. In spite of the fact that smart wheelchair systems exist, it is necessary to develop a next-generation of smart wheelchairs, namely a telepresence wheelchair to fill the gap between telepresence technologies and assistive technologies.

A telepresence wheelchair is an assistive technology which plays a vital role in supporting people with physical disabilities to improve their independence, mobility, and quality of life. An efficient integration of technology into a wheelchair to develop a telepresence wheelchair would offer a variety of healthcare applications. Exploring telepresence wheelchairs will reduce the need for support services, increase the accessibility of assistive technologies, and potentially provide significant benefits for many people with disabilities and carers. Moreover, the use of telepresence wheelchairs, including with video conferencing and remote monitoring allows the individuals who have difficulty with mobility to be able to interact with the wheelchairs remotely without carers. Furthermore, this assistive technology would be helpful and meaningful for people with disability to be more independent.

Different from the existing telepresence systems mentioned above, which are mostly based on robot platforms and a single camera, the telepresence wheelchair designed in this thesis will take into consideration to the production of a 360-degree vision telepresence wheelchair that can execute in real-time via wireless communications. Our telepresence wheelchair is designed with a dual-fisheye camera for 360-degree vision. Our research proposes a notable improvement in a novel telepresence wheelchair which is integrated

with a wheelchair platform and immersive 360-degree view of its surroundings to create solutions for improving healthcare for those who have difficulty in mobility through currently available technologies. Our Centre for Health Technologies, the University of Technology Sydney has developed a standardized prototype for intelligent wheelchairs which can be further improved in this project to design and implement a telepresence wheelchair with appropriate modifications to extend the flexibility in controlling the wheelchair from a distance.

The ultimate goal of this thesis is to develop an assistive system for the people with disabilities, namely, a telepresence wheelchair prototype which can provide the wide field of view of the remote environment to enhance the user perception, to allow the remote users to interact with the telepresence system efficiently in real-time. To improve the visual perception of the remote environment, we have investigated integrating an array of cameras or a dual-fisheye camera into the telepresence wheelchair. Then, the image processing techniques including image compression and image stitching algorithms have been exploited to improve the quality of video and visual perception. To meet the various needs of the disabled people in moving in the indoor or outdoor environments, we have experimented with the telepresence wheelchair on different network frameworks such as local Wi-Fi networks, global Internet networks, and wireless cellular networks. The research has conducted extensive experiments on the designed telepresence wheelchair in practice to evaluate the system performance.

Chapter 3

Real-Time Video Streaming for a Telepresence Wheelchair

3.1 Introduction

As discussed in Chapter 2, a significant number of telepresence robots and intelligent wheelchair systems have been developed around the world. Also, various approaches and applications have also been presented for such research fields in the literature review. It is obvious that the video quality and interactive communications are among the key parameters to measure the performance of the telepresence robots. These performance indicators rely highly on the video encoding and data communication techniques. In this chapter, we shall introduce our designed telepresence wheelchair framework with the aim to investigate the feasibility of the telepresence wheelchair with panoramic images streaming over the wireless networks. This chapter will present the parameters of the data communications which can provide support for video streaming. To make the video data transmission feasible, image compression techniques must be used. We will describe how to adopt the JPEG compression scheme to the telepresence wheelchair.

This chapter is concerned with real-time video streaming for a telepresence wheelchair. The panoramic JPEG images from a spherical camera system are considered for video transmission over the wireless network. Most of the existing telepresence systems have not investigated full view display. The existing telepresence systems typically use one camera or webcam to capture a single image and, thus, remote users have to manipulate the view direction. Recently, exploiting a panoramic camera for an immersive panorama TV service system was introduced in (Dongmahn et al. 2012). It is clear that the applications of the panoramic field of view model could have the potential for developing a telepresence

wheelchair to provide the view surrounding the wheelchair from the unknown environment to assist the wheelchair user. The potential benefits of the panoramic field of view in the telepresence wheelchair applications provide the motivation to find effective communication methods and protocols to obtain the panoramic field of view in real-time.

Although there exist various types of traditional cameras used for live video streaming using wireline or wireless over the Internet, there is limited research about streaming panoramic video from a spherical camera in real-time. It should be emphasized that unlike offline video streaming, interactions in telepresence wheelchairs require real-time video communication. This means that the system must be able to process to capture, encode, transmit, receive, decode, display the signals in real-time. This field of research is of practical interest but is a technical challenge. This chapter aims at introducing an approach to handling panoramic images streaming from the telepresence wheelchairs. Then, it will present extensive experiments to evaluate the quality of video at the receivers.

Moreover, the telepresence wheelchair is a mobile wheelchair platform which is capable of providing remote video from the unknown environment for remote monitoring and collaborative control to provide assistive support for people with disabilities. Due to the mobility of the wheelchair, real-time video streaming from the wheelchair to a remote user using the wireless network is challenging. The work in this chapter also aims to demonstrate the feasibility of a real-time video streaming solution, which leads to a prototype for a telepresence wheelchair in real-time.

The objective of this chapter is to propose a telepresence wheelchair by developing a video streaming framework where JPEG will be transmitted in the wireless network conditions. We investigate, design, and implement appropriate techniques to stream *real-world panoramic images* surrounding a wheelchair to a remote location via wireless networks in real-time. Particularly, we conduct real-time experiments in high-resolution panoramic images extraction and partition the images into packets to be suitable for the physical layer of the wireless transmission channel and then apply the transmission control protocol/Internet protocol and real-time transport protocol for transmission. The streaming rate and image quality are also assessed by objective and subjective quality evaluations.

A portion of the contents of this chapter has also been published in the author's conference article (Ha, Nguyen & Nguyen 2015).

The remainder of this chapter is structured as follows. Section 3.2 is devoted to the system architecture whereas Section 3.3 introduces the wireless video transmission. JPEG-based video coding and video quality assessment will be described in Section 3.4. Experimental results are given in Section 3.5. Finally, Section 3.6 summarizes the contributions in this chapter and discussion will be presented.

3.2 System Architecture

The design of this study aims to develop a telepresence wheelchair that provides video communication for the remote user. The system will be built to stream video which has captured the environment surrounding the wheelchair to the remote user. To this end, the image received from the cameras will be represented in panoramic images and be used to stream over the wireless network.

Our telepresence wheelchair system topology is presented in Figure 3.1. The system is based on server and client topology. The system consists of a telepresence wheelchair, a remote user, and wireless networks. The telepresence wheelchair consists of a powered wheelchair which is equipped with two main components, including a Mac mini computer and Ladybug3 camera. The Ladybug3 is an Omnidirectional camera, manufactured by Point Grey Research (Point Grey Research 2014). The detailed hardware of the wheelchair is described in Figure 3.2.

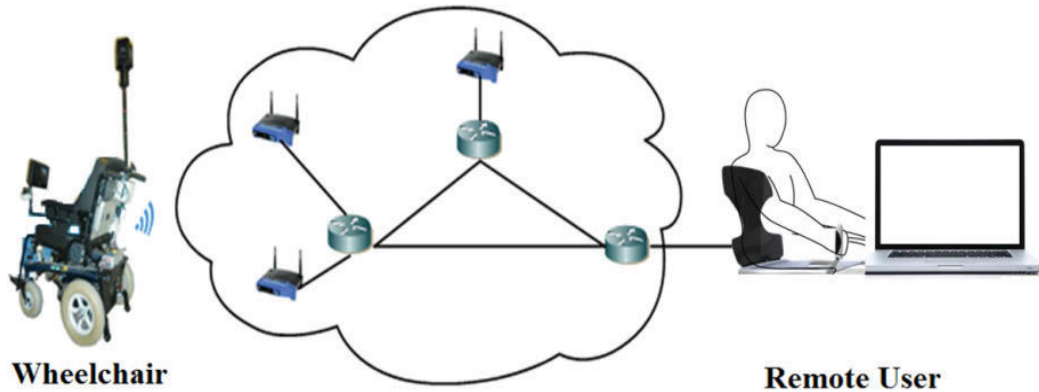


Figure 3.1: The telepresence wheelchair system topology.

In this study, the hardware of the telepresence wheelchair designed at the Centre for Health Technologies, the University of Technology Sydney is shown in Figure 3.2. Figure 3.2 illustrates the additional items attached on the front and the back of the wheelchair. In the normal configuration, the wheelchair is controlled by the joystick of the DX master remote. In order to integrate the modified parts into the system as a whole, the DX master remote has been reprogrammed so that it can work with control signals received from both the joystick (as the standard wheelchair) and the external signals fed via an additional DX module called a virtual joystick module. This modification allows the users to operate the wheelchair as either a standard power wheelchair or the navigationally assistive wheelchair. Instead of using the joystick, the on-board computer generates appropriate control signals following the calculations of the assistive navigation software. These signals are transferred to the virtual joystick module via USB 6008 to drive the wheelchair. The USB-6008 is used to interface between the computer and the virtual joystick module. The digital signals from the computer will be converted into the analog form and transferred to the virtual joystick via the pins of the USB – 6008. The device is also utilized to receive the joystick's signals via its analog input pins and to send them to the onboard computer.

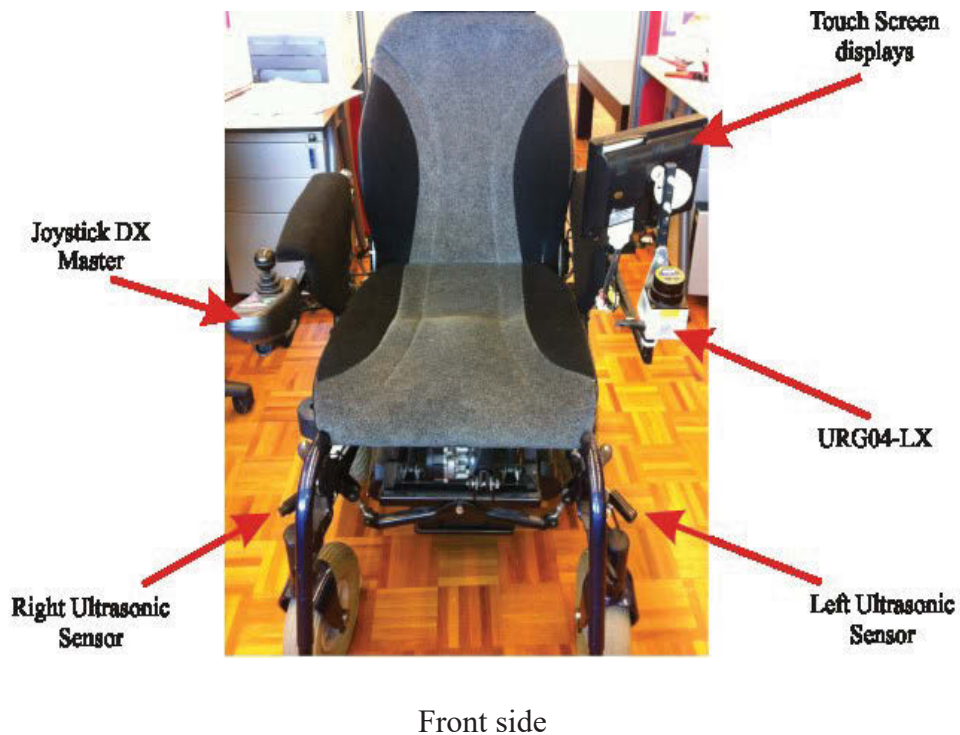


Figure 3.2a: The detail of the telepresence wheelchair hardware.

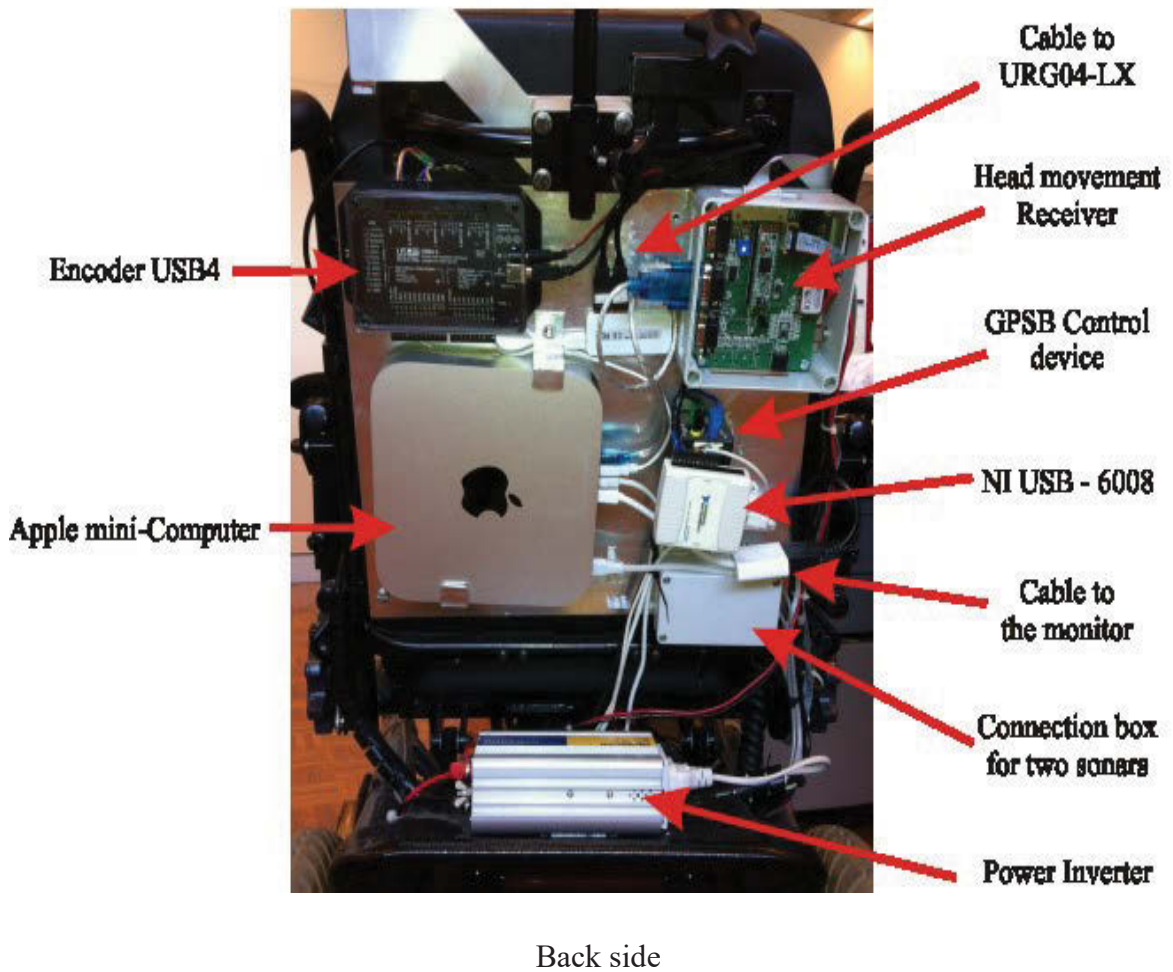


Figure 3.2b: The detail of the telepresence wheelchair hardware.

This system will process and transmit real world images around the wheelchair to the remote user over a wireless network in real-time while the wheelchair is moving at a remote location. In the wheelchair, the image frames are acquired from an Omnidirectional camera, then post-processed and encoded. Due to the limited bandwidth of the wireless channels, the original images are required to be compressed before being transmitted over the networks. Once the frame has been compressed, it is ready to transmit over the wireless networks.

3.3 Wireless Video Streaming

As discussed above, the data communication network infrastructure has a great impact on the performance of the telepresence systems. Thus, it is necessary to understand the operation of the wireless communication and how the video stream is encoded for transmission. Therefore, this section is continued by briefly discussing the basic background of wireless networking, some of its properties, video streaming over a wireless network and the factors which affect video streaming system design.

3.3.1 Wireless Networking

Our telepresence wheelchair will be connected to one of the most popular wireless networks, known as WLAN or "wireless fidelity" (Wi-Fi). WLAN or Wi-Fi are referred to as the IEEE 802.11 standard series. WLAN defines the Physical and Media Access Control layer for wireless communications in the Open Systems Interconnection (OSI) reference model. The OSI model is a reference model developed by the International Organisation for Standardisation to provide the universal standard setting for different manufacturers and to ensure connection compatibility among different software and devices. The original version of the OSI model, as shown in Figure 3.3, consists of seven layers: the physical layer, the data link layer, the network layer, the transport layer, the session layer, the presentation layer, and the application layer. In the OSI model, data communication is passed from the first layer of the sending side to the lowest layer and, then traverses the physical network connection to the bottom layer on the receiving side.

The IEEE 802.11 standards are commonly used to establish wireless connectivity for wireless terminals. The most popular standards in the 802.11 standard families are 802.11b, 802.11g, and 802.11n. The release years of different IEEE 802.11 standards are described in Figure 3.4. The IEEE 802.11a, b standard is the second generation of the wireless network which was first released in 1999. IEEE 802.11a operates in the available 5 GHz bands (5.15-5.35 GHz, 5.47-5.725 GHz, and 5.725-5.825 GHz) and IEEE 802.11b works in the band of 2.4 GHz. IEEE 802.11b can provide a data rate of 11 Mbps within around a 100-meter range in the 2.4 GHz frequency band.

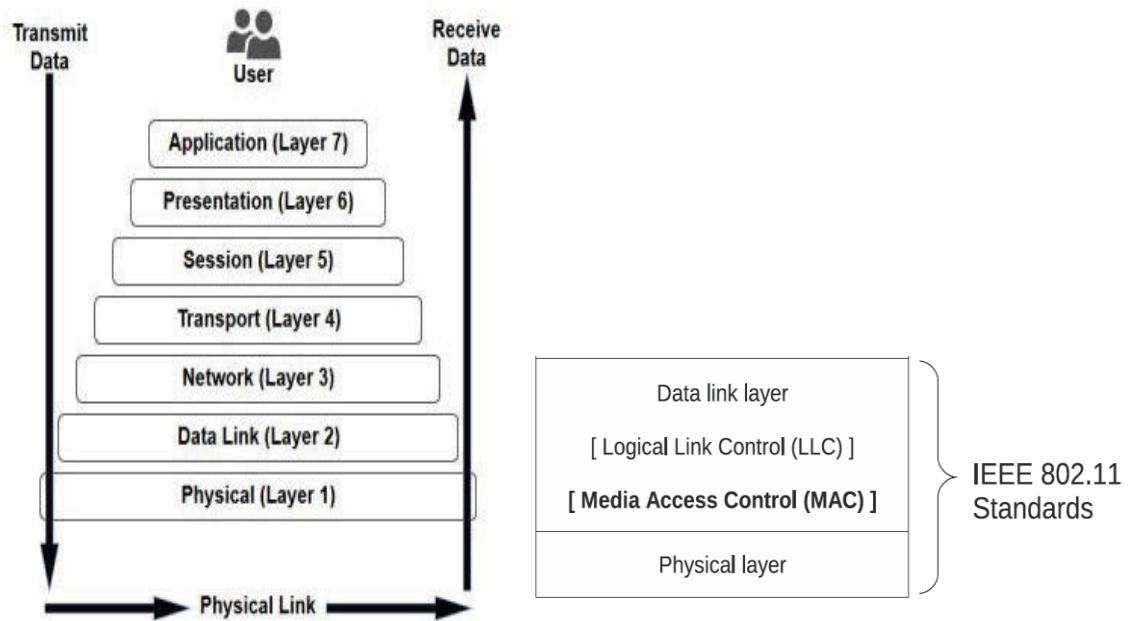


Figure 3.3: The relation of the IEEE 802.11 standard to the OSI reference model.

The growth and popularity of the wireless Internet connection and the huge demand for higher wireless throughput kept pushing forward the evolution of the IEEE 802.11 standard set. To improve the transmission data rate, the IEEE 802.11g was introduced in 2003. The IEEE 802.11g standard used the orthogonal frequency-division multiplexing (OFDM) modulation scheme to raise the maximum raw data rate to 54 Mbps in the same 2.4 GHz frequency band.

In the constant quest for greater transmission speed, IEEE 802.11n was released in 2009 as presented in Figure 3.4. IEEE 802.11n was formed to provide from 54 Mbps to 600 Mbps data rate in wireless communication. IEEE 802.11n operates both in 2.4 GHz and 5 GHz. It is an improvement of previous IEEE 802.11 standards such as a/b/g by adding multiple inputs multiple outputs (MIMO). When operating in 2.4 GHz, it is allowed by regulation to use 20 MHz wide channels and 40 MHz-wide channels in 5 GHz. The another significant advancement of IEEE 802.11n is the media access control (MAC) layer. IEEE 802.11n standard enables a WLAN to achieve up to 600 Mbps by employing four antennas tuned to the same channel and each transmitting a different spatial stream at a channel width of 40 MHz.

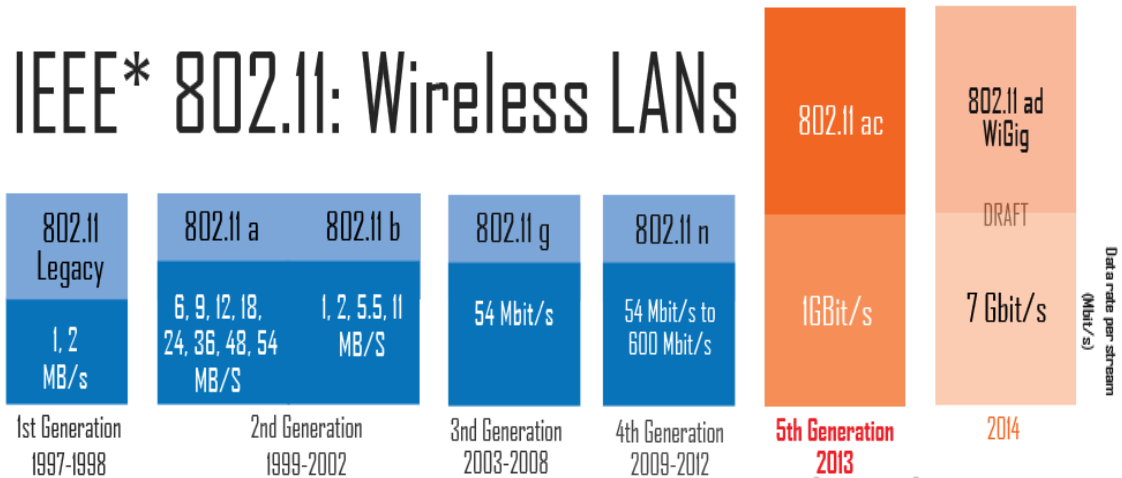


Figure 3.4: The bandwidth of the IEEE 802.11 standard (IEEE 2014).

To enhance performance, IEEE 802.11ac is designed to use the 5 GHz spectrum as illustrated in Figure 3.6, which is better and faces less interference than the 2.4 GHz frequency as shown in Figure 3.5. In comparison, the 5 GHz band provides 11 non-overlapping 40 MHz channels. The IEEE 802.11ac is the solution for high-bandwidth, high-quality of service (QoS) Wi-Fi communications. Moreover, IEEE 802.11ad uses the much higher 60 GHz spectrum to deliver up 7 Gbps. However, IEEE 802.11ad only works for short distances. With the increasing bit data rate, the current Wi-Fi networks can facilitate high-quality video streaming, which makes the telepresence systems with panoramic image streaming feasible.

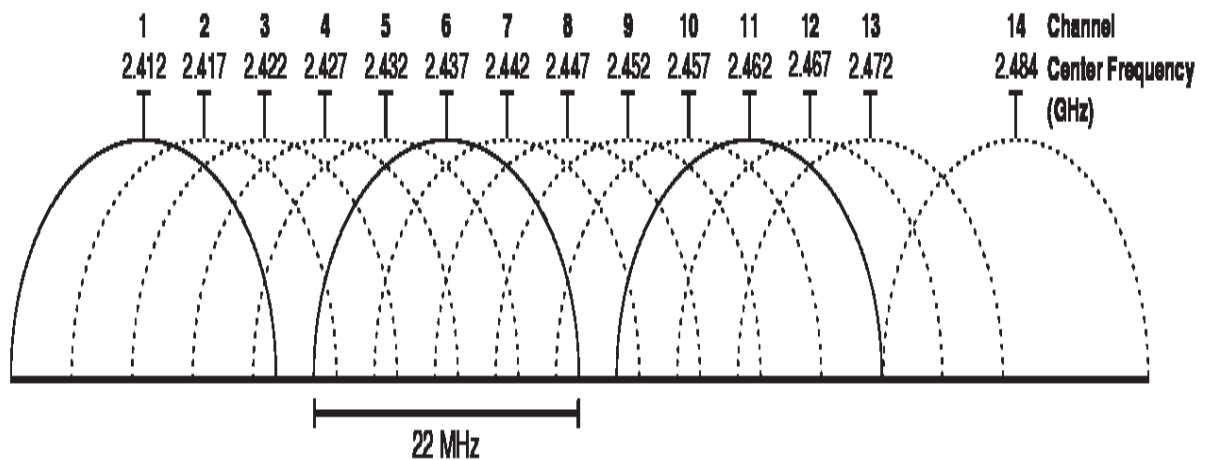


Figure 3.5: 2.4 GHz band for Wi-Fi channels (IEEE 2014).

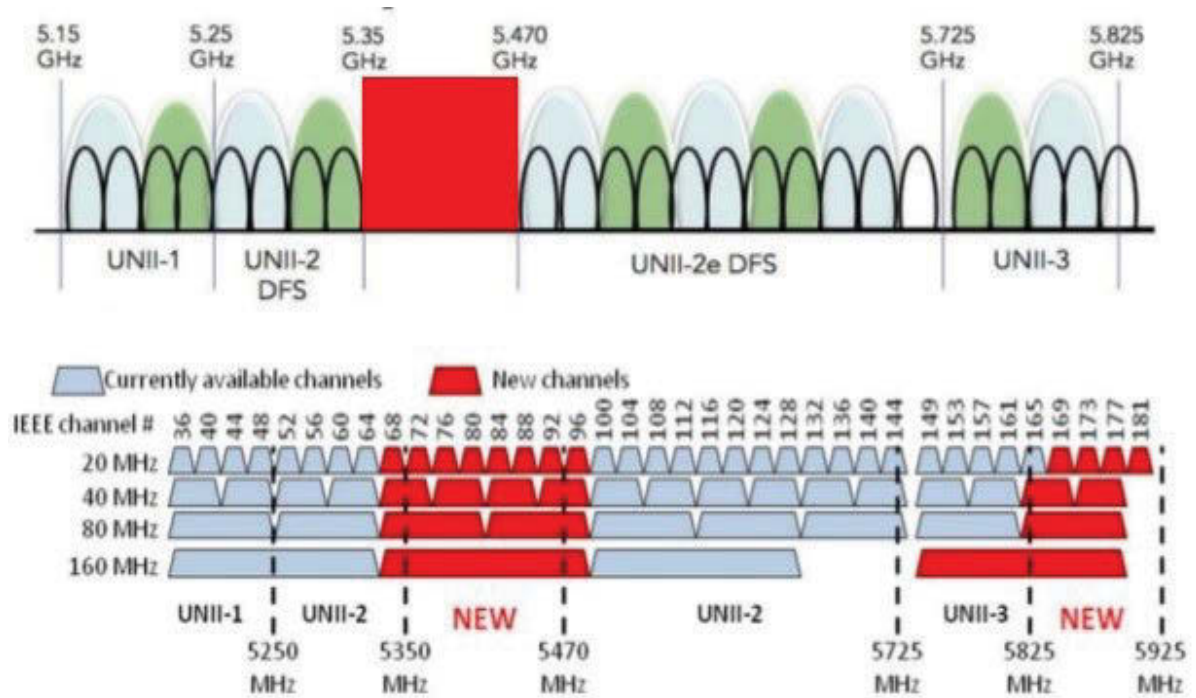


Figure 3.6: 5 GHz spectrum for Wi-Fi (Intel 2014).

3.3.2 Video Streaming over Wireless Networks

Video streaming has been an important media for communications and entertainment and has become very popular these days. There exists a very diverse range of different video streaming applications. However, video transmission over wireless networks is a significant challenge as compared with off-line, pre-encoded video streaming. This section shall discuss the video streaming approach and key challenges in wireless video streaming.

Real-time video streaming applications over wireless networks face various problems due to both the nature of wireless radio channels and the stringent delivery requirements of media traffic. One of the biggest issues is unpredictable behaviors of wireless channels due to the fluctuation of the wireless signals and wireless channel conditions. Bandwidth variations in wireless networks result in not only packet losses, but also residual bit errors (Martini et al. 2010). These losses and errors have a severely adverse effect on the compressed image bit stream. In some cases, the decoder fails to recover the compressed image. Thus, it is highly desired that the image streaming scheme can adapt the bit rate according to network conditions.

In the following, the video transmission system transports the input video in digital form from a source to destinations using wireless communication channels as described in Figure 3.7. Figure 3.7 presents the functional block diagram of the video transmission system over the wireless network. This figure illustrates the processing steps through a typical wireless video streaming.

- **Input video:** This block generates the video data which will be transmitted from the source to the destination.
- **Video encoder:** The purpose of the video encoder is to transform the incoming data generated by the input video to a more compact representation by using fewer bits. The process itself is called video encoding or compression.
- **Wireless transmission channel:** It is a physical medium used to carry the information from the sender to the receiver. A channel usually attenuates the signal, introduces noise, errors, and losses to the original information which may degrade the quality of the reconstructed information at the receiver side.
- **Video decoder:** This block tries to recover the source image from the received channel data.

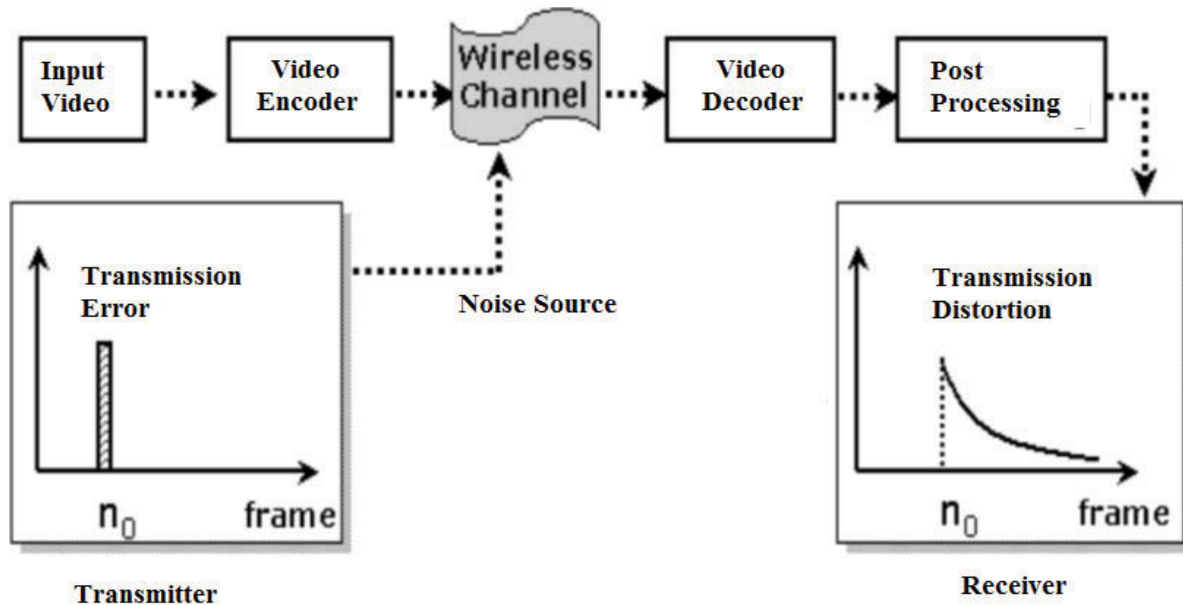


Figure 3.7: Video transmission over the wireless channel.

Note that interactive video communication of telepresence systems requires real-time processing, including capturing, encoding, transmission, receiving, decoding and displaying video. Video streaming for telepresence system design varies significantly if the characteristics of the communication channels, such as bandwidth, delay, and loss, are dynamic over wireless channels. For the wireless networks, the wireless channels are unreliable and provide no guarantees on bandwidth, delay jitter, or loss rate. The transportation service is in the best effort manner only and does not give any guarantee to deliver the multimedia packets in time. Some factors affect video streaming, and the quality of streaming video depends on the following limitations that can occur during the transmission:

- Packet loss

Packet loss refers to the failure of packets transmitted across networks to arrive at the destination. There are many factors which may cause packet loss. Packets are lost due to the network congestion in wired networks whereas in wireless channels the most common reasons are interference, fading, multipath effect and other impairments of the wireless links. Apart from channel errors during transmission, the main causes for packet dropping are queue overflowing and transmission collision. Packet loss can be measured as a percentage of packets lost on packets sent. Packet dropping may cause the degradation of video quality. A single packet dropped during the wireless transmission may not affect the video quality. Packet loss in the wireless network will cause the incomplete reconstruction of the frame at the receiver, leading to display distortion.

- Delay

Delay is defined as the period between the times that a video stream is sent from the source node to the time that the destination node receives it. Video streaming requires short delays. Packets that are received by the destination node are re-assembled by the decoding software in real-time to play out the video without pausing. The fluctuating bandwidth of a wireless link causes different delays for different video packets. When the delay of some packets is longer than others, the decoding process can be paused slightly from time to time. If a required packet arrives too late, either the packet is replaced with a fixed value by the

decoder or the decoder has to pause and wait for the arrival of the packet. The video quality is reduced in both situations.

- Jitter

Jitter is the variation of delay time taken for packets to travel between identical end-hosts due to queuing delays and link-level retransmissions. Jitter can lead to a degradation in system error performance. Jitter can cause jerkiness in playback of the receiver due to the failure of video packets and have to be skipped or delayed. The transmission jittering occurs mainly due to congestion in the wireless network. However, unlike packet dropping, jittering affects the smoothness of the video playback.

- Bandwidth fluctuation

Bandwidth is defined as the amount of data that can be transmitted in a fixed amount of time. The bandwidth is measured in bits per second (bps) or bytes per second or Hertz (Hz). The available bandwidth between two points in the wireless connections is time-varying. The congestion may occur if the sender transmits faster than the available bandwidth, which leads to packets loss. When a video is transmitting, it is important to make sure that the bandwidth available is larger than the video encoding bit rate. In order to overcome the bandwidth problem, a possible solution is to estimate the available bandwidth to control the transmitted video bit rate so that the video frame rate can be guaranteed for reconstructing the video smoothly at the receiver.

There exist some different approaches which need to be taken into consideration for video streaming over a wireless network. There are mainly four techniques to combat the effect of lost packets and increase the perceptual quality of the multimedia in the presence of lost packets. In order to combat the drawback of packet losses, a video streaming system is designed with error control. Four approaches have been considered including forward error correction, retransmissions, error concealment, and error-resilient video coding. (Grangetto, Magli & Olmo 2003; Seo & Jung 2011); and (Baruffa & Frescura 2015).

- Forward Error Correction (FEC) is used to add some redundant data to the source data so that the original data can be decoded even when some of the packets are lost.

- Automatic Repeat Request (ARQ): ARQ uses combinations of time-outs and positive and negative acknowledgments to determine which packets should be retransmitted. The sender keeps track of successfully received packets by using positive or negative acknowledgments received from the receiver. The packets which are not acknowledged in a certain time frame are retransmitted automatically.
- Error Resilient Coding: This technique tries to minimize the effect of lost packets at the compression level. It considers the semantic meaning of the compressed bit stream and attempts to limit the scope of damage caused by lost packets. Thus the decoded quality degrades gracefully with lost packets.
- Error Concealment: It is a receiver driven post-processing scheme which is applied at the receiver side. For example, in the case of video transmission, lost packets may cause some glitches in the decoded video which can be minimized by interpolating the missing parts from the received neighboring parts.

According to the IEEE 802.11 MAC layer standards, a packet is re-transmitted if the current transmission fails. Once the number of retransmission reaches the maximum allowed number, the packet is dropped. Retransmissions are attempted at the MAC layer in the absence of acknowledgment. Packets may be in error if the acknowledgment is not received from the destination within a specified duration after transmission. During retransmissions, queue backup in the transmit queue at the MAC layer can impact performance.

There is considerable existing literature on developing protocols for efficient data delivery over wireless networks. Some researchers proposed to use Transmission Control Protocol/Internet Protocol (TCP/IP) in streaming (Linck et al. 2006). TCP/IP is used for transportation to prevent the risk of data being corrupted while traveling over the wireless network as the protocol is connection oriented. The report in (Linck et al. 2006) has shown that TCP rate control has some advantages to provide guaranteed delivery and reduce packet losses. It has been observed that TCP-friendly rate control has the ability to coexist with other TCP-based applications. The other approach which is a scalable video streaming system using an adaptive control scheme was presented in (Hsiao et al. 2012). This method uses adaptive control techniques by composing of micro-control to change video frames at

the sender and receiver side to increase the performance without any quality of services (QoS) support from the network. The purpose was to improve video QoS during bandwidth fluctuation and bit rate changing hence to prevent packet losses.

There are several possibilities of image transmission through computer networks, including raw image data transmission and compressed image data transmission. In the compression case, the image is generated from a video digitizer and compressed before transmission. The idea of video streaming is to reduce the image size by encoding then to transmit it to the receiver to decode and display the video as these parts are received. The reduction of image size depends on the compression method and compression rate. Methods such as JPEG or MPEG are popularly used to downsize the images (Fowdur & Soyjaudah 2011). Recently, in the study of (Vajda et al. 2014), the authors presented a stream controller over the wireless network and algorithm control of the transfer rates in video channels where JPEG compression was presented.

Therefore, one of the primary objectives of video streaming over the wireless network is to deal with the fluctuation of an unknown and dynamic wireless signal. It is important to reduce the packet dropping, delay and jittering to improve the quality of video transmission in a wireless network. Moreover, it is crucial to match the video bit rate to that which the channel can support. To reduce the effect of packet loss and delay jitter, what is needed is a video coding method which can enhance the video quality in the receiver.

3.4 JPEG-Based Video Encoding and Quality Assessment

3.4.1 JPEG Coding

Regarding video transmission, images require high storage capacity and an enormous amount of transmission bandwidth. It is important to identify the type of video that is being transmitted and the video processing technique which is needed for efficient transmission. Thus, this section provides video compression which is developed for video streaming.

To begin, this work considers the JPEG image compression, which is designed to exploit the compression images for wireless video streaming. Video compression standards provide some benefits, foremost of which is ensuring interoperability, or communication between

encoders and decoders. Compression in the multimedia system is able to make an implementation possible, and the complexity of the technique used should be minimal.

To reduce the image data for efficient transmission, we exploit the image compression scheme, namely Joint Photographic Experts Group (JPEG). JPEG is an image compression method for image data transmission over the digital communication networks (ITU-T 1993). The JPEG has been used in several different applications. These can be anything from a digital camera to a popular video game. The key steps in JPEG encoding and decoding are outlined in Figure 3.8 and Figure 3.9, respectively.

At the encoding side in Figure 3.8, the uncompressed image samples are grouped into data units of 8×8 pixels and passed to the encoder for color transform calculated by

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299000 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500002 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (3.1)$$

C_b and C_r components are down-sampled since human eyes cannot see considerably into the brightness of the image. These data units of 8×8 shifted pixel values are defined by $f(x, y)$, where x and y are in the range of zero to seven. Each of these values is then transformed using Forward Discrete Cosine Transformation (FDCT). The FDCT is then applied to each transformed pixel value by Equation (3.2). This transformation is carried out 64 times per data unit. The resulting 64 coefficients of $F(u, v)$ are given by

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \left[\frac{\pi(2x+1)u}{16} \right] \cos \left[\frac{\pi(2y+1)v}{16} \right] \quad (3.2)$$

for $u = 0, \dots, 7$ and $v = 0, \dots, 7$

$$\text{where } C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases}$$

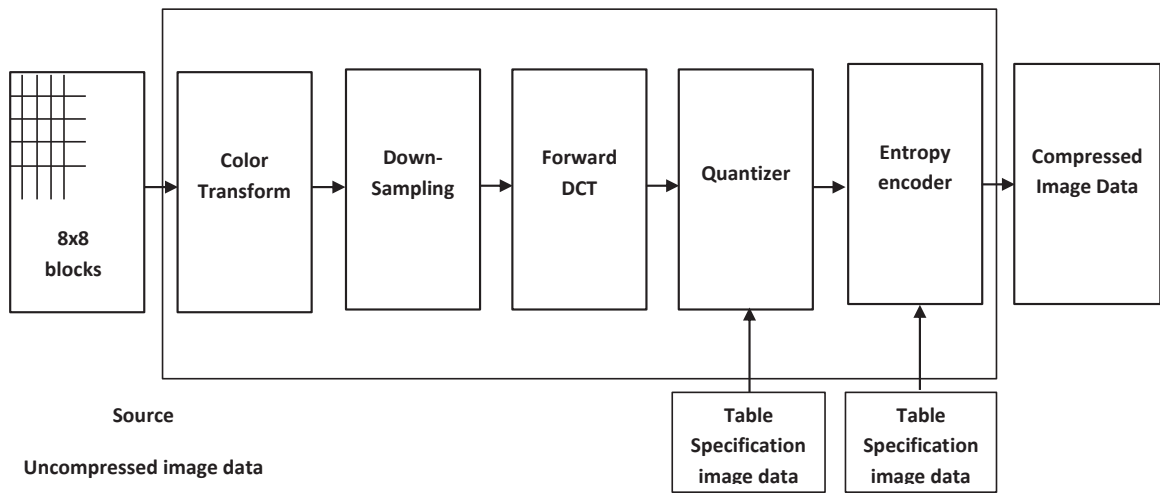


Figure 3.8: Block diagram of the JPEG Encoding.

- **Quantization:** Following the forward DCT steps in Figure 3.8, the quantization of all DCT-coefficients is performed. In this step, the JPEG application provides a table with 64 entries. Each entry will be used for quantization of one of the 64 DCT-coefficients. These coefficients should be determined according to the characteristics of the source image. The possible compression is influenced at the expense of the possible image quality.
- **Entropy encoder:** during the initial step of entropy encoding, the quantized DC-coefficients are treated separately from the quantized AC-coefficients. This processing order of the whole set of coefficients is specified by the zig-zag sequence. The Huffman and arithmetic encoding are used as entropy encoding methods.

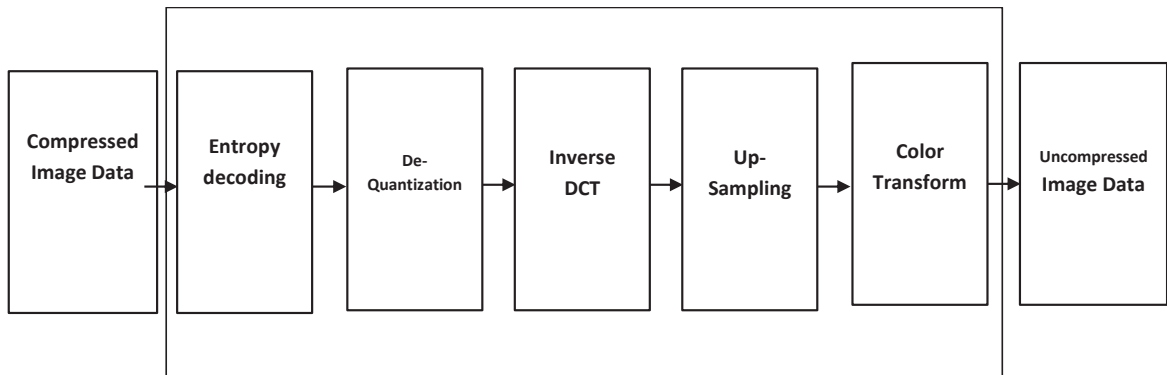


Figure 3.9: Block diagram of the JPEG Decoding.

After the transmission of media, the resource must be decoded, that is, the image reconstruction will be done at the receiver. The block diagram of the JPEG decoding processing is as shown in Figure 3.9. In the reverse direction, the decoding steps are as follows:

- Entropy Decoding: The first phase of the JPEG decoding process is entropy decoding. This processing step is a form of lossless data decompression, which uses Huffman coding to decompress the image data. Huffman uses the predefined tables to decode the compressed data. Huffman tables and quality tables are described in the JPEG image's header.
- De-quantization: in this step, all 8x8 pixel blocks are now ready to be decompressed. De-quantization is done through 64 integer multiplication by values from a JPEG standard quality matrix. A corresponding constant multiplies each value of the frequency domain for that component in the quality matrix.
- Inverse Discrete Cosine Transform (IDCT): To change the frequencies into color values, Inverse Discrete Cosine Transform has to be applied to all blocks. IDCT works by converting parts of different frequencies into signals from the entire 8x8 block. The IDCT equation computes the i, j th entry of the DCT of a block, given by

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v)F(u, v) \cos\left[\frac{\pi(2x+1)u}{16}\right] \cos\left[\frac{\pi(2y+1)v}{16}\right] \quad (3.3)$$

for $x = 0, \dots, 7$ and $y = 0, \dots, 7$

After applying IDCT on all 8x8 blocks, all block values are level shifted by 128.

- Up-sampling to increase the resolution of the CbCr components by the specified ratio. After up-sampling, the pixel ratio between all three elements is one. In other words, there are as many chrominance components as luminance components. The Y component is never up-sampled or down-sampled because the human eye is more sensitive to low-frequency color information, the brightness of a pixel, rather than high-frequency information, color of a pixel.
- Color space transformation: the colors represented in the image are converted from RGB to YCbCr. Y describes the brightness of the pixel, and the two chrominance components CbCr describe the color of the pixel. Equation (3.4) describes the conversion of YCbCr to RGB when decoding JPEG data.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.40210 \\ 1.0 & -0.34414 & -0.71414 \\ 1.0 & 1.77180 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix} \quad (3.4)$$

3.4.2 Video Quality Assessment

Understanding and evaluating the qualities of video are very necessary for measuring the system performance and verifying the feasibility of the video transmission over the wireless network. Signal-to-noise ratio (S/N or SNR) is the most widely used parameter for the measurement of signal quality in the field of transmission. Signal-to-noise ratio expressed in decibels is the amount that the signal level exceeds the noise level in a specified bandwidth. The SNR is expressed for video transmission at the corresponding receiver as:

$$\frac{S}{N} = \frac{\text{Average power of a signal}}{\text{Noise power}} \quad (3.5)$$

Video quality is subjective to the viewer and it is closed related to the S/N of the image. The Television Allocations Study Organization (Freeman 2005) developed a 4-point scale known as “TASO ratings” to evaluate the quality of the image as shown in Table 3.1.

Table 3.1: TASO Picture Rating.

Quality		S/N
1	Excellent	45 dB
2	Fine	35 dB
3	Passable	29 dB
4	Marginal	25 dB

For lossy image compression, the quantization step is responsible for the source distortion in the code and determines the compression rate. The performance measurement parameter, namely mean square error (MSE) is calculated by

$$MSE = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - K(i,j)]^2}{M * N} \quad (3.6)$$

Where $I(i,j)$ is the original image, $K(i,j)$ is a reconstructed image, M and N represent the dimensions of the images.

To measure the quality of reconstructed images, the peak signal-to-noise ratio (PSNR) is calculated from the obtained MSE (Martini et al. 2010). PSNR is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR (dB) is most commonly used to measure the quality of reconstruction of lossy compression codecs and wireless transmission. The signal in this case is the original data, and the noise is the error introduced by compression or transmission. PSNR is most easily defined via the mean squared error (MSE). Given a noise-free $M \times N$ monochrome image I and its noisy approximation K , MSE is defined as:

$$\begin{aligned}
 PSNR &= 10 \log_{10} \left[\frac{MAX_I^2}{MSE} \right] \\
 &= 20 \log_{10} \left[\frac{MAX_I}{\sqrt{MSE}} \right] \\
 &= 20 \log_{10}(MAX_I) - 10 \log_{10}(MSE)
 \end{aligned} \tag{3.7}$$

where MAX_I is the maximum possible pixel value of the image. When the pixels are represented by 8 bits per pixel, the MAX_I value is 255. Typical values for the PSNR in the lossy image and video compression are between 30 and 50 dB. The International Radio Consultative Committee (CCIR) developed a 5-point scale for picture quality versus impairment is listed in Table 3.2.

Table 3.2: CCIR Five-Grate Scale (Freeman 2005, Xu et al. 2015).

PSNR (dB)	Quality		Impairment	
≥ 37	5	Excellent	5	Imperceptible
32-37	4	Good	4	Perceptible, but not annoying
25-32	3	Fair	3	Slightly annoying
20-25	2	Poor	2	Annoying
< 20	1	Bad	1	Very annoying

3.5 Experiments and Results

Our designed telepresence wheelchair consists of the system topology in Figure 3.1 and the hardware system presented in Figure 3.2. The technical solutions including wireless video streaming in Section 3.3 and JPEG-based video encoding in Section 3.4 are integrated into the system to develop a telepresence wheelchair system. The purpose of the following experiments is to demonstrate the feasibility of the images streaming wirelessly from the telepresence wheelchair system. In this section, the experiments are divided into two major parts. Section 3.5.1 contains the experiments which are implemented to acquire images and to encode the video into JPEG data. By encoding the images in real-time, the output compressed images are employed for the experiment in the next section. Section 3.5.2 describes the experiments of streaming images over wireless channels when the wheelchair is moving to a remote user. By practical experiments, the streaming performances are also evaluated and analyzed in order to find appropriate video streaming approaches for developing a telepresence wheelchair.

3.5.1 Experiment 1: Images Acquisition

In the first experiment, we aim to generate and encode the images into a suitable format for data streaming over the wireless network. In order to capture live images around the wheelchair, an omnidirectional camera is attached to the wheelchair and connected to a computer as a server. The purpose of the experiment is to analyze the ability of the system to extract and compress the source video into JPEG codec from the camera in real time at a fast frame rate and high quality. In this experiment, images processing techniques including images acquisition, post-processing, and JPEG based encoder are implemented.

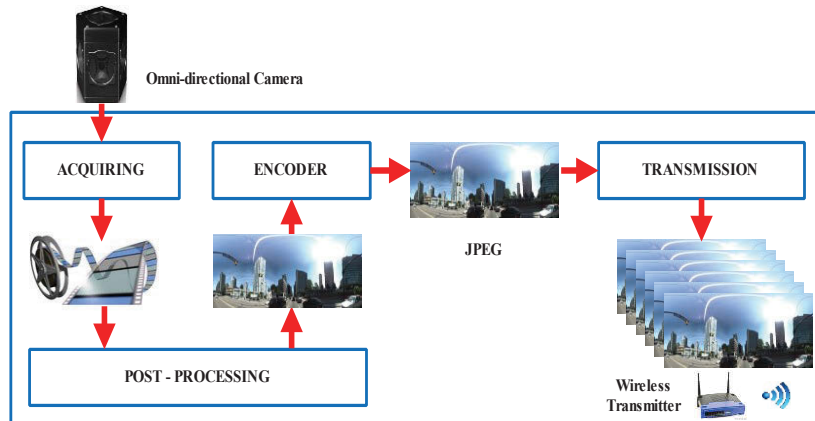


Figure 3.10: Framework for video streaming over the wireless network.

The diagram in Figure 3.10 describes the operation of the telepresence wheelchair server. In this server, frames are acquired from a Ladybug3 camera and then they are post-processed and encoded. Due to the currently available technology and quality requirements, the original images must be compressed before being transmitted over the networks to minimize the congestion of the networks. Once the frame has been extracted and compressed, it is ready to transmit over wireless networks.

In this experiment, a Ladybug3 camera was used. The Ladybug3 is a spherical camera produced by Point Grey Research which offers high megapixel resolution and a high-speed interface. Ladybug3 provides an IEEE 1394B Firewire interface for video streaming. This multi-sensor system has six cameras: one points up, the other five point out horizontally in a circular configuration. However, when the camera operates, stream files are saved on the computer. The output videos are written to a set of Ladybug stream files. If the compression is disabled, the camera transmits uncompressed images at transfer rates of 800 Mbit/s and 30 frames per second (fps) to a hard disk. The size of each stream file is limited to 2 Gigabytes. The stream files are named as stream base name - stream serial number. Pgr is as shown in Figure 3.11 (Point Grey Research 2014).

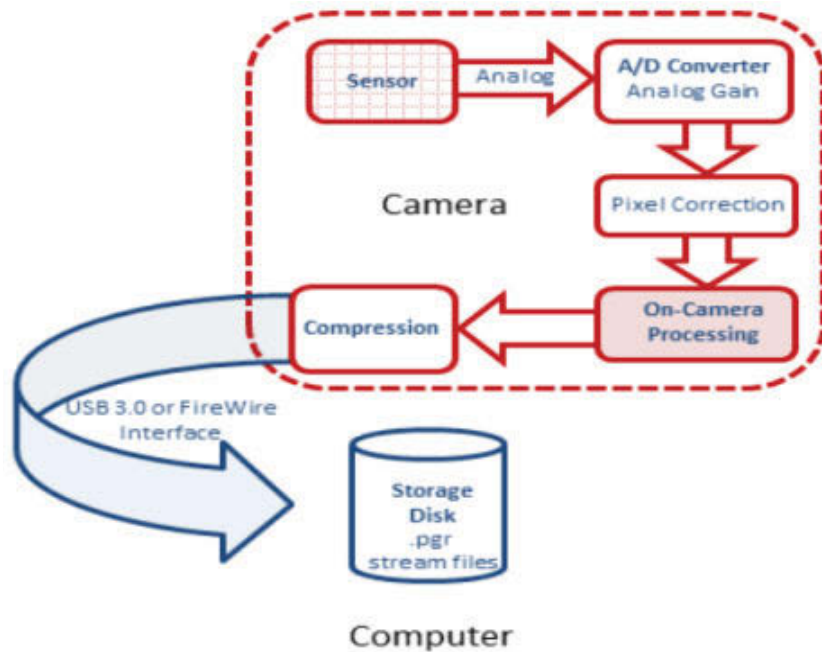


Figure 3.11: Video stream from Ladybug3.

For images acquisition in the experiments, we extracted the output of the captured images of a sequence of six images. These six images are stitched together to form a single panoramic image. This process is done by projecting a spherical mesh using the Ladybug Software Development Kit (SDK). The configuration and side views of the Ladybug3 camera are illustrated in Figure 3.12. The specifications of LadyBug3 used in our system are listed in Table 3.3.

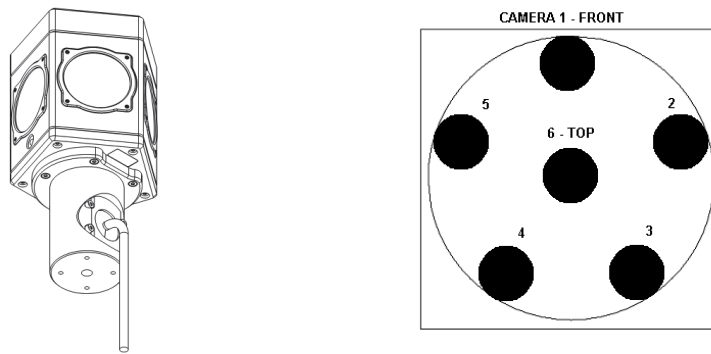


Figure 3.12 The configuration and side views of the Ladybug3 camera.

Table 3.3: Camera Specifications. (Point Grey Research 2014)

General Specifications	Description
Camera Model	LD3-20S4C
Overview	Single unit, high-resolution IEEE-1394b spherical digital video camera
Imaging Sensor Type	Six (6) Sony progressive scan color CCDs (five in horizontal ring, one on top)
Maximum Resolution	1616(H) x 1232(V) (each sensor)
Pixel Size	4.4 μm (H) x 4.4 μm (V)
Video Data Output	8-bit raw Bayer (color) digital data
Signal To Noise Ratio	62 dB at 0 dB gain
Digital Interfaces	9-pin 1394b (FireWire) 800 Mb/s
Power Consumption	7.2W at 12V
Dimensions (W x H)	134mm x 141mm

A post-processing module receives panoramic images from panorama cameras and, then, adjusts the images to improve their contrasts, brightness, and colors. After processing, the adjusted images are sent to the encoding module. At this stage, the images are converted and encoded with different resolutions to reduce network bandwidth consumption. All of the panorama images are encoded into JPEG format. In our experiments, these six images were processed to be composited together to form a panoramic image. This obtained image is compressed into JPEG format in real-time for transmission.

Algorithm 1 presents the procedure that is used to implement in C++ to extract the high-resolution panoramic images from the Ladybug3 camera.

Algorithm 1: Real - time extraction panoramic images

Input: Resolution

Output: BMP/JPEG image file

1: Begin: *initialize camera*

Ladybug front 0 pole 5

Color processing method = downsample4

ladybug_panoramic

2: Start the camera

3: Set the size

4: Loop: {*Grab an image*

Convert to 6 RGB buffers

Send RGB buffers to graphics card

Stitch the images

Output image file}

5: End

We developed software modules in which an Algorithm 1 is implemented based on Visual Studio Dot Net framework and Ladybug3 API. The image acquisition, post-processing, and the encoding module were implemented using C language. The streaming module was developed in C#. The following is a recommended system configuration of a computer to

support the Ladybug 3 camera. The Mac book Pro with Intel Core i7 Duo or Quad processor, 8 GB of RAM, video card with 512 MB RAM, IEEE-1394b interface, Windows 7. In the implementation, the encoder takes original video and compresses the image frames into JPEG. The JPEG data are converted to bitstream, which is streamed to the decoder at the remote user to produce the reconstructed video. In this study, the image data stream can be encoded and decoded in real-time using C# on Windows Operation System platform.

3.5.2 Experiment 2: Images Streaming

The second experiment was conducted to stream the JPEG images over the wireless network from the wheelchair to a remote computer. To simplify data communications model networking, this experiment investigates the image streaming in real-time over a server and client topology. A streaming process was created consisting of two hosts (server and client). The server sends JPEG panoramic images to the client via the topology as shown in Figure 3.13. The Wi-Fi connections with IEEE 802.11b WLAN configuration were used for wireless links between the server and the client.

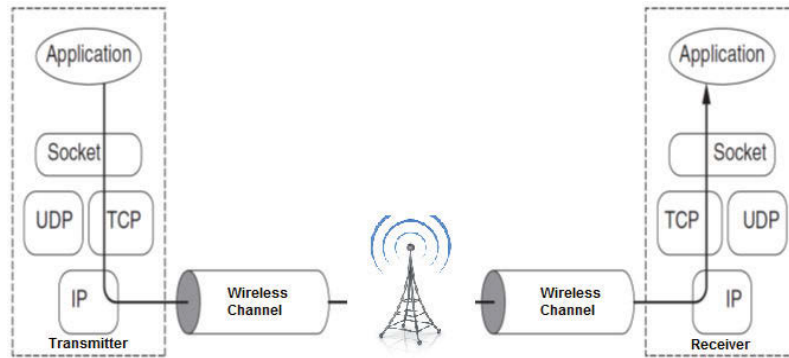


Figure 3.13: Server and client topology.

The most important thing in the system implementation is the evaluation of the system performance during the coding and transmission process. Some software modules were developed and implemented. Media streaming applications over wireless links face various challenges, due to both the nature of the wireless channel and the stringent delivery requirements of media traffic which can lead to packet losses and residual bit errors. There are several requirements on the communication data networks when the images are transmitted: the network must accommodate burst data transport, which requires reliable transportation, time-dependence.

There were many study tools used in the experiment for video streaming and system evaluations, including MATLAB and Wireshark (Wireshark 2016). Wireshark is a network protocol analyzer tool for capturing traffic on a computer network and displaying the contents from a network packet for analyzing. It was used to analyze the traffic passing between the server and the client in order to analyze the performance and packet transmission.

In this experiment, the quality of received images was taken to observe the performance of the system when the link experienced data loss. At a receiver side, under certain circumstance, there is a distortion due to the interference, the channel noise, the compression. Therefore, the Transmission Control Protocol/ Internet Protocol (TCP/IP) is adopted for the transmission protocol because there is a guarantee of reliable communication. Also, to minimize the distortion and overcome the errors on a compressed JPEG image streaming over wireless networks, there exist various error protection and error correction techniques.

In our method, we handle the error problems by applying an image-packetization scheme, retransmission mechanism, and buffer control. The receiver checks the packet for any errors and sends a positive or a negative acknowledgment (ACK) to the transmitter depending on whether the packet is received correctly or erroneously. As the transmitter receives a negative ACK signal, it resends the packet. On the other hand, if the transmitter receives a positive ACK signal, it sends the next packet. With such an error control, we can decrease packet losses to enhance the image quality. The main issue of this technique is that retransmission timeout may delay the streaming.

We carried out ten experiments at various locations to stream panoramic images with a resolution of 2048x1024 pixels from a wheelchair at zone 1 and zone 2 to a client at the remote location via a wireless network. In our set up, both the wheelchair and the client used Wi-Fi connections. Table 3.4 lists the parameters of system configuration.

Table 3.4: Network settings for experiments.

Parameter	Value
Wireless	IEEE 802.11b/g/n in 2.4GHz
Transport protocol	TCP/IP
Bandwidth	20 MHz
Number of Access Points	2
Speed Requirement	Real-time
Flow direction	Duplex
Vehicle speed	5m/s

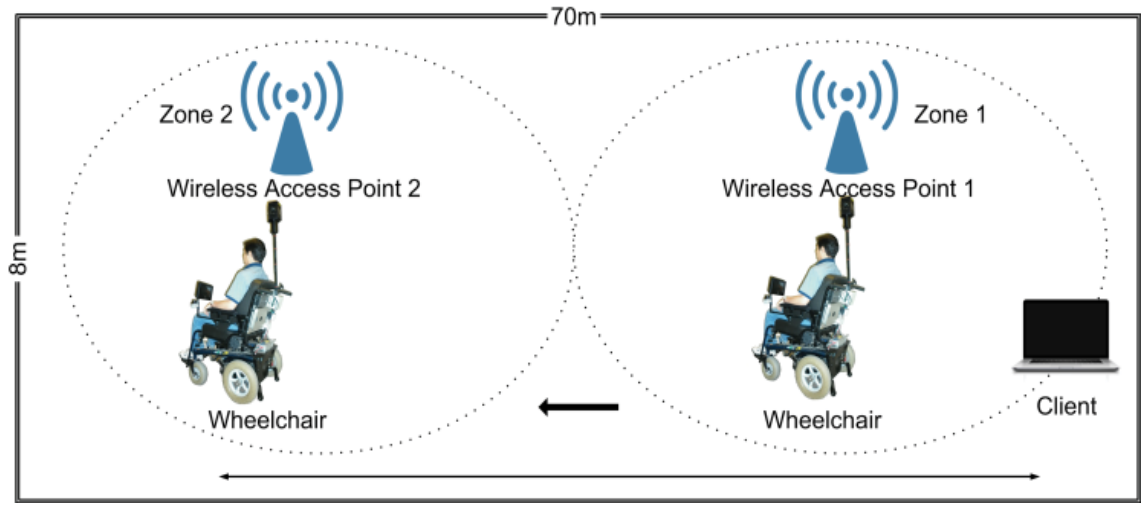


Figure 3.14: System performance test at the Centre for Health Technologies.

In our experiments, we assumed a topology in which one or two wireless access points were involved. The wheelchair sent images over a wireless connection to a client. Experiments were performed at the University of Technology Sydney, Centre for Health Technologies laboratory as shown in Figure 3.14. The wheelchair battery is expected to last for 8 hours and should be charged overnight. The wheelchair speed was set to 5 m/s and below to ensure that the wireless connection could be maintained.

3.5.3 Results

Experimental results showed that the proposed methods are successful in processing for images acquisition and encoding of high-quality images. The samples of the output images are illustrated in Figures 3.15 and 3.16. In order to evaluate the performance of the output images, a set of 500 samples of JPEG images were extracted to compare with the second

data set contained 500 samples of the Bitmap samples with the same resolutions. The comparison of the two datasets is presented in Figure 3.17. Figure 3.17 shows the comparison of panoramic images capacity with two different types of file extensions used for graphic files. The resolution of each image is 512x256, 1021x512, 2048x1024, 3500x1750 and 9600x1080 pixels, respectively. As compared to Bitmap format which is uncompressed images, JPEG images have significantly smaller sizes. JPEG images are significantly decreased in file size. JPEG uses a lossy compression algorithm that removes portions of data from the image.



Figure 3.15: Six images were taken from six camera units of Ladybug3.



Figure 3.16: A JPEG panoramic frame generated after processing.

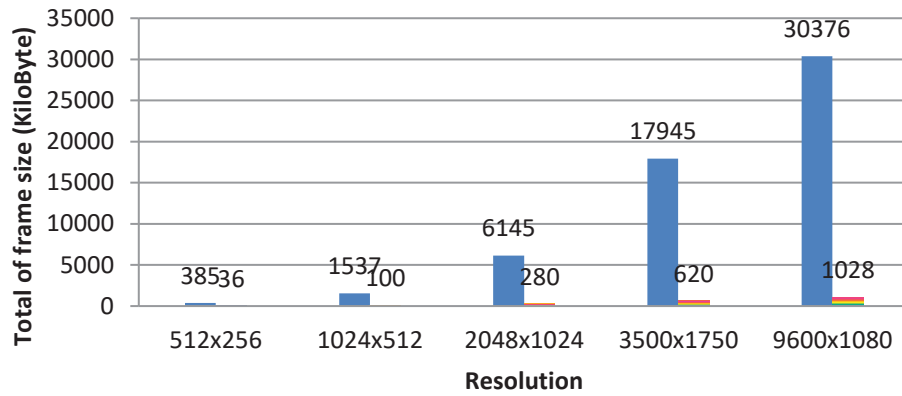


Figure 3.17: Bitmap and JPEG file size comparison.

In terms of images streaming over the wireless, the experiment showed that the wheelchair system was able to stream the images when it was moving in real-time. However, at the first stage of the experiments, the system detected that the change of wireless signal could lead to packet loss as shown in Figure 3.18. Therefore, the quality of images had dropped more than expected when packet loss increased. The identified problem is illustrated in Figure 3.21.

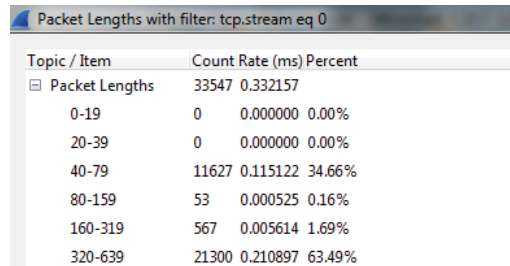


Figure 3.18: Packet loss during transmission.

From theory, it is expected that TCP would have adequate functionality to handle packet drops in the network. However, in wireless transmission, packet loss affects the system performance, and even low packet loss rates may cause poor quality of obtained images. The experimental results of wireless streaming without errors control are illustrated in Figures 3.19, 20, 21 and 22. The figures illustrate the original image at the server and the received image at the client. From the below Figure 3.19, it can be seen that performance decreased when images are streamed over the wireless network. It is possible to see from Figures 3.20 to 3.21 and the result of the histogram of images that the quality of streaming is not acceptable.



Figure 3.19: The original image at the server.

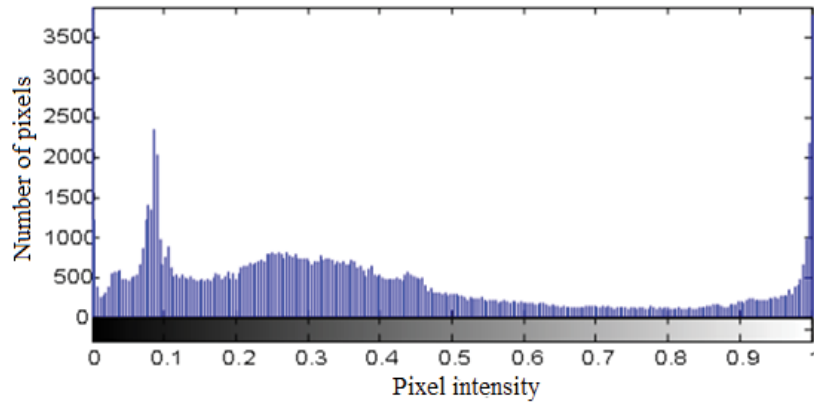


Figure 3.20: The histogram of the original image.



Figure 3.21: The image received at the client.

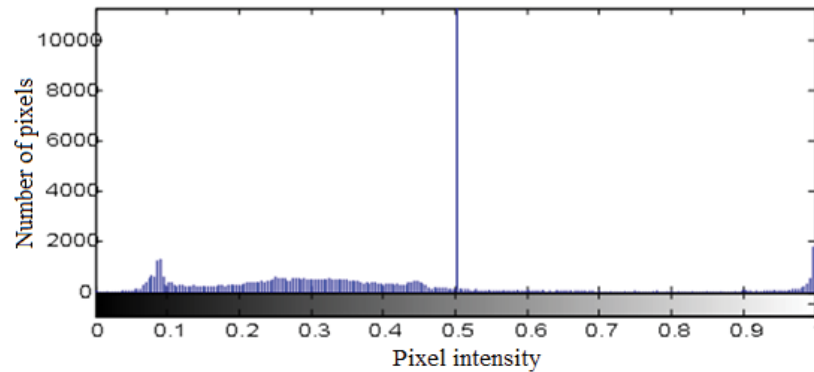


Figure 3.22: Histogram of the obtained images during the streaming process with $MSE = 1.8651e+03$ and $PSNR = 15.4239$ dB.

At an early stage of the experiment, it can be observed that the distorting images obtained by the experiment were carried out without error protection. The errors introduced in the data stream by the lossy channel over wireless are shown in Figures 3.23 and 3.24. We noted that transmission errors in JPEG compression images distort a significant portion of the image. Fading and Interference problems plague wireless channels. Any data loss in bit stream will affect the consequent bit and even possibly destroy the whole picture. Bit errors that occur at marker segment positions can severely degrade an image. In some cases, an image will even be lost completely. The decoder fails to recognize and recover the JPEG compressed image.

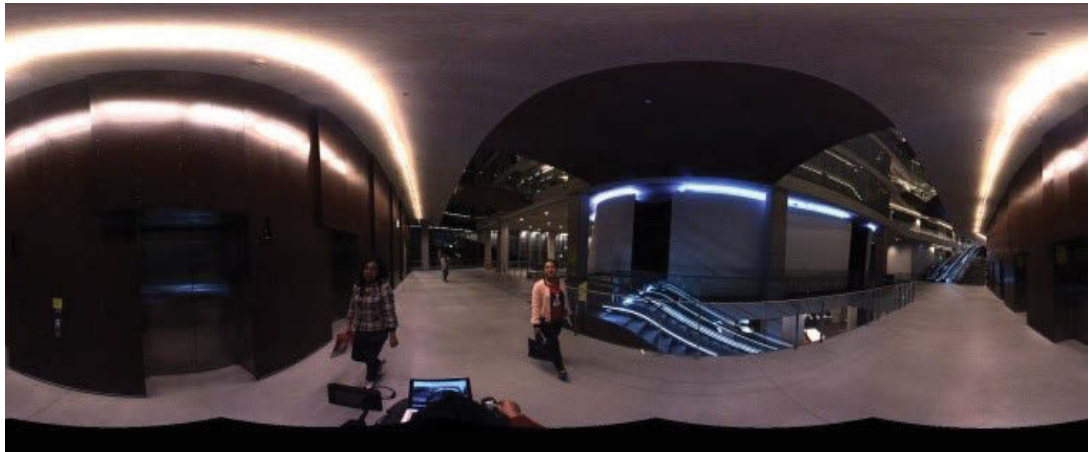


Figure 3.23: An original panoramic image extracted from the wheelchair.

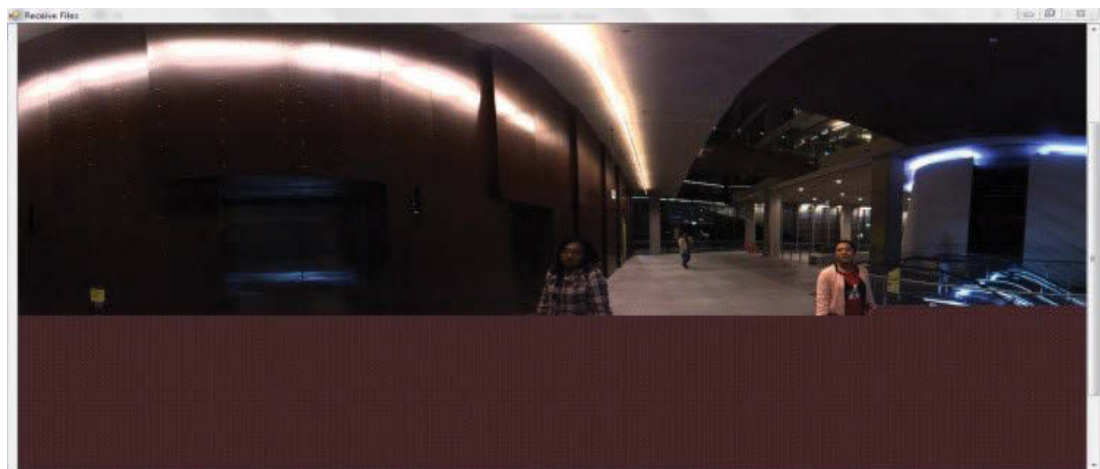


Figure 3.24: The image obtained over wireless with errors.

In order to overcome errors in a compressed data bit stream over the wireless network, many different protection errors and error-control techniques, such as forward error correcting (FEC) codes and automatic repeat request (ARQ) schemes, must be used. In ARQ schemes, a feedback channel is provided to retransmit erroneously received data packets. ARQ schemes are capable of delivering almost error-free data packets to the destination at the expense of massive delays. Depending on the error sensitivity of the compressed image, we implemented the ARQ protocol; the transmitter adds an error detection code and a cyclic redundancy code (CRC) to the data and sends the data in packets to the receiver. The receiver checks the packet for any errors and sends a positive or a negative acknowledgment to the transmitter depending on whether the packet is received correctly or erroneously. A negative acknowledgment signals the transmitter to resend the packet; a positive acknowledgment signals the transmitter to send the next packet. We implement such an error control that can decrease packet losses and packet delivery delays to enhance the image quality. As a result, the performance of image streaming was smooth on the receiving side with good quality and better visualization as shown in Figure 3.26.

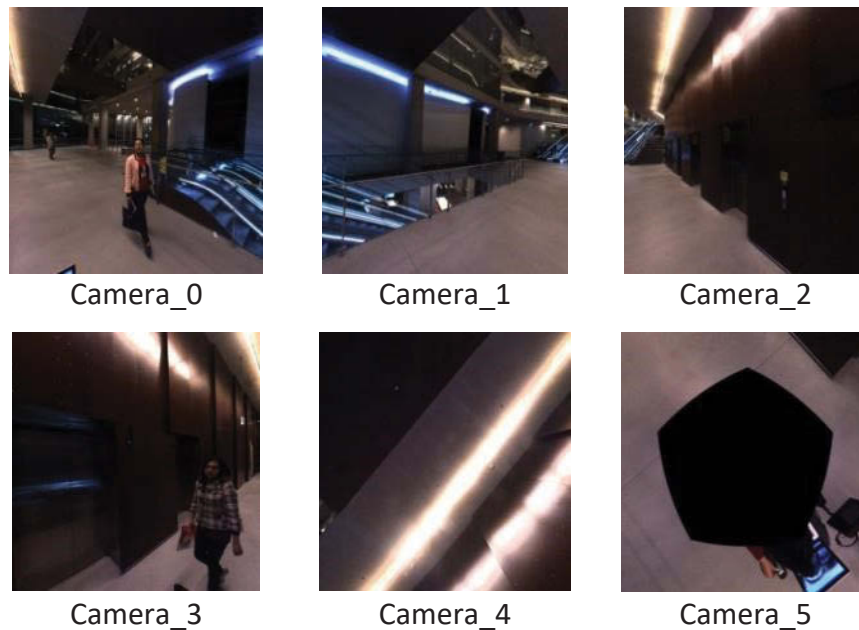


Figure 3.25: A sequence of six frames is extracted from the camera attached to the wheelchair.

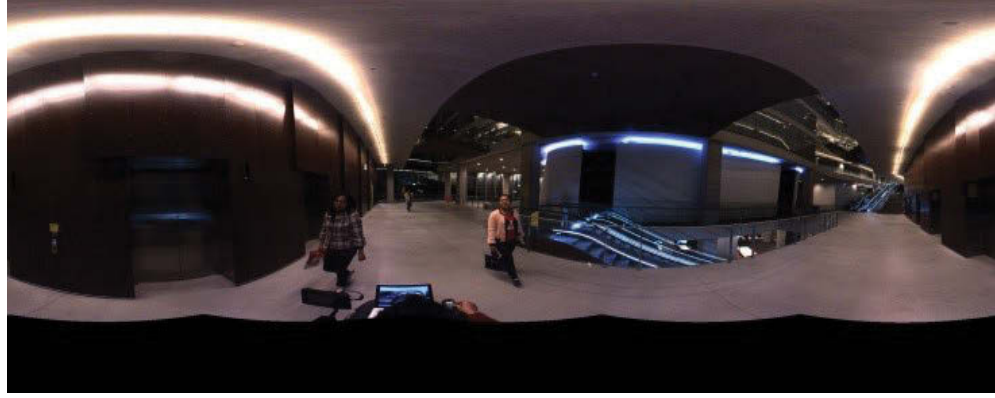


Figure 3.26: The image obtained at client side over wireless with our scheme.

For evaluation, we analyzed both image quality and network streaming performance. We compare the images of the receiver and the image of the transmitter by measuring the mean square error (MSE) and peak signal to noise ratio (PSNR) for images assessment. The extracted 100 frames were analyzed. At the receiver side, the experimental results demonstrate that the received images are of high quality as illustrated in Figure 3.26. From Figure 3.26, it can be seen that the images obtained at the receiver were similar to the original images without data loss. The mean square error is shown in Figure 3.27. Moreover, the peak signal to noise ratio is shown in Figure 3.28. It can be observed from Figure 3.28 that the average PSNR of the whole sequence of one hundred frames is 31.64 dB. In a comparison of the rating of CCIR Five-Grate Scale in Table 3.2, it can be observed that the average PSNR is in the high-quality image performance threshold.

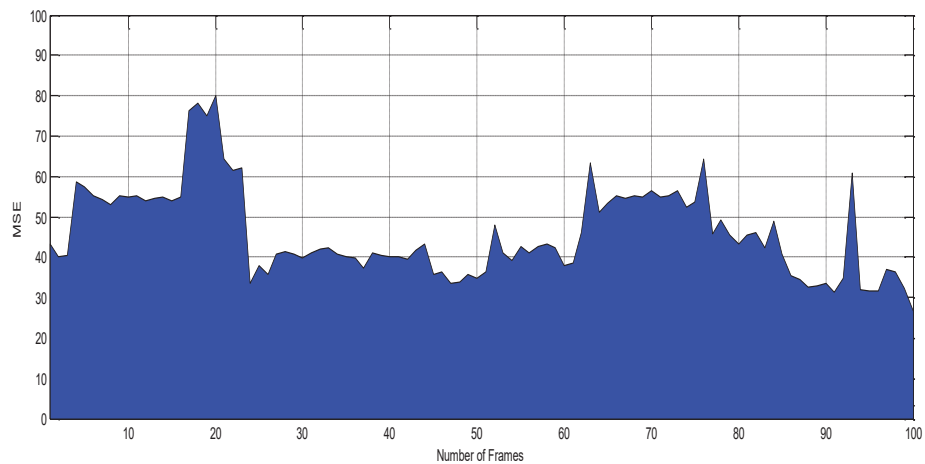


Figure 3.27: Images quality measured by mean squared error.

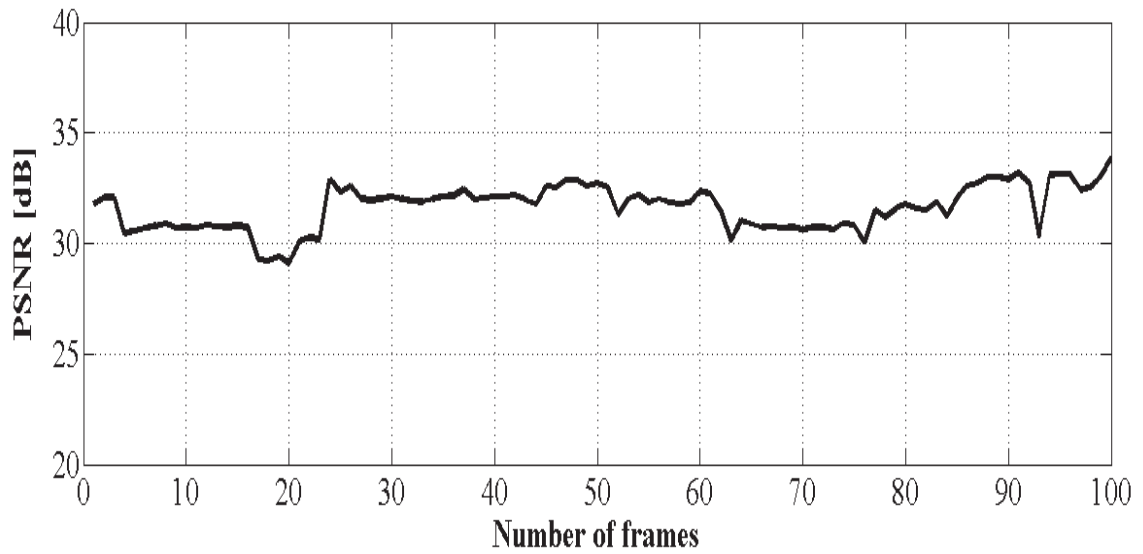


Figure 3.28: PSNR measured received image quality.

In order to ensure the performance of the system, during the experiments, we observed data collected from spectrum trace view analyzers which display in real time to visualize the wireless signals and detect sources of wireless interference that impact the performance. Figure 3.29 shows that the wireless signal was performing in fluctuation conditions.

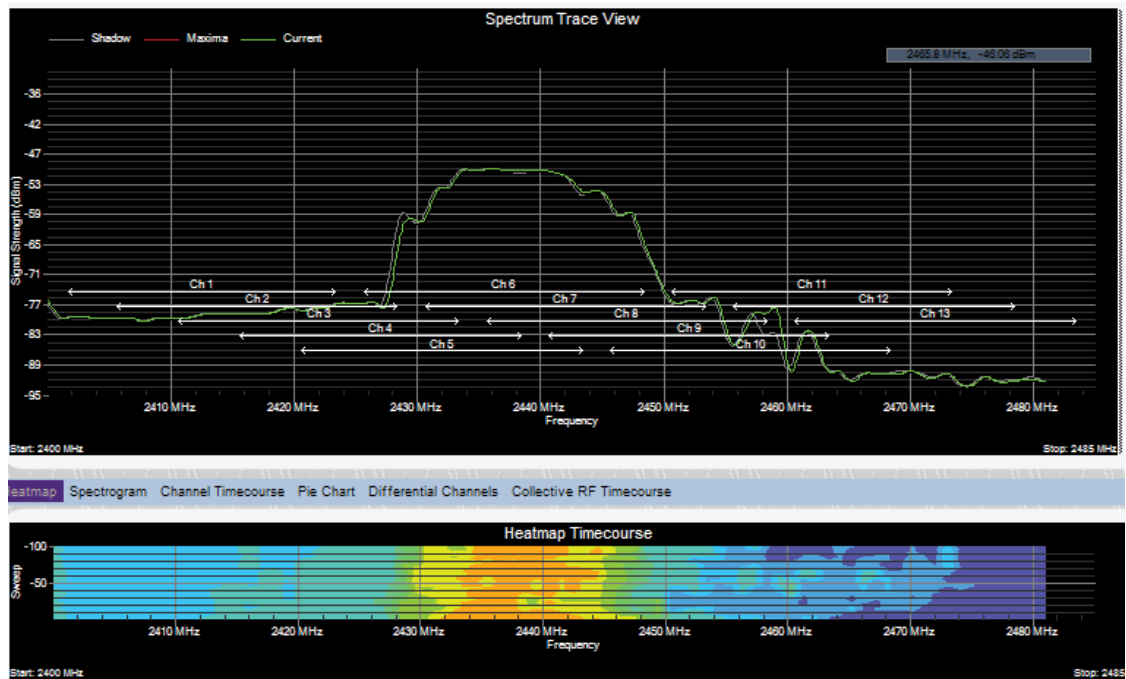


Figure 3.29: Wireless Network spectrum trace view.

In terms of network streaming performance, the experimental results show that streaming speed is varied during the period of streaming. It can be seen from Figure 3.30 that the highest streaming rate is up to 250 Kilobytes per second (KBps).

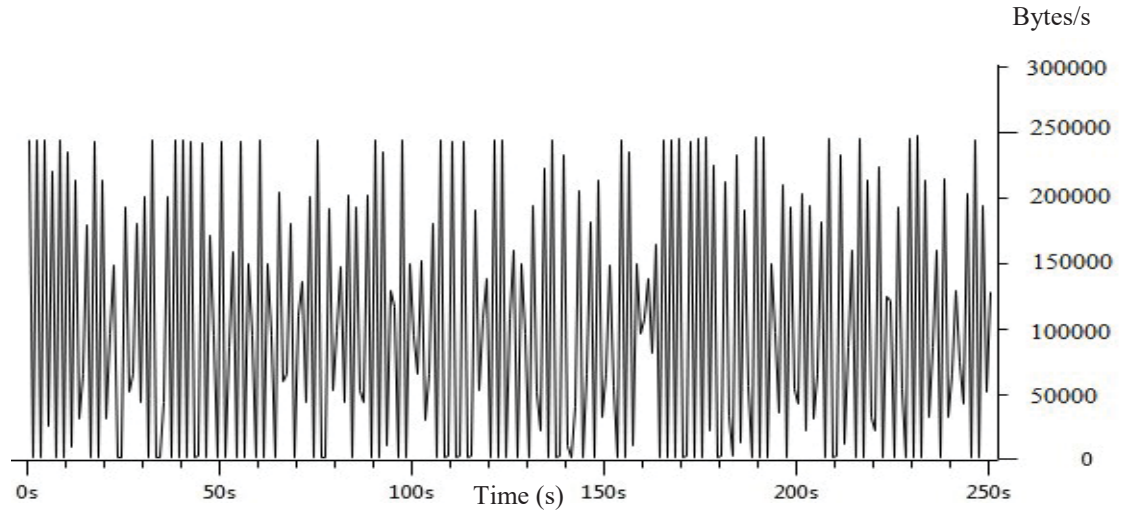


Figure 3.30: Streaming rates (Bytes/s) vs. Time (s).

Table 3.5: Average PSNR Results for Various Distances.

Trial Times	Distance [m]	Access Point	Average MSE	Average PSNR [dB]
1	5.00±1.00	1	4.98	43.79
2	10.00±1.00	1	4.68	43.34
3	15.00±1.00	1	4.47	43.72
4	20.00±1.00	1	5.45	42.14
5	25.00±1.00	1	5.78	41.61
6	30.00±1.00	1	7.88	39.95
7	35.00±1.00	1	18.19	38.85
8	40.00±1.00	2	17.10	37.07
9	45.00±1.00	2	50.97	31.64
10	50.00±1.00	2	82.58	29.78
Mean	27.50±1.00	1.30	20.21	39.19
Minimum	5.00±1.00	1.00	4.47	29.78
Maximum	50.00±1.00	2.00	82.58	43.79

Table 3.5 lists the results of wireless streaming with various distances from the client to the wheelchair which was connected to the closest wireless access point (AP). It is clearly seen from Table 3.5 that the average PSNR is reduced when the wheelchair moves far away from the wireless access point. The average PSNR of the above ten experiments is 39.19 dB. In comparison with the previous study (Oleksii 2012; Xiaolin, Cheng & Zixiang 2001; Xu et al. 2015; Yuwei, Deng & Nowostawski 2013), it is worth noting that the quality of the image is excellent for $\text{PSNR} \geq 37$ dB and is good for $32 \text{ dB} \leq \text{PSNR} < 37$ dB. The results from Table 3.5 show that the proposed scheme obtains high-quality images for the transmission distances of less than 35m from one AP. The wheelchair can stream in a longer distance with two wireless access points in the same network as illustrated in Figure 3.31. However, the wheelchair is mobile, and the wireless link between nodes is rapidly changing over time. Due to the handover from Wi-Fi, the wheelchair has to disconnect with one AP and then reconnect with another stronger signal AP. As reselection wireless access point may result in reduced performance, we have implemented an algorithm for automatic network reconnection to overcome this challenge.

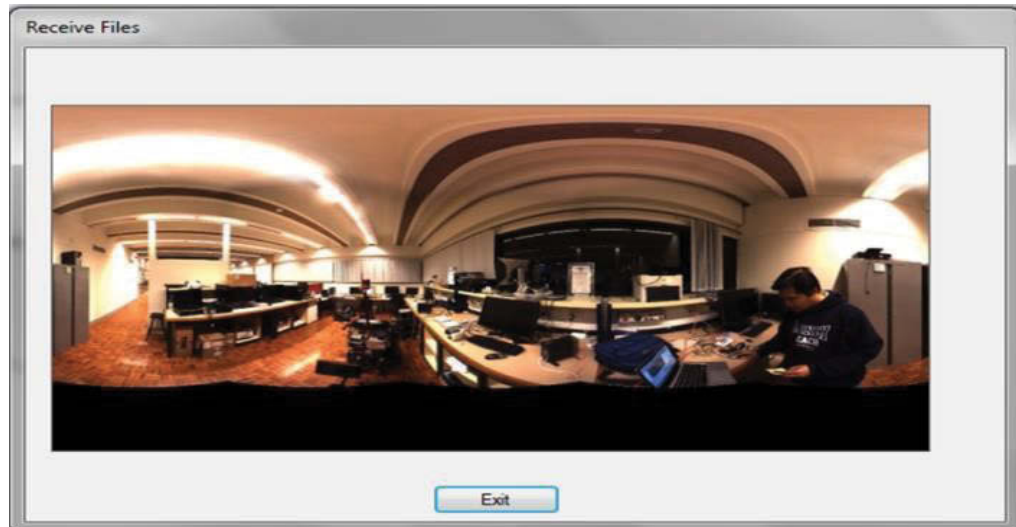


Figure 3.31: Client receives JPEG images from Server.

From these experiments, it can be concluded that different fluctuation of the wireless signal has an impact on performance when links experience data loss. The test results showed that streaming JPEG over a wireless network is feasible. However, there are still challenges to explore to ensure the available bandwidth allows efficient streaming for the telepresence wheelchair system.

3.6 Discussion

In this chapter, we have presented the architecture of the proposed system, the standard of IEEE 802.11 based wireless video transmission and how the interference happens when transmitting packets in a wireless network. The JPEG-based video coding techniques and the study on video streaming over a wireless network involve both video coding and wireless transmission. The challenges of video codec and the suitable physical layer of the wireless transmission are also explored for video streaming in real-time. This chapter also presented a comprehensive evaluation of the video streaming framework for the proposed system. The experiments to implement to stream video images over the wireless network and measure PSNR at the receiver side are also provided.

The state-of-the-art technology for high-resolution panoramic image generation, compression, and streaming over wireless networks in real-time was designed and implemented. The experimental results obtained on the wheelchair in a practical environment demonstrated the effectiveness of the images processing techniques. The subjective quality assessments show that the received images were smooth during the streaming period. The experimental results revealed that streaming speed is up to 250 KBps which means the system streamed two frames per second with resolutions of 1024x512 pixels. The results from the experiments conducted have confirmed that JPEG streaming from a wheelchair over the wireless network is feasible. However, the video streaming using JPEG video coding and standard transport protocol are still facing many challenges when delivering high-quality video streaming over the wireless network.

This study has considered rate control mechanisms that are implemented at the sender and buffer control at the receiver. The corporation of rate control and buffering helps to some extent to overcome short-term drops in transmission rate. However, there are some disadvantages. First, retransmission leads to the additional delay corresponding roughly to the round-trip-time between receiver sender-receiver. Second, the need to store multiple copies of the same media incurs higher storage cost, processing time and limiting its ability to adapt to varying transmission rates.

The main contributions of this chapter are to provide a good understanding of images processing and video transmission over wireless networks in real-time. The design and implementation of the telepresence wheelchair system provide an architecture and concept for remote assistance for people with disabilities. In addition, the evaluation of the system performance and the findings contribute to the wireless streaming, telepresence system, and real-time application development. Experiments have shown the feasibility of the prototype system. However, it can be observed from the results of Section 3.5.3 that the fulfillment of the performance of the panoramic image streaming from a telepresence wheelchair is acceptable but could still be improved. There are still interesting challenges to explore in streaming panoramic images over a wireless network in terms of streaming speed rates. Future research should attempt to seek out other solutions to enhance the streaming speed rate of the system.

From the technical perspective, apart from handling the enormous amount of images extractions and compressions, the timing requirements for all components of programming, the synchronization of buffer processing and data streaming is the greatest challenge. In our work, we assumed that the available bandwidth changes during the transmission which may lead to poor performance. However, the performance of the system can be protected from the fluctuations in the available bandwidth by keeping the target bitrate much lower than the available bandwidth. We should monitor the available bandwidth during the transmission. We did not investigate how the system should adapt to the fluctuation bandwidth.

In the next chapter, research on telepresence system based on the protocol of a cross-platform real-time application development and multiple video streaming are discussed to enable the approach to focus on enhancing video streaming over the wireless network.

Chapter 4

A Telepresence Wheelchair Based on Ubiquitous Platforms and Advanced Wireless Mobile Networks

4.1 Introduction

In Chapter 3, the concept and approach to the design of a telepresence wheelchair system were presented and analyzed. The study has proven the possibility of video streaming from the wheelchair to a remote location. The developed telepresence wheelchair based on client-server topology and a standard Internet transport protocol over a wireless local area network is feasible. However, the experimental results presented in the previous chapter (Section 3.5.3) have shown a few limitations. The video streaming performance of the telepresence wheelchair can be affected by several factors, such as the radio signal fluctuation and slow streaming speed rate which lead to the inefficiency of the overall video streaming during the movement of the telepresence system. The network conditions and the transport protocols play a major role in determining the performance of the developed methodology. As a result, it is necessary to develop a streaming approach for dealing with the streaming speed and outdoor streaming environments. Therefore, more protocols, advanced mobile wireless networks and approach strategies need to be explored in order to improve the overall performance of the whole system.

This chapter will focus on discussing a *cross-platform framework* for video streaming to enhance the performance of the telepresence wheelchair system. First, the ubiquitous platform which can be developed to stream video and transfer the navigation signal efficiently for the wheelchair is explored. Second, the *multiple video streaming* approach is investigated which allows the system to be capable of multiple video streaming from the

wheelchair to a remote location in real-time for safety control. After that, a combination of multiple video streaming and the mobile wireless network is proposed to implement the system in a real-time operating in an outdoor environment. Finally, remote control experiences of a telepresence wheelchair based on *virtual reality technology* is explored to encourage the user in using and interacting with the telepresence wheelchair.

Despite the fact that various streaming techniques have been implemented in the field of telepresence robots, almost all studies have attempted to develop telepresence robots based on existing video conference applications. For example, the early reports in the area of telepresence which developed telepresence robots based on cross-platforms were presented in (Tatsuno et al. 2010b) and (Corke, Findlater & Murphy 2012). Also, the efficacy of these platforms has been demonstrated in the telepresence robot design (Denojean-Mairet et al. 2014a). In contrast to all the approaches as mentioned earlier, the research in (Khoswanto 2014) presented a brief method with open protocol framework for telepresence robots. However, there has been limited work in developing such a framework for controlling the mobility of the wheelchairs from a remote location, although such meaningful systems are useful for people with disabilities and it would benefit them greatly if they could remotely control the wheelchairs.

It is worth noting that the vast majority of existing telepresence robots use one camera for video communication. Therefore, a single view is available only at any one time. In order to address these drawbacks and to have a wider view around the wheelchair at the same time, an alternative solution would be to use a multi-camera for a telepresence wheelchair. This work allows a remote user to predict the real-world target and process the action in real-time for safety purposes. Additionally, the original concept of the telepresence system was developed for a meeting rather than an assistive technology for people with disabilities. In order to fill the gap, this chapter will present various approaches for the telepresence wheelchair to create a useful assistive system that improves the quality of life of people with disabilities.

The main objective of this chapter is to propose a practical approach in which the integration of cross-platform computing, mobile wireless networking, and image processing

into a telepresence wheelchair can markedly improve the performance of the system. The approach implicitly exploits technological advancement in telecommunication, automation control, and healthcare systems. In terms of multiple video streaming, this chapter will present the development and implementation of appropriate techniques for a telepresence wheelchair system based on multi-video streaming of the real world surrounding a wheelchair to transmit to a remote location in real-time. Moreover, we highlight an interesting remote control methodology which allows a user to be able to control the wheelchair from a long distance.

Our key contributions demonstrate the feasibility of a telepresence wheelchair for face-to-face communication and efficient remote control via a *cellular network* medium. In particular, we have developed appropriate software for two-way video interaction and telepresence wheelchair remote control based on the Skype framework using mobile wireless networks in outdoor environments. The remote user can navigate the wheelchair via the Internet. At the same time, the remote user can hear the sound and see the surrounding areas of the wheelchair with the *full field of view* in real-time. Technical performance measurements are conducted to collect transmission data for the evaluation of the system performance. Then, the video streaming performance, the round-trip time, and the remote control tasks are also assessed to evaluate the feasibility and efficiency of the designed system. The contributions of this chapter have been partly published in the author's conference articles (Ha, Nguyen & Nguyen 2016a, 2016b).

The rest of this chapter is organized as follows. Section 4.2 is devoted to a practical framework for video streaming based on a ubiquitous platform which can be developed for the telepresence wheelchair. Then, Section 4.3 presents the advanced wireless mobile networks for video transmission. The designed system and the proposed methodology of multiple videos streaming and controlling are described in Section 4.4. Section 4.5 introduces *virtual reality* and *immersive experience* of the telepresence wheelchair system. Experiments and results of video streaming including performance evaluation and remote controlling are given in Section 4.6. Finally, this chapter ends with Section 4.7 which presents a discussion and summarizes the contributions of this chapter.

4.2 Streaming Speed Enhancement based on Ubiquitous Platforms

Skype has changed the world with its free voice-over-IP service, proving high-quality two-way video communication over the Internet (Skype 2015). Skype offers free video conference services which allow Skype users to establish two-way audio, video streaming with each other and exchange information in real-time. The typical Skype architecture can be used to establish a video call between two users. Skype Client A and Skype Client B in a peer-to-peer networking are illustrated in Figure 4.1. For communication, Client A starts off by logging into Skype. The Skype server authenticates Client A and logs Client A in. Then Client A wants to call Client B, so she or he initiates a Skype video call, and the connection is direct from Client A's computer to Client B's computer via the Internet. If a Skype client is behind a network address translation (NAT) router or firewall, the Skype client cannot establish a direct connection to another peer. In such situations, the node peers act as relaying agents to help Skype peers behind firewalls or NAT routers establish connections to other peers. By connecting to nearby nodes, Skype reduces utilization and decreases the latency in response times, thus providing a fast and scalable communication network. Skype also recommends the firewall or Internet gateway support IP packet fragmentation and reassembly. Fragmenting the packets allows the stream of data to be broken into smaller packets that can be sent simultaneously over multiple ports to the destination. This packet fragmentation can dramatically improve the quality and performance of the video transmission by allowing higher throughput, which results in the more efficient usage of the bandwidth.

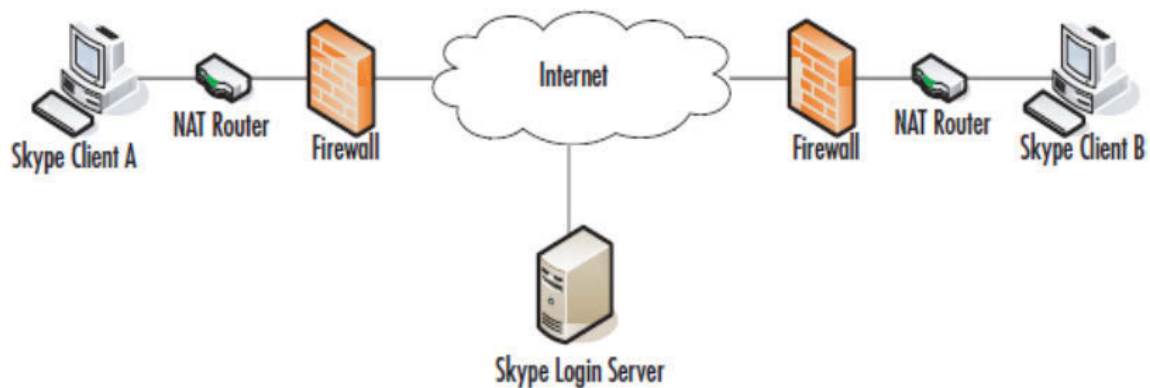


Figure 4.1: A typical Skype architecture.

Regarding transport protocol, Skype uses User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) to communicate with other Skype clients. UDP is primarily used to establish connectivity and perform global directory searches. Skype determines the best method for communication via the firewall or NAT router using various UDP mechanisms. If no UDP ports are open, Skype will attempt to use TCP port 80, then TCP Port 443. Once the connection is created, the clients can communicate and transfer data. TCP requires a three-way handshake to verify that data reaches its destination, whereas UDP just sends that data, and does not require acknowledgment of delivery. Because UDP does not require all of the overhead in the message structure, the messages are smaller, and UDP headers are always the same size. The UDP message structure allows better video call quality and faster data transfers.

Despite its popularity, little information is known about Skype's encrypted protocols and proprietary networks. However, Skype provides a well-known cross-platform in which users can develop their applications by using an Application Programming Interface (API). It is an ideal framework for constructing an inexpensive system. For that reason, a telepresence wheelchair based on this ubiquitous framework has been developed in this work. Our approach is adapted to the particular application in the field of health care. In this study, a proposed telepresence wheelchair based on Skype framework is illustrated in Figure 4.2.

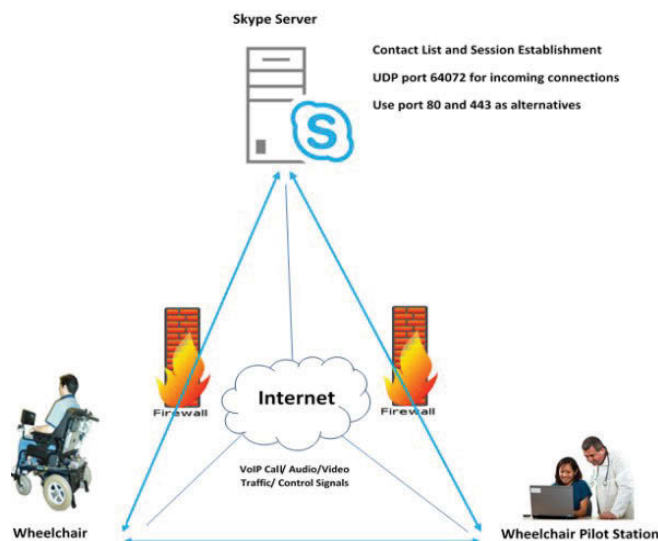


Figure 4.2: A telepresence wheelchair based on Skype architecture.

4.3 Advanced Wireless Mobile Networks

Much research has been done in the field of telepresence. Telepresence involves many technologies such as robotics, computer, and telecommunication systems. However, current telepresence works have been limited in the area of robotic applications used only in indoor environments. Moreover, mobile wireless network technology has grown rapidly over the last few years with emerging applications and services. Particularly, the fourth generation (4G) cellular network has acted as a bridge between telecommunication technology and daily life applications. In this section, we present an investigation into a telepresence wheelchair in outdoor environments by employing cellular network infrastructures instead of using local wireless networks in indoor environments.

The development of the mobile wireless network has revolutionized the transfer of information, particularly the fourth generation mobile network (4G) or Long-Term Evolution (LTE). The LTE is a standard for wireless communication of high-speed data for mobile phones and data terminals which have been developed by the 3rd Generation Partnership Project (3GPP) (Siwach & Esmailpour 2014). In theory, 4G cellular networks increase the capacity and speed using a different radio interface together with core network improvements with the peak of download speeds up to 100 Mbps and 50 Mbps upload. The LTE network architecture diagram is shown in Figure 4.3. Figure 4.3 illustrates the connection paths from the end user equipment (UE) to the evolved base stations (eNodeB). Each eNodeB is a base station that communicates with and controls the mobiles. The mobility management is responsible for choosing the serving gateway (S-GW). S-GW acts as a router, and exchanges data between the base station and the packet data network gateway (P-GW), and communicates with the outside Internet or the IP networks.

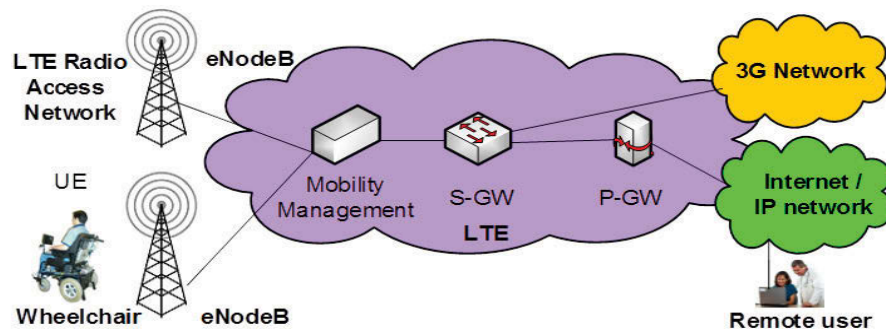


Figure 4.3: A simplified LTE cellular network architecture.

In this study, a mobile broadband data 4G modem is plugged in the wheelchair and acts as a UE for telecommunication interfaces. Therefore, the telepresence wheelchair can establish connections to the Internet via the telecommunication infrastructures for transferring video and control signals.

The telepresence wheelchair requires good quality video streaming and stable communication links between geographically distributed users. Understanding the network performance is important for effectively designing and analyzing a telepresence system. The control processing capabilities of the wheelchair rely on the transmission delays and other factors. In particular, due to mobility issues and radio conditions, fluctuation signals and the lack of bandwidth cause latency and jitter which can impact the overall system performance. Latency or round-trip time (RTT) refers to the amount of time it takes a bit from transmission to reception and back again. Large RTTs result in system connection time-out. Jitter causes packets to arrive at their destination with different timing. RTT and jitter calculation has been presented in (Kreher & Gaenger 2010). By denoting the corresponding event times as Ts_i and Tr_i for sending and receiving time respectively (for packet ID = i), the RTT_i is the difference between the event times given by

$$RTT_i = Tr_i - Ts_i \quad (4.1)$$

For jitter calculation, expression (4.2) is used. In this formula, the individual latency measurement samples for two subsequently received packets of the same stream are defined as RTT_i , RTT_j and the difference of these two latency measurement results is defined as $J(j, i)$:

$$J_1 = 0, \quad J_2 = (2,1)$$

$$J_i = J_{i-1} + \frac{(|J(i, i-1)| - J_{i-1})}{16} \Big|_{i=3}^{i=\infty} \quad (4.2)$$

The factor of 16 in the denominator of (4.2) accounts for a smoothing factor proposed in the standard RFC 1889 (Kreher & Gaenger 2010).

4.4 Telepresence with Multiple Cameras

Our goal is to develop a telepresence wheelchair which is capable of providing video conferences and interaction from an unknown starting location. The telepresence wheelchair platform that we have developed is shown in Figure 4.4. This designed system is expected to be useful in remote monitoring and collaborative control to provide assistance and enhance safety mobility for people with disabilities. In order to achieve the desired features of providing video conferences and remote controllability from a long distance, the powered wheelchair is equipped with a Mac mini computer, multimedia components, control interfaces, telecommunication interfaces. These devices are connected to process, control and transmit video around the wheelchair while it is moving to a remote location over the wireless network in real-time.

For the purpose of the ability to use video conferences as a function of telepresence, our powered wheelchair is equipped with a touchscreen display which is used for two-way video communication. Moreover, to have a wider view of an unknown environment surrounding the wheelchair from a remote site, along with a number of accessories for wheelchair platform, a 7-port USB extension hub is used to extend the connections for the vision system. *Four cameras* are also mounted in four directions at an angle of 90 degrees in front of the wheelchair to capture images surrounding the wheelchair, as can be observed from Figure 4.4.

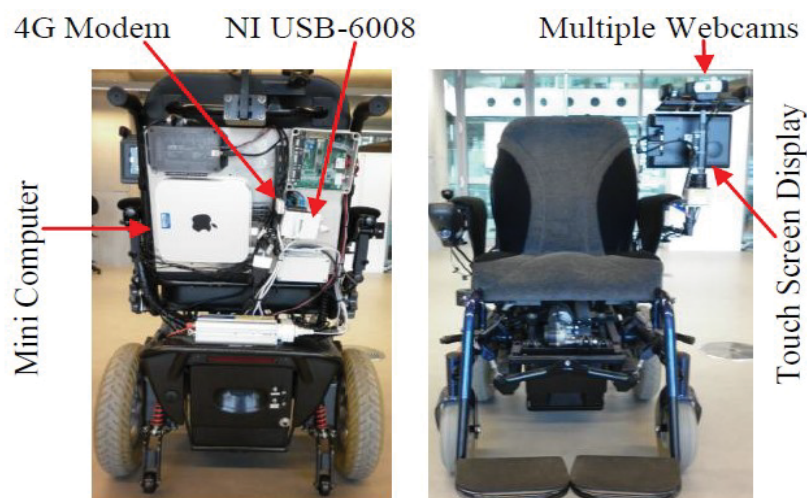


Figure 4.4: The back and front view of the telepresence wheelchair.

In this work, the process of multi-cam streaming is described in Figure 4.5. The four cameras namely cam#1, 2, 3, 4 are connected to a computer. Live video frames are sent from these cameras to the central processing unit (CPU) of the computer to process. Cameras are actual hardware devices that communicate with Skype through the CPU. In the Windows operating system, a filter can read and write requests to Skype. In fact, it multiplexes the video source from the physical cameras. It passes video data from one or more physical cameras to Skype by appropriately transforming the data. Skype will stream incoming frames from a live video source with a reference frame.

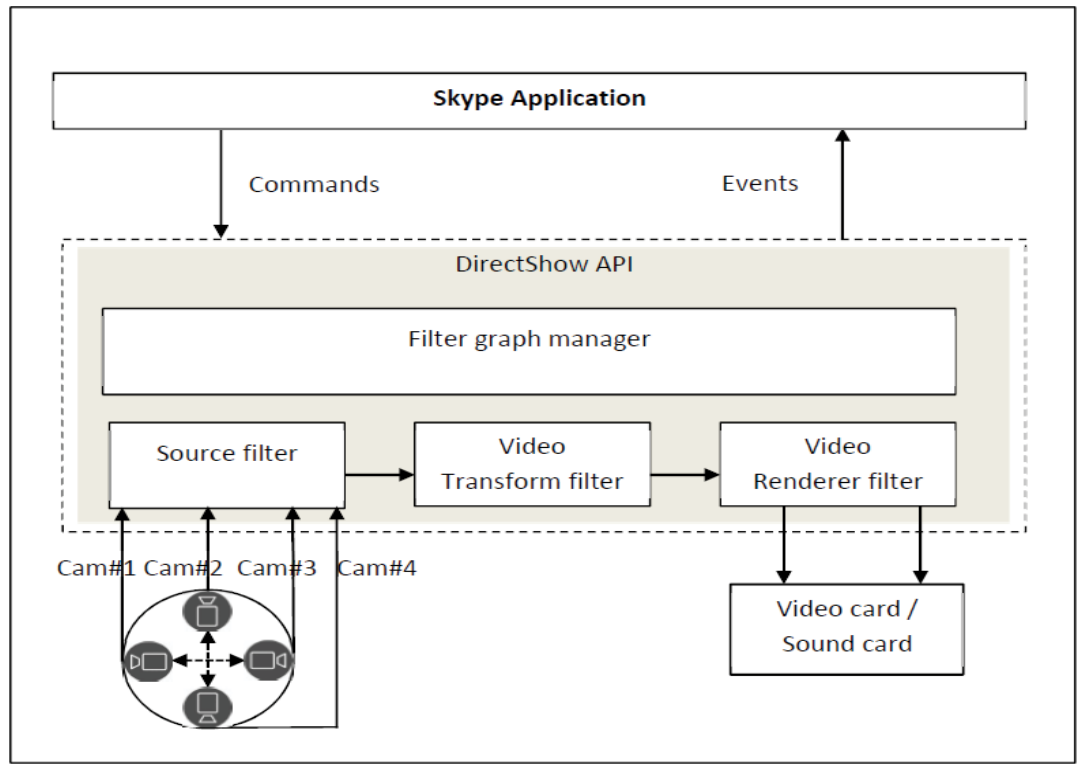


Figure 4.5: The diagram of a multi-video streaming based on Skype framework.

Our system is different from the existing telepresence robots designed with one camera based on Skype framework (Corke, Findlater & Murphy 2012), (Denojean-Mairet et al. 2014a). Here, our cameras and the telepresence technology were applied to the wheelchair for safety health care purposes while navigating the wheelchair. We developed a third-party application to stream multi-video in order to achieve a wider visualization at the remote site. To this end, four Logitech C930 webcams were connected to a computer via a USB extension hub.

In this study, an application module for multi-video acquisition and processing was developed and plugged into the Skype framework. This module can access, perform video processing, and display multimedia streams from ubiquitous cameras. Multi-cam is a free and open source and can be developed from Microsoft DirectShow (MacCormick 2013). This technique is based on the Component Object Model (COM) technology. COM objects in DirectShow are called filters. A typical DirectShow application comprises three kinds of filters: source, transform, and render. They are connected to process a media stream from the source to render by a filter graph manager. The filters need to be synchronized in order to keep media samples in synchronization with the media streaming process.

To be more specific, source filters support multimedia stream acquisition from digital cameras. Once the media stream has been captured, DirectShow filters can be used to transform it. Transform filters have been written to resize video images. Media streams can also be multiplexed together, taking two or more streams and making them one synchronized video. Transform filters act on the acquired multimedia stream and alter it in some way. After the transform filters have been implemented, the final task is to render the media stream to the display, speakers, or a device (Pesce 2003).

From the technology perspective, real-time communication is the technical challenge for such a telepresence system. The data information must be transferred wirelessly in real-time to guarantee that the system can execute the commands in real-time to avoid risk and make it safer with desired actions. In order to evaluate the performance of a telepresence wheelchair system, the data transmission between the remote user and the wheelchair must be considered in the wireless network. To ensure the system performs well and is suitable for real-time communication, the system demands some standard quality of video streaming and response time. The network processing delay would affect the interactivity of the data communication. There are many existing methods and techniques approaches to evaluate a real-time system in the telecommunication and computer network society. The criteria to assess the system performance in existing reports have shown that the frame rate of video streaming and round-trip time are the key parameters taken into account to evaluate the real-time system (ETSI 2006; ITU-T-G.1010 2001).

By observing multiple views around the wheelchair from the multi-video data, a remote user can send an appropriate control command to the telepresence wheelchair. To be able to control the telepresence wheelchair moving in different directions, the telepresence wheelchair system is designed to have two front freewheels and two rear driving wheels. The two rear wheels are connected and controlled by direct current motors. A hardware connection diagram of the wheelchair control components is shown in Figure 4.6. It consists of an add-on DX power component and a virtual joystick module. Instead of using the joystick, the computer generates appropriate control signals following the calculations of assistive navigation software which is integrated into Skype as a third-party application programming interface (API).

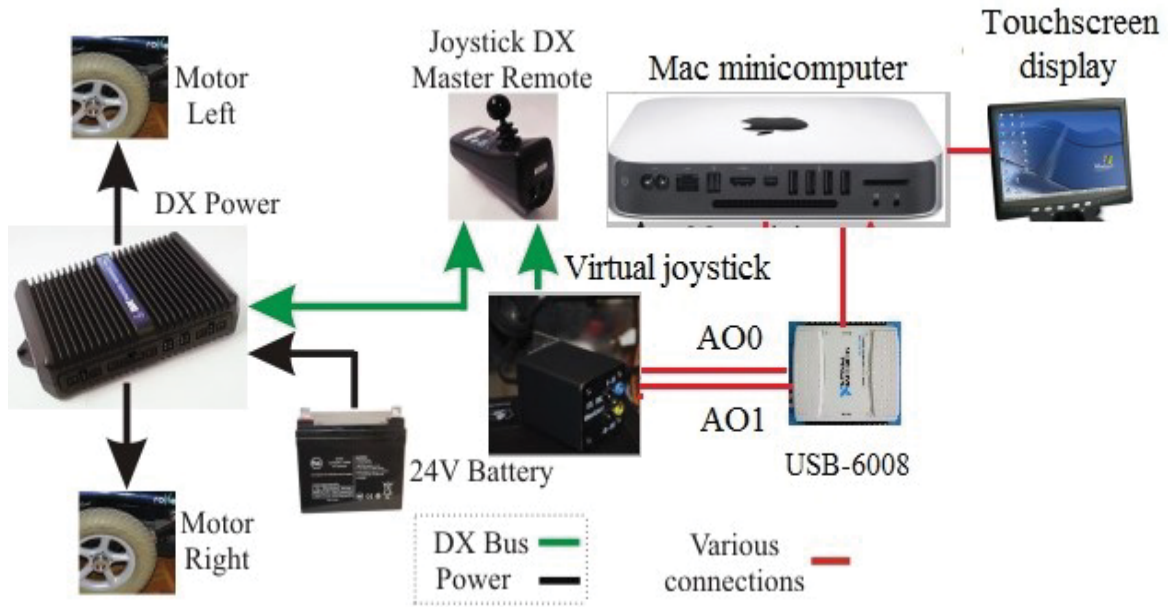


Figure 4.6: The connections of the wheelchair control components.

In order to handle the movement of the wheelchair, the National Instrument (NI) device which is a multifunction data acquisition NI USB-6008 with a USB interface is connected to the minicomputer at the back of the wheelchair. The NI USB-6008 has 8 analog inputs and 2 analog outputs which allow the system to transform command tasks to control the wheelchair. Particularly, an NI USB-6008 is connected on the back of the wheelchair to transform movement commands from the remote user to control the wheelchair. NI USB-6008 is controlled by an application that can be plugged into the Skype framework. To

control the movements of the telepresence wheelchair, five states, namely go forward, turn left, turn right, go backward and stop, have been designed. When the remote user sends commands, the API of the wheelchair will receive them and then transform them into appropriate moving states. These digital signals from the computer will be converted into the analog forms and transferred to the virtual joystick module via the analog output pins of the NI USB – 6008 to control the wheelchair. In practice, as the digital values are obtained, they are converted into the appropriate voltage values and then fed into the virtual joystick module. Define that u_1 and u_2 are the input voltages, v and ω stand for the output velocity and the angular output velocity of the wheelchair, respectively. The range of the speed v of the vehicle is 0 to 1 m/s, then when $v=0$, the signal value for input voltage u_1 sent to the AO0 of the module is expected to be 2.5 V and if $v=1$ m/s, u_1 should be 3.77 V. The relationship between u_1 and v can be determined by

$$u_1 = 2.5 + 1.27v \text{ (V)} \quad (4.3)$$

Similarly, the signal u_2 , which is fed into the pin AO1 of the module can be calculated based on the value of the rotational velocity or turning speed ω in radian per second by

$$u_2 = 2.5 + 1.3 \frac{2\omega}{\pi} \text{ (V)} \quad (4.4)$$

According to (4.3), the wheelchair velocity v can be expressed as

$$v = \frac{1}{1.27}(u_1 - 2.5) \text{ (m/s)} \quad (4.5)$$

It follows from (4.4) that the turning speed ω is given by

$$\omega = \frac{\pi}{2.6}(u_2 - 2.5) \text{ (rad/s)} \quad (4.6)$$

In our proposed approach, the speed signal is approximately $2.5V \pm 1.27V$. The 2.5V value is defined as the stop signal. The forward signal with the speed of 1 m/s is 3.77V, and the backward signal is 1.24V. The rotational signal range is about $2.5V \pm 1.38V$. The right turn signal is greater than 2.5V and the values less than 2.5V are represented as the left turn state of the wheelchair.

4.5 Telepresence Wheelchair and Virtual Reality

Virtual reality technology has revolutionized the way people interact and communicate. One of the most remarkable aspects of virtual reality (VR) is that it provides the user an experience of being virtually present in that environment and feels realistic. The VR headset allows a user to step into the immersive environment and look in any direction for interaction (Bolas et al. 2013; Mathur 2015; Sharma, Rajeev & Devearux 2015). To date, the most notable virtual reality devices are the Oculus Rift headset (Oculus 2016) and Samsung Gear VR (Samsung 2016). The former one works with the computer. The latter is compatible with Samsung Smartphones. The design of Oculus Rift and Samsung Gear VR are shown in Figure 4.7. Neither are standalone devices. Therefore, they must be connected to a computer or a smartphone. On the other hand, 3D video based on stereo video signals and eyeglasses are currently being used for 3D cinema and many other fields. Producing 3D video based on stereo systems was mentioned in (Jaju, Banerji & Pal 2013; Muller, Merkle & Wiegand 2011) in which the authors presented an approach that was able to generate stereoscopic displays at the receiver by two input cameras.

Likewise, in our work, a combination of VR and *dual-camera* is made to blend virtual reality and real life 3D video for augmented reality. This hybrid form aims at increasing the immersion of the user in the realistic environments. In order to generate the 3D video, we have developed a post-processing program to multiplex the two video sources from the cameras and, then, stream them with a reference frame. At the receiver, the video streams will be de-multiplexed to display on the screen. We have considered a combination with the dual-camera stream to achieve two slightly different scenes of views with depth information onto the two eyes of the viewer. This approach results in 3D visualization and immersive representation through a VR headset. The telepresence wheelchair using virtual reality devices is as shown in Figure 4.8



Figure 4.7: The front and the back view of the Oculus Rift and Gear VR.

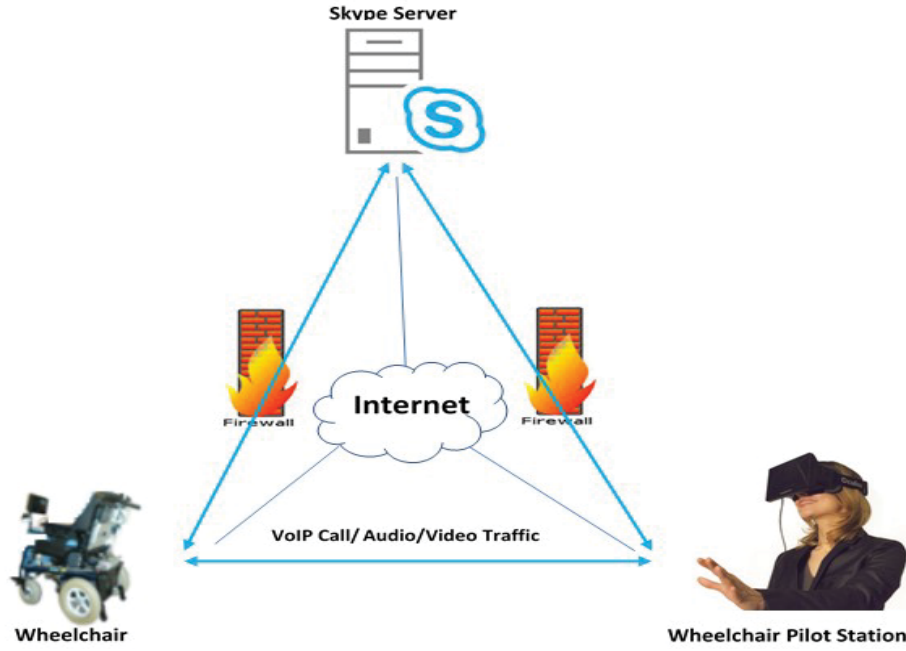


Figure 4.8: A remote user is wearing virtual reality device and control the wheelchair.

Quality assessment of video is important for improving user satisfaction, and quality evaluation methods for 3D video have been presented to the video processing community. The commonly used parameters are the frame rate (ETSI 2006; Kodera et al. 2014) and the peak signal-to-noise-ratio (PSNR) (Gurler et al. 2011; Joveluro et al. 2010; Oh et al. 2009). Frame rate refers to the number of individual frames that an imaging device displays per second, as measured by frames per second (fps). The maximum frame rate allowable for each of the cameras depends on the resolution of the images and the bandwidth (BW) of the network. The frame rate can be roughly approximated using the following formula from (4.7) (assuming all cameras are set with the same resolution).

$$FPS = \frac{BandWidth}{\frac{(Pixel\ per\ Frame \times Bytes\ per\ Pixel)}{Number\ of\ Camera}} \quad (4.7)$$

Peak Signal-to-Noise Ratio (PSNR) is typically used as a performance metric to measure the quality of the image when it is transmitted over channels. The PSNR is usually expressed in terms of the logarithmic decibel scale. In our model, the PSNR reveals the quality of the generated 3D views and 3D perception. The higher the PSNR is, the closer the received image is to the original one. Theoretically, the rendering PSNR can be calculated from the obtained mean squared error (MSE) and the structural similarity index

measure (SIMM). Assume that n is the number of pixels in an n -dimensional image vector. x_i and y_i denote the i -th pixel of the original image vector $\mathbf{x} = \{x_i | i = 1, 2, \dots, n\}$ and the reconstructed image $\mathbf{y} = \{y_i | i = 1, 2, \dots, n\}$, respectively. Then, the MSE is defined as the average squared distance between the image vectors and can be expressed mathematically as

$$MSE(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2. \quad (4.8)$$

Then, the PSNR is given by

$$PSNR = 10 \times \log_{10} \left(\frac{MAX^2}{MSE(\mathbf{x}, \mathbf{y})} \right), \quad (4.9)$$

where MAX is the maximum possible pixel value of the image, for example, MAX is 255 for 8-bit images.

There is a close relationship between the PSNR and the SSIM. The SSIM quality assessment index is a multiplicative combination of the three terms, namely the luminance comparison, the contrast comparison and the structural comparison. For original and reconstructed images \mathbf{x} and \mathbf{y} , respectively, the SSIM index is defined as

$$SSIM(\mathbf{x}, \mathbf{y}) = [l(\mathbf{x}, \mathbf{y})]^\alpha [c(\mathbf{x}, \mathbf{y})]^\beta [s(\mathbf{x}, \mathbf{y})]^\gamma, \quad (4.10)$$

where the relative significance of each component is controlled by the parameters $\alpha > 0$, $\beta > 0$ and $\gamma > 0$, respectively. The luminance, contrast, and structural components are respectively defined as

$$l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \quad (4.11)$$

$$c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \quad (4.12)$$

$$s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \quad (4.13)$$

where μ_x and μ_y are the means of the original and reconstructed images, respectively. σ_x

and σ_y are the standard deviations of the images and σ_{xy} is the covariance of two images x , y . The constant C_1 is included to avoid the numerical instability when $\mu_x^2 + \mu_y^2$ is close to zero. It is obvious that $SSIM(x, y) = 1$ if and only if $x = y$, i.e., two images are identical. For the simplification of calculation $\alpha = \beta = \gamma = 1$ (the default for Exponents), and $C_3 = \frac{C_2}{2}$ (the default selection of C_3) are taken here. From (4.10), (4.11), (4.12) and (4.13), SSIM index simplifies to

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \quad (4.14)$$

Note that the MSE in equation (4.8) can be rewritten as

$$MSE(x, y) = \sigma_x^2 + \sigma_y^2 - 2\sigma_{xy} + (\mu_x + \mu_y)^2. \quad (4.15)$$

Then, the SSIM defined in equation (4.14) can be expressed as

$$\frac{1}{SSIM(x, y)} = \frac{MAX^2 \times \alpha(x, y) \times e^{-PSNR \times \ln(10)/10} + \beta(x, y)}{l(x, y)s(x, y)}, \quad (4.16)$$

$$\text{where } \alpha(x, y) = \frac{1}{2\sigma_x\sigma_y + C_2}, \quad \beta(x, y) = \frac{2\sigma_{xy} - (\mu_x - \mu_y)^2 + C_2}{2\sigma_x\sigma_y C_2}$$

Assuming that $C_2 \ll \sigma_x, \sigma_y$ and $C_3 \ll \sigma_x, \sigma_y$, the PSNR can be obtained by

$$PSNR = \log_{10} \left[\frac{2\sigma_{xy} (l(x, y) - SSIM(x, y))}{MAX^2 SSIM(x, y)} + \left(\frac{\mu_x - \mu_y}{MAX} \right)^2 \right] \quad (4.17)$$

Equation (4.17) reveals the relationship between the objective image performance metric (PSNR) and the human visual system-based image quality metric (SSIM). From a theoretical perspective, the acceptable PSNR for the images transmitted over wireless links is about 26 dB to 32 dB (Oleksii 2012; Xiaolin, Cheng & Zixiang 2001; Xu et al. 2015; Yuwei, Deng & Nowostawski 2013).

4.6 Experiments and Results

4.6.1 Experiment 1: Real-time Video Streaming with Multi-Camera for a Telepresence Wheelchair

A. Description

The first experiment was designed to test the effectiveness of the proposed framework. This experiment aims at comparing the video streaming performance of our telepresence wheelchair based on the *Skype framework* with those of the previous telepresence systems. This work demonstrates a new approach for the telepresence wheelchair equipped with *multiple cameras*. Multiple videos are streamed in real-time from an array of cameras mounted on the wheelchair, allowing wide visualization surrounding the wheelchair. The work explores the integration of the Internet of Things, such as multimedia, wireless Internet communication, and automation control techniques into a powered wheelchair system as an assistive system for the elderly and people with disabilities.

The experiments were carried out in four scenarios at various locations in which a remote user was far away from the wheelchair. Ten trials were conducted for each scenario to stream multi-video from four cameras with a resolution of 640x480 pixels per camera from the wheelchair to a remote client using the Internet through wireless local area network connections based on Skype framework. Figure 4.9 illustrates the test positions. A remote user was in room A and the wheelchair was in different rooms which were labeled room B, C, D, and the balcony E, respectively.

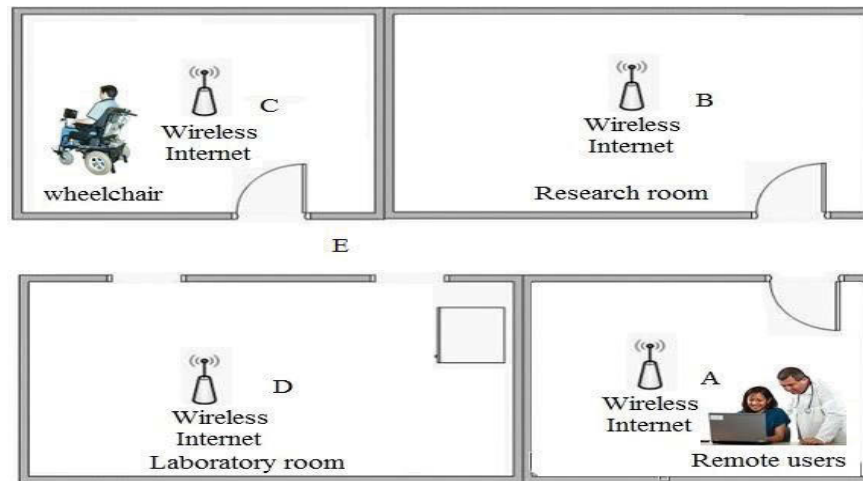


Figure 4.9: System experiments at the University of Technology Sydney.






In the experiments, to meet the functions of telepresence to monitor the environment of the wheelchair, the remote users were required to sit in room A and be connected to the wheelchair at various locations (room B, C, D, and E). The detailed information of the scenarios is listed in Table 4.1. The shortest distance from the remote user to a wheelchair is the scenario 4 with the distance of 20 ± 1 m. The longest distance is the scenario 2, with the distance of 60 ± 5 m, in which the remote user was in room A and the wheelchair located in room C. The wheelchair was connected and controlled by the remote user using the wireless local network infrastructure. Four wireless access points were set up for network connections.

Table 4.1: The Test Positions of the Experiments.

Experiments	Remote Position	Wheelchair Position	Distance (m)
Scenario 1	A	B	30.00 ± 5.00
Scenario 2	A	C	60.00 ± 5.00
Scenario 3	A	D	40.00 ± 5.00
Scenario 4	A	E	20.00 ± 1.00

To assess the performance of video streaming, a set of data was measured, collected and analyzed to evaluate the quality of the video streaming rate. In addition, an analysis of network data was recorded in detail, including the system information and the round-trip time. In this work, RTT is a measure of the consecutive transceiver data packets between the remote user and the wheelchair. Before doing the tests, the wheelchair battery was fully charged, and the wheelchair speed was set to 1 m/s to ensure that the wireless connection could be maintained. In order to fulfill the requirements of the remote assistance, a very user-friendly control interface application with five buttons was designed to send the command signals to Skype. These control signals are written and read by an API developed from C#. This application is able to connect with the USB 6008 for control processing. At the wheelchair side, these commands will be received and transmitted to the USB 6008 to drive the wheels in real-time. From the remote location, the user can establish a connection to the wheelchair by clicking a connect button and starting a multi-video conference. Then, the user can also control the wheelchair with the function of telepresence from five buttons. The explanation of the five buttons is listed in Table 4.2.

Table 4.2: Command buttons available during experiments.

Button	Command	Explanation
	Stop	Wheelchair halts
	Go Forward	Wheelchair moves directly forward
	Go Backward	Wheelchair moves directly backward
	Turn Left	Wheelchair keeps turning left until next command task
	Turn Right	Wheelchair keeps turning right until next command task

B. Results

The results of the experiments are described in Figures 4.10, 4.11 and 4.12. It can be observed from Figure 4.10 that the streaming rate in all scenarios fluctuates considerably between 25 and 30 fps. The scenario 1 peaks the highest average streaming rate up to 30 fps. On the other hand, the scenario 5 has the lowest average streaming rate of 25 fps. It is not surprising that the results of the scenario 5 are lower than the others due to the weak signals at the balcony areas where the wireless coverage is limited.

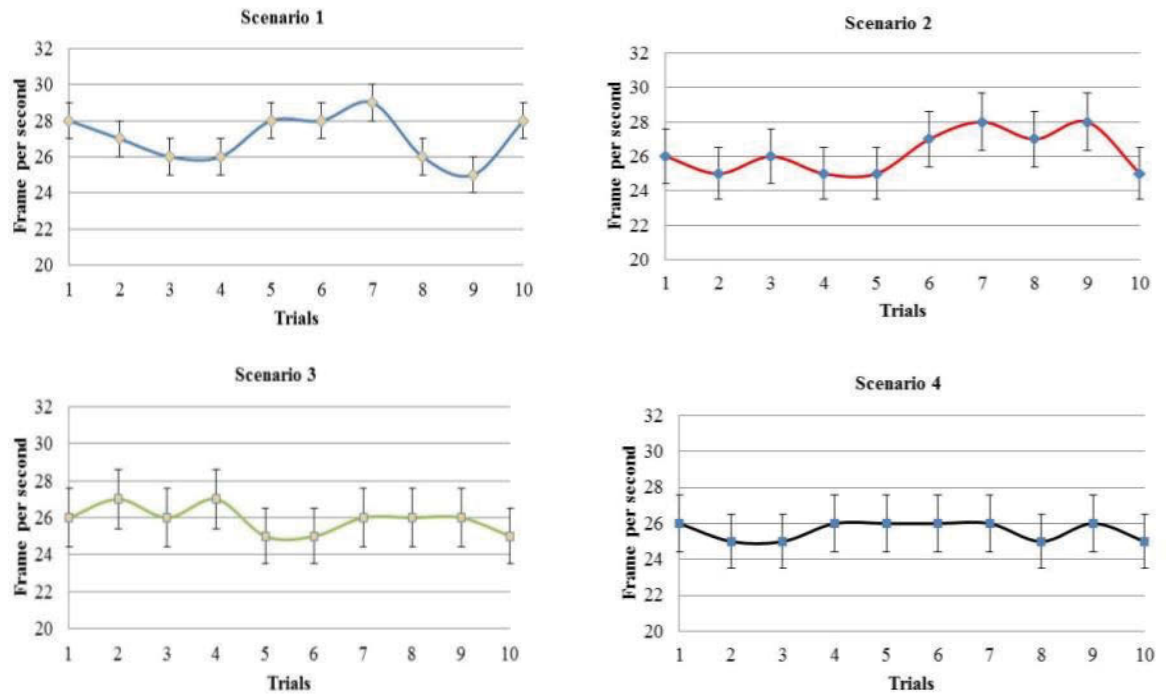


Figure 4.10: The average streaming rate of ten trials in the experiments.

In comparison with the reports in (ETSI 2006), it is worth noting that the standard video frame rate is 25 fps (Europe) or 30 fps (North America). Thus, the overall streaming rate performance of the system has reached the standard of real-time video communication. Furthermore, the system has higher performance than the previous similar work (Kodera et al. 2014) which presented an approach of the multi-view video stream with the frame rate of 15 fps in wireless streaming.

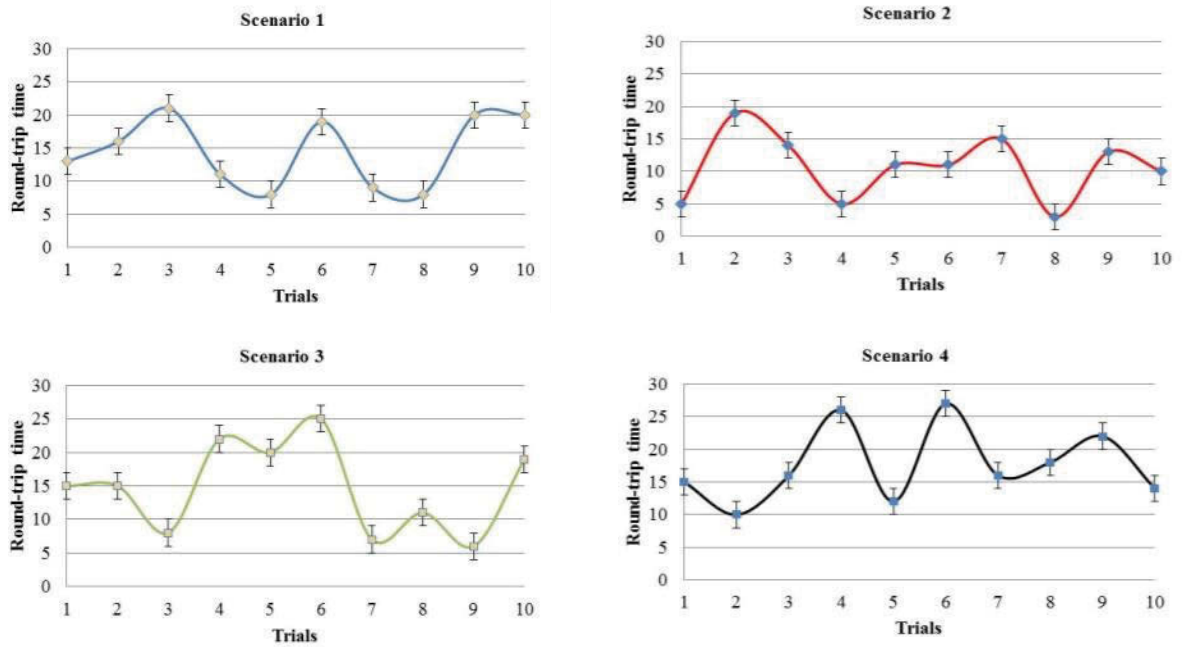


Figure 4.11: The average round-trip time of ten trials in the experiments.

Regarding the round-trip time, according to Telecommunication standardization sector of the International Telecommunication Union (ITU-T) recommendations for the quality of services and transmission performance (ITU-T-G.1010 2001), it is worth noting that the RTT of less than 400 milliseconds (ms) is considered as real-time multimedia communications. The results in Figure 4.11 illustrate the average round-trip time of ten trials in each scenario in the experiments. It can be observed from Figure 4.11 that the RTT fluctuates from 3 to 27 ms. Obviously, it is an excellent streaming quality under the RTT threshold value (ITU-T-G.1010 2001). In further comparison with findings in the recent previous telepresence robot study (Denojean-Mairet et al. 2014a), the authors presented that the telepresence robot with less than 40 milliseconds delay was considered a good performance. Thus, the results in Figure 4.11 have confirmed the efficacy of our approach for a real-time telepresence wheelchair system.



Figure 4.12: Multi-view video obtained from a telepresence wheelchair and the control interface at the remote site.

By obtaining the multi-view video, at the remote location, a remote user could hear and observe different angles of view around the wheelchair and feel as if he or she is present at the wheelchair. A remote user could navigate a wheelchair via the Internet through wireless connections in a distant location as can be seen in Figure 4.12.

Table 4.3: The average accuracy of command tasks.

Experiments	Upload BW [kBps]	Download BW [kBps]	Number of command task	Accuracy [%]
Scenario 1	156	80	50	100
Scenario 2	148	9	50	100
Scenario 3	135	40	50	100
Scenario 4	78	3	50	97
Average	129.2	33	50	99.25

The results from Table 4.3 show that the wheelchair can be controlled remotely in different types of bandwidth connections. The wheelchair can successfully navigate at various distances with the average accuracy of 99.25 %. During the experiments, it can be observed that there is a correlation between the fluctuations of wireless signal strength, the streaming rate, round-trip time and accuracy of command tasks due to the quality of the wireless signal. It may be improved by using more wireless access points and wireless repeaters to guarantee the signal strength. Nevertheless, these overall results indicate that the proposed method is promising for a full view display approach and would be more effective for telepresence wheelchair systems in real-time.

4.6.2 Experiment 2: A Telepresence Wheelchair System Using Cellular Network Infrastructure in Outdoor Environments

A. Description

The second experiment was implemented to evaluate the performance of the system at various locations when the telepresence wheelchair operates in outdoor environments. The aim of the experiments in this section is to demonstrate remote interaction and control from a long distance and across countries by using mobile wireless communication networks. A large amount of communication data based on real network measurements were collected and analyzed to evaluate the system performance.

The experiments were conducted with two case studies to evaluate the flexibility and accessibility of the wheelchair from everywhere in the real world environment. For each case, the same experimental procedures were repeated ten times. In the first case, experiments were carried out with internal connections in the same country. In this scenario, remote users and a wheelchair user would be in Australia. In the second scenario, the experiments were set up to connect and control the wheelchair with the functions of telepresence from a different country. The remote user was in another country (Vietnam), and the wheelchair user was in Australia. In both case studies, the wheelchair was located within the pedestrian walkway at Jones Street in Sydney, Australia. The system configurations of a remote user and wheelchair are listed in Table 4.4.

Table 4.4: Details of the computers used for implementation.

System information	Remote computer	Wheelchair computer
Model	MacBook Pro	Mac mini
CPU, DRAM	Intel Core i7, 8 GB	Intel Core i7, 8 GB
OS	Window 7 (64 bit)	Window 7 (64 bit)
Number of webcams	1	4
Skype	Version 7.17.0.105	Version 7.17.0.105
Wireless	Wi-Fi	4 G
Transport protocol	TCP/IP, UDP	TCP/IP, UDP
Movement speed	0	1m/s

Both of the computers had a similar configuration. The wheelchair computer was configured with a 4G modem on the Telstra mobile network in Sydney, New South Wales, Australia. The remote user used the wireless network for connections. In the setup of the experiment, a wheelchair user was required to sit in the wheelchair without doing anything and remote users connected, interacted and navigated the wheelchair. Once the connection is established, a remote user will be able to view live video around the wheelchair and communicate with a wheelchair user by using the 4G cellular wireless network. A wheelchair in an outdoor environment at Jones Street, Sydney, Australia and connection from a remote user in Vietnam to the wheelchair user in Sydney are illustrated in Figures 4.13a and 4.13b.

To analyze and evaluate our system performance, several measurements were taken at each site during the connection to measure the impact factors for ongoing connections. Four key performance parameters were taken into account, including signal strength, round-trip time, jitter and central processor unit usage. The corresponding outputs were measured for 5000 ms. This procedure was repeated ten times for each scenario. The results were used to analyze the performance of our system and made comparisons with previous studies. Wireshark was used to capture and analyze the network traffic (Orebaugh et al. 2006). Pings were simultaneously sent from the remote user to the wheelchair for RTT measurements, and the respective latency data was saved into a log file.



(a)



(b)

Figure 4.13: (a) Wheelchair in an outdoor environment at Jones Street – Sydney. (b) A connection from a remote user in Vietnam to the wheelchair user in Sydney.

B. Results

The results of the experiment are shown in Figure 4.14 which illustrates the remote user controlled the wheelchair from a long distance. It can be seen that the user at a remote side could observe the surroundings of the wheelchair in four view directions and could control the wheelchair by five command buttons.



Figure 4.14 Remote user controlled wheelchair from a long distance.

The measurement results are presented in Figures 4.15-4.18. Figure 4.15 shows wireless signal strength generated by measuring the signal while conducting the experiments of the two scenarios. Radio reception strength of the 4G modem on the computer of the wheelchair was measured in reference signal received power (RSRP) in decibel milliwatt (dBm) at a different time while the wheelchair was moving. The measurement results are described in Figure 4.15 in which RSRP varied from -101 to -93 dBm for the first scenario and from -107 to -93 dBm for the second scenario. In comparison with the report in (Ltd 2014), such mobile reception signal of the wheelchair was sufficient for a remote user to connect to the wheelchair and control the wheelchair with telepresence functions.

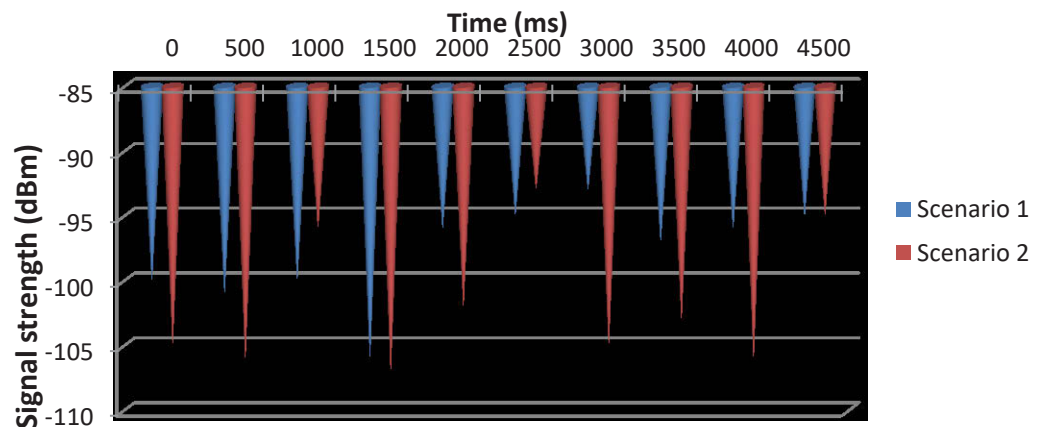


Figure 4.15: The RSRP measurements from the wheelchair of experiments.

Figure 4.16 presents the results for round-trip time, and jitter of the connections from the same country whereas Figure 4.17 shows the round-trip time and jitter of the connections from different countries. When analyzing the results in Figures 4.16 and 4.17, it can be observed that the average RTT and jitter measurement results for local and global connections are varied. It can be observed that round trip times fluctuated considerably between 226 and 271 milliseconds (ms) for local connections. RTTs of global connections were more than twice as high as that of local connections. In some cases, the global connection RTTs were in excess of 400 ms. It is worth noting that the round trip time of less than 400 ms is considered as real-time video conference (ITU-T-G.1010 2001). In comparison with a similar previous study (Terrazas Gonzalez et al. 2012), our proposed system has superior performance.

Regarding the jitter of real-time traffic, there are significant differences between the local and international connections. The minimum jitter in the local connection is 20 ms, and the lowest jitter in the global connection is under 60 ms. The results reveal the feasibility of using the 4G network for a telepresence wheelchair in healthcare applications. However, it has been recognized that different types of network connections can lead to different impacts on the quality of the results.

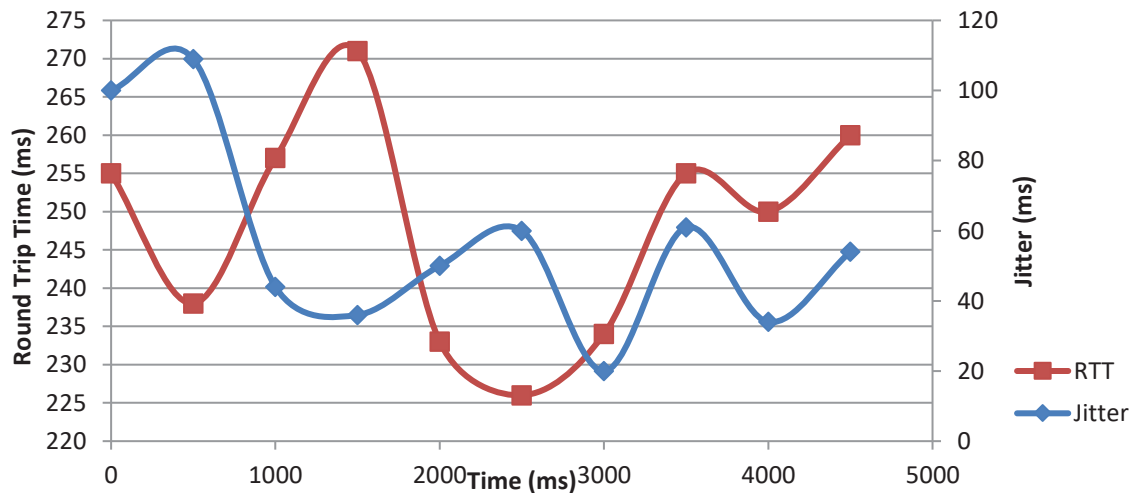


Figure 4.16: Round trip time vs. jitter over a period of 4500 milliseconds of same country links (Australia – Australia).

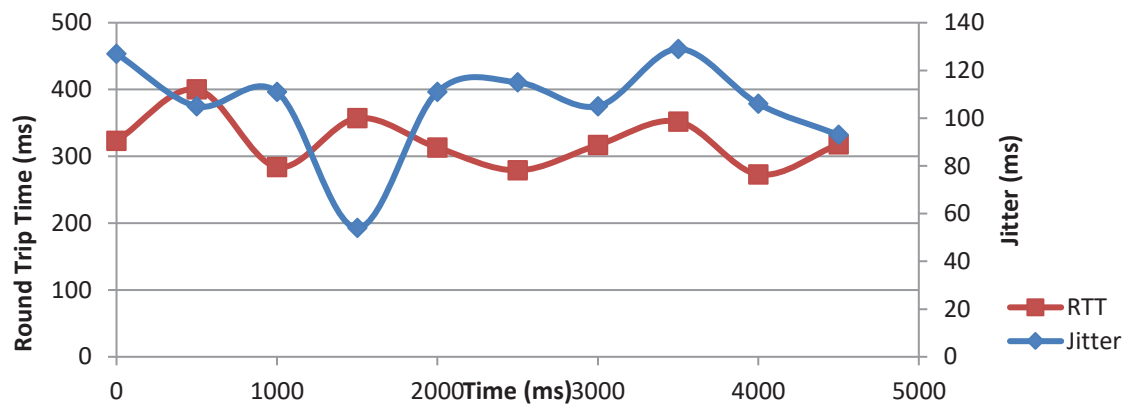


Figure 4.17 Round trip time vs. jitter over a period of 4500 milliseconds of different country connections (Vietnam – Australia).

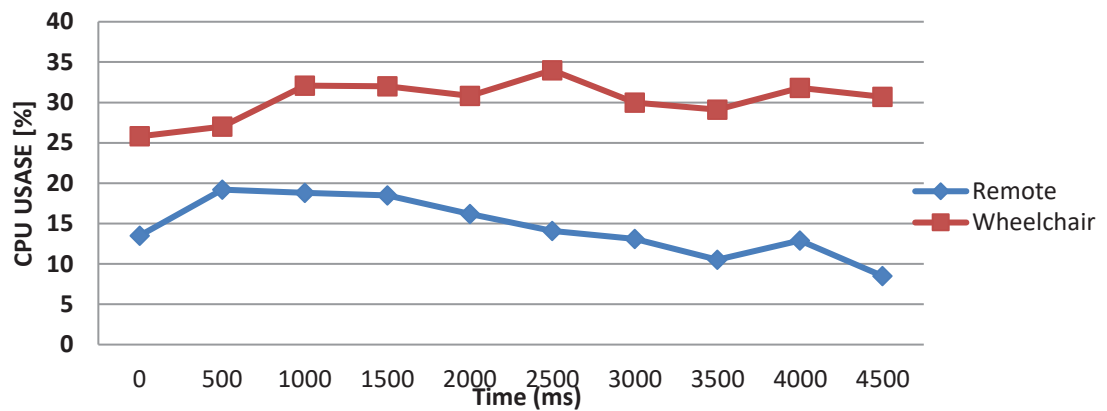


Figure 4.18 CPU usage of wheelchair computer and remote user's computer.

Moreover, during our experiments, we also found that the performance of the central processor unit (CPU) of the wheelchair computer and remote computer made a big difference in terms of CPU usage as shown in Figure 4.18. The usage of the wheelchair computer is nearly double the remote user computer due to control processing of the wheelchair computer.

Ultimately, all these parameters are correlated to the impact of mobility issues in system performance. There also is a correlation between the distance, radio reception strength, and delay. The data show greater latency in long distance connections than in local connections. While these findings may not come as a surprise to most, it can be enhanced by implementing diversity network optimizations in the 4G networks. However, this study's RTT is still under the theoretical RTT threshold value (ITU-T-G.1010 2001). It indicates that the latency impact of the cellular network is acceptable for remote users to control the wheelchair.

The second experiment has demonstrated the successful experimental deployment of the telepresence wheelchair in an outdoor environment associated with a certain scenario. It was observed that in some cases the cellular network links disconnected and reconnected due to signal loss, radio interference, and cell re-selection. Nevertheless, the experimental results obtained have demonstrated the potential of a telepresence wheelchair using 4G cellular networks. These positive outcomes may help the robotic and wheelchair community to make a significant investment in developing the telepresence wheelchair for healthcare.

4.6.3 Experiment 3: Telepresence Wheelchair and Virtual Reality

A. Description

In the preceding experiments, we have illustrated the feasibility and effectiveness of our telepresence wheelchair system. In this subsection, we present the experiments and results obtained from the real-world environments with the combination of *virtual reality technology*. Through an integration of multimedia, the wireless Internet, and wearable devices, the remote user could vividly experience the remote environment of the wheelchair. In this third experiment, the experiments were also done in two stages. The setup of the experiment is illustrated in Figure 4.19.

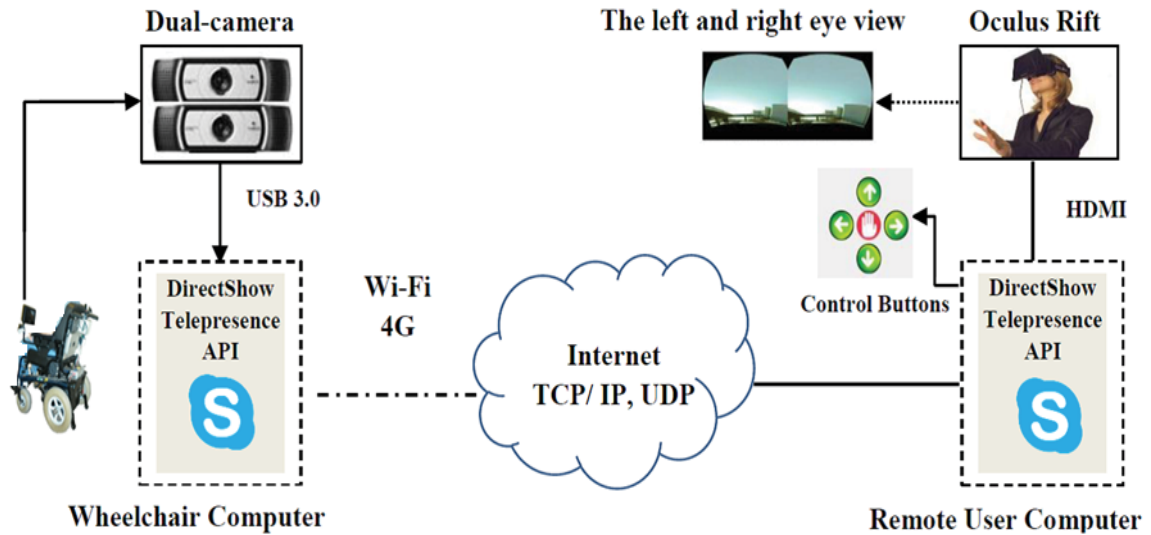


Figure 4.19: The diagram of the telepresence wheelchair with virtual reality in wireless connections.

In the first stage of evaluation, three experiments were conducted with the mobile wireless and wireless local area network connection to evaluate the feasibility of the system. The aim of these real-world experiments is to test the performance of *immersive experiences* and remote controllability of the wheelchair from a remote location over the Internet in real-time. Our experiments focused on evaluating the system performance and the feasibility of using the telepresence wheelchair for users with special needs. We conducted experiments to demonstrate the situations in which people with disability and older adults with impaired mobility want to move the wheelchair towards their position independently without any help from another person or caretaker.



Figure 4.20: A remote user was wearing Oculus Rift and controlling the wheelchair in outdoor environments.

In the second stage, the experiments were carried out in an indoor environment with participants involved to assess the system performance. In order to verify the system, we recruited volunteers to participate in the case studies. This study was approved by the Human Research Ethics Committee of the University of Technology Sydney. Ten participants aged from 23 to 40 years old, consisting of 6 able-bodied males (MS1-MS6) and four able-bodied females (FS1-FS4) were involved. The majority of the participants have the experience of using the Internet for more than five years while all participants did not have experience in using the virtual reality headset before. All volunteers were carefully informed of the purposes of the experiments, system operations, risks, and benefits before conducting the experiments. Each participant spent approximately one-hour running tests.

The experiments were carried out in an indoor scenario in the Centre for Health Technologies at the University of Technology Sydney, Australia. The study was conducted with 50 experiments (5 experiments per participant). The wheelchair was set up as shown in Figure 4.21 and a remote user was located in a different location at a distance away. In the experiments, to simulate real-life contexts, the participants who play roles as remote users were required to stay in another room. The wheelchair was outside the balcony areas where the wireless coverage was available. After establishing a connection between a remote user and a wheelchair, the remote user wore an Oculus Rift headset and controlled the wheelchair. By wearing the virtual reality headset, the participant was able to observe the depth information of three-dimensional video visualization surrounding the wheelchair inside the virtual reality headset. The remote user could see, hear and look at environments around the wheelchair and felt present at the wheelchair location. The remote users controlled the wheelchair from the starting point (A) through the obstacles (B, C, D) towards the ending point (E) with a total distance of 23 m as shown in Figure 4.21.

To be able to control the movements of the telepresence wheelchair in different desired directions, we designed user-friendly application programming interfaces (APIs) with five state buttons, namely go forward, turn left, turn right, go backward, and stop. When the remote user clicks on these buttons, control signals will be sent to the wheelchair, the application program interface of the wheelchair will receive them via the Internet connections, and then transform them into moving states to control the mobility of the wheelchair.

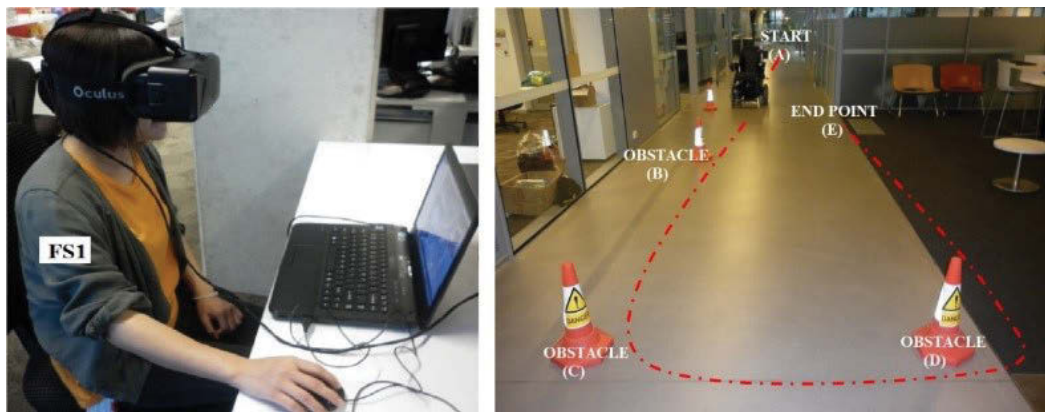


Figure 4.21: Participant (FS1) wore a virtual reality headset to observe the surrounding of the wheelchair and controlled the wheelchair to the desired locations from another room.

For further evaluation of remote control capabilities, we also conducted experiments to drive the wheelchair by using the joystick to make comparisons. The participants were required to sit down in the wheelchair and control the wheelchair by using a joystick from the starting point (A) through the obstacles (B, C, D) towards to the destination (E) as shown in Figure 4.22. The routes are the same as those in remote control over the Internet in Figure 4.21 with the functions of telepresence. The wheelchair speed in manual mode was also configured at 1m/s. Moreover, in order to validate the effectiveness of the function of telepresence to drive the wheelchair, in the experiments we recorded all the data of the pathway of the wheelchair in two control methods.



Figure 4.22: Ten participants in the experiments.

B. Results

As a consequence of the first stage evaluation, the telepresence wheelchair has been evaluated by measuring the video streaming and data transferring performance of the wireless network in real-time. The results of system performance evaluations are described in Figures 4.23 and 4.24. It can be observed from Figure 4.23 that the signal strength varies during the period of experiments due to the wheelchair movements. The results of the measurements show that the signal strength of the cellular network fluctuated in a range of -100 dBm to -85 dBm. Meanwhile, the wireless signal strength of the wireless signal is better than the mobile wireless signal which fluctuates from -76 dBm to -40 dBm. The closer the value is to 0, the stronger the wireless signal is.

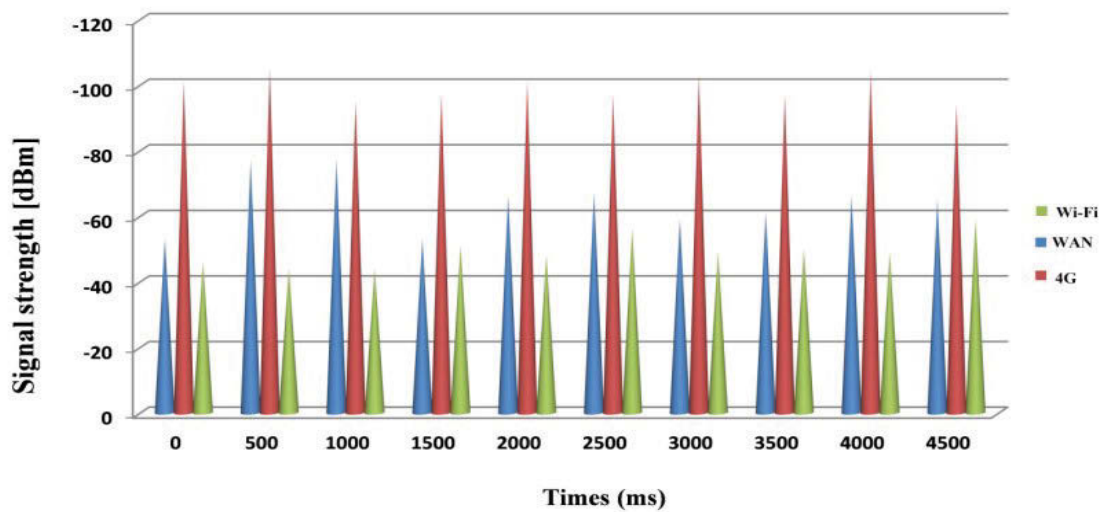


Figure 4.23: The signal strength of the wireless networks.

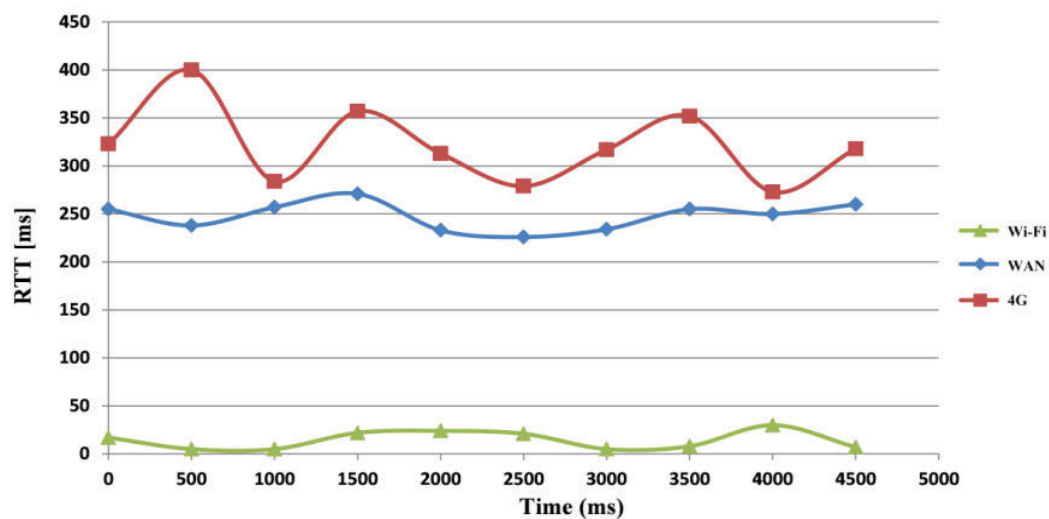


Figure 4.24: A comparison of the latency of different connections.

Also, as the telepresence wheelchair is connected with the different wireless connections, the performance of round-trip time may be different. The analysis results of the round-trip time of data transmission are shown in Figure 4.24. Figure 4.24 indicates that the RTTs fluctuate considerably between 5 and 30 milliseconds (ms) for Wi-Fi connections. RTT is in the range of 230-271 ms for local connections (WAN). RTT of 4G connections is nearly twice higher than local connections, which is in the range of 263-401 ms. After analyzing the response of the system from the results, it can be seen that the wireless network is more efficient than a mobile wireless network for video streaming and data transfer.

In comparison with the Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T-G.1010 2001), it is worth noting that the round-trip time of less than 400 ms is considered as a real-time video conference. It is obvious that there is a correlation between the distance, reception radio strength, and latency. The measured data shows greater latency in long distance connections than in local connections and Wi-Fi connections. However, this study's RTT is still under the acceptable RTT threshold value (ITU-T-G.1010 (2001)). Furthermore, in comparison with previous similar studies about telepresence via a cellular data network, the means RTT was reported in (Terrazas Gonzalez et al. (2012) about 545.45 ms which is greater than those of our proposed system.

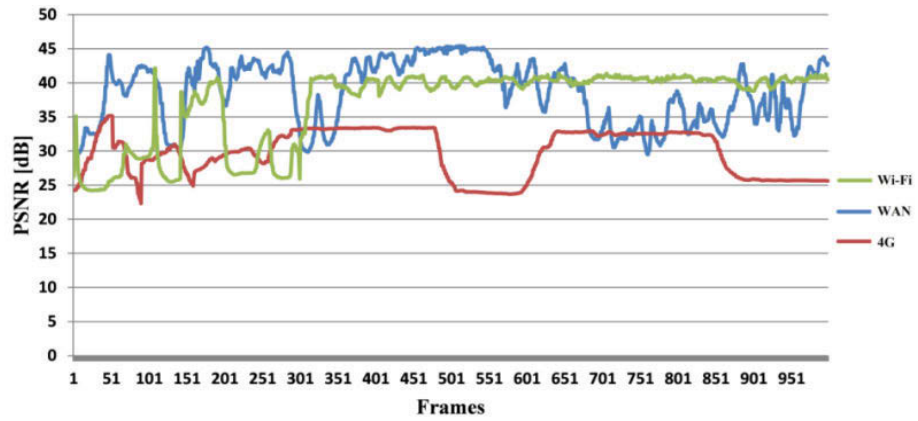


Figure 4.25: The measurement of PSNR of 1000 video frames.

Furthermore, in terms of 3D video quality evaluation, the objective evaluation was done by extracting frames from the source, and the received video. One thousand frames were extracted for samples analyzing in each scenario. The measurement results of PSNR are illustrated in Figure 4.25. It can be observed that there is a strong correlation between the type of connection and the quality of the video. The average PSNR of the whole sequence of 1000 frames for Wi-Fi, WAN, and 4G connections are 38.6, 36.9, and 29.7 dB, respectively. In comparison with the criteria assessment of previous studies of images and video quality evaluation (Oleksii 2012; Xiaolin, Cheng & Zixiang 2001; Xu et al. 2015; Yuwei, Deng & Nowostawski 2013), it is notable that the quality of the image is satisfactory for $26 \text{ dB} \leq \text{PSNR} \leq 32 \text{ dB}$ and good for $\text{PSNR} \geq 32 \text{ dB}$. Our overall results reveal that the proposed system has reached the high standards of 3D video quality and real-time performance.

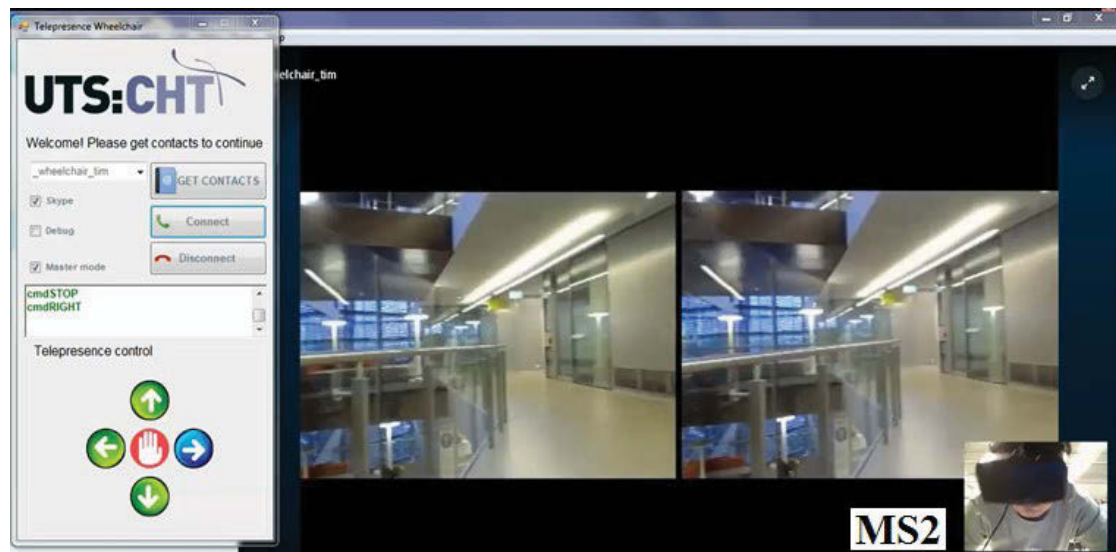


Figure 4.26: A remote user (MS2) was wearing Oculus Rift and controlling the wheelchair in indoor environments.

On the other hand, when the experimental studies were conducted with a participant in an indoor environment, the system allowed the user to observe the surroundings of the wheelchair in both left and right eye view. Figure 4.26 illustrates the video obtained at the remote side. The remote user was wearing Oculus Rift and controlling the wheelchair in outdoor environments.

Table 4.5 lists the average results of the experiments over 10 participants, with the average of the round-trip time, frame rate, peak signal to noise ratio, and accuracy of wheelchair control for all the experiments. The average round trip time for data streaming was 24.12 ± 1.60 ms which is very similar to the test result in Figure 4.24. The overall average frame rate of video streaming was 29.11 ± 0.52 fps, and the average of peak signal to noise ratio was 37.06 ± 0.67 dB. The results from Table 4.5 also show that the wheelchair can be controlled remotely. The wheelchair can be successfully navigated at various distances with the average accuracy of $97.72 \pm 0.91\%$. During the real-world experiments, it can be observed that the accuracy of the control task depended on the quality of the wireless signal. The system performance may be improved by using more wireless access points and wireless repeaters to guarantee the signal strength.

Table 4.5: Performance of the System with Ten Participants.

Participant	RTT (ms)	FPS (fps)	PSNR (dB)	Accuracy of control (%)
MS1	36.00±1.58	29.05±0.83	31.22±0.83	96.40±1.14
MS2	25.80±1.92	29.07±0.84	36.89±0.74	96.80±1.30
MS3	7.60±1.14	29.10±0.83	39.31±0.45	100.00±0.00
MS4	26.40±2.30	29.20±0.44	30.96±0.64	96.78±1.28
MS5	34.00±2.54	29.12±1.11	40.03±0.71	96.82±1.33
MS6	41.20±1.30	29.23±0.54	38.96±0.65	96.55±1.25
FS1	27.60±1.14	29.15±0.05	39.80±0.45	100.00±0.00
FS2	13.40±2.07	29.20±0.45	36.75±0.82	97.00±1.58
FS3	9.60±1.14	29.00±0.11	38.73±0.72	96.83±1.34
FS4	19.60±0.89	29.02±0.01	38.00±0.70	100.00±0.00
Mean	24.12±1.60	29.11±0.52	37.06±0.67	97.72±0.91

Additionally, the pathways of the wheelchair in two control methods are illustrated in Figure 4.27. It can be seen that the trajectories of the wheelchair in two approaches show an insignificant difference compared to the desired path. For example, there was a difference when the wheelchair moved from C to D since the wheelchair had to turn left.

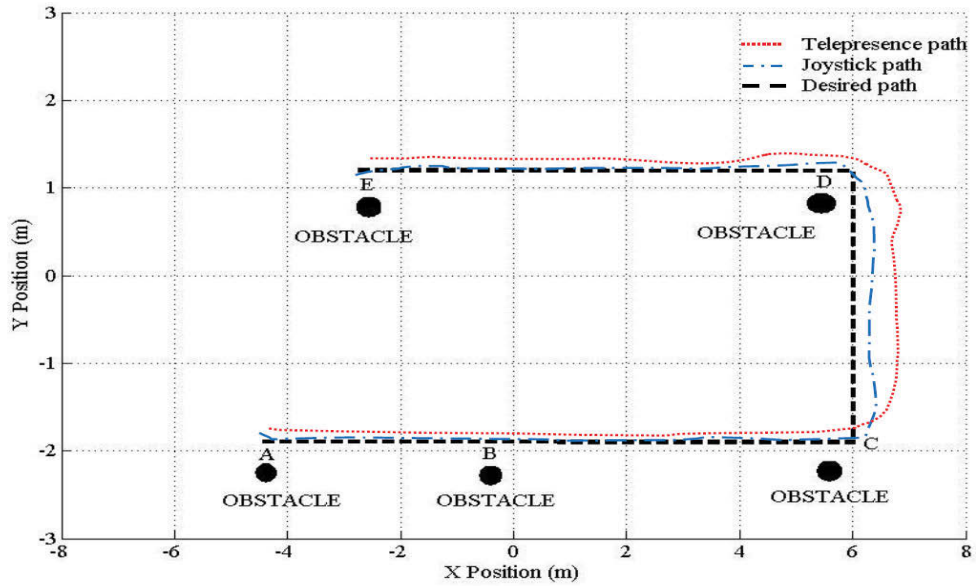


Figure 4.27: The trajectories of the telepresence and joystick control wheelchair in the experiments.

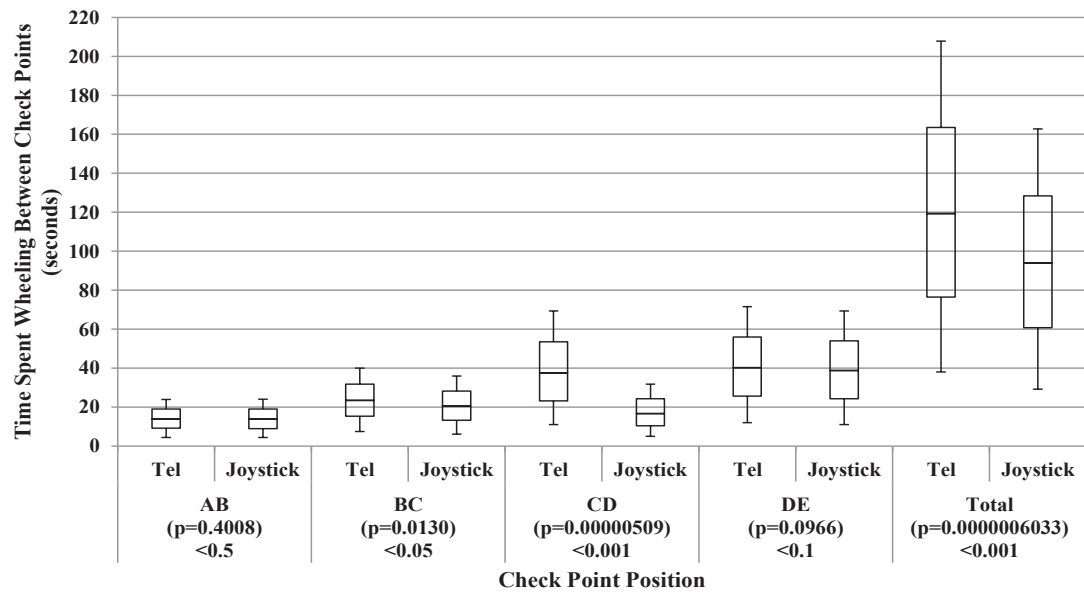


Figure 4.28: Time spent wheeling comparison between the manual control (Joystick) and remote control using telepresence (Tel).

Figure 4.28 also shows the time spent by ten remote wheelchair control subjects (Tel) and manual wheelchair control subjects (Joystick). Each bar represents the minimum, maximum and median value of time spent. It can be seen from the statistics that the amount of time spent on the remote telepresence control and joystick control of the ten subjects from point A to C is not much more different. It is also similar while moving the wheelchair straight forwards from B to C, and D to E. However, the remote telepresence control spent much time than the joystick control when driving from C to D. Overall, there was a statistically significant correlation $p < 0.001$ in both control methods to control the wheelchair in the entire pathway. The total time of remote control spent 10.23 seconds longer than joystick control for all of the participants. The maximum time spent to drive the wheelchair the whole path from A to E of telepresence and joystick control were 45.6, and 35.37 seconds, respectively. The result was understandable due to the time spent to make a decision while turning the wheelchair at point C and D. Nevertheless, the wheelchair could be controlled in the desired directions towards the target.

4.7 Discussion

The first experiment (real-time video streaming with multiple-camera based on the ubiquitous framework) effectively increases the quality of the connection between the remote user and the telepresence wheelchair system. The comparison results between the standardization, the previous studies and the responses of the wheelchair system show that the proposed framework is successfully implemented on the system. In addition, the second experiment (regarding a telepresence wheelchair system using cellular network infrastructure) demonstrated the successful experimental deployment of a telepresence wheelchair in an outdoor environment. It was observed that in some cases the cellular network links disconnected and reconnected due to signal loss, radio interference, and cell reselection. Nevertheless, the experimental results obtained have demonstrated the potential of a telepresence wheelchair using 4G cellular networks. The real-time experiments are conducted for estimating the effectiveness of the proposed approach. Finally, the third experiment (the telepresence wheelchair with virtual reality implementation) over the wireless connections was also conducted. The results show that the quality of the video and the performance of the system was at acceptable levels.

In this chapter, we have developed a telepresence wheelchair system. These works are developed upon our results published in (Ha, Nguyen & Nguyen 2016a, 2016b) in which the telepresence wheelchair system was implemented by using ubiquitous framework and multi-camera. The proposed method has demonstrated the feasibility of the deployment of the telepresence wheelchair and that the system is able to operate in both indoor and outdoor environments. The experimental results obtained from the Internet-enabled wheelchair in real-world environments showed the effectiveness of the combination of advanced technologies for a telepresence wheelchair and the feasibility of the designed system. The results indicate that the video is streamed smoothly and the wheelchair is efficiently controlled over the wireless Internet in real-time. The system allows a remote user to control the wheelchair and communicate with clients from a distant location through voice and video. The experimental results are positive and reliable, which shows potential for telepresence innovations in the health care domain.

One of the major contributions in this chapter is to implement a telepresence wheelchair based on the ubiquitous framework which increases the quality of streaming of the system. Although initially the framework was introduced in the field of telepresence robots, little has been done to implement this method for practical healthcare applications. The implementations and results presented in this chapter show that our technique, through the integration of technologies into a wheelchair platform, can effectively stream the video and control the wheelchair with an improved real-time performance.

Another significant contribution in this chapter is to provide an efficient video streaming and remote controlling method that relies on the ubiquitous framework. Although there have been some reports in the literature review, none of them is able to provide a detailed procedure for the telepresence wheelchair implementation in real-time and outdoor experimental results. Our results indicate that the telepresence system based on Skype framework and the mobile wireless network is able to adapt to the changes of the environment. In particular, it is able to recognize the environment type and provide the control that is appropriate to it. This provides a significant improvement for the assistive system which requires not only effective transmission but also satisfactory response time and quality of video streaming.

Until this point, the streaming and controlling problem had been approached. The performance of the telepresence wheelchair which has been developed based on the ubiquitous framework is improved significantly. However, a complete telepresence system based on the third party application with such framework is not the ultimate aim of this project. This work provides a more comprehensive understanding of real-time technology from which the system can be further developed. These positive outcomes may help the robotic and wheelchair community to make a significant investment in developing the telepresence wheelchair for healthcare. For the long-term development of telepresence wheelchairs, it would be better to develop an independent framework which can enable real-time streaming and control of the system.

In the next chapter, recent advances in web real-time communication of information technology will be explored for further development of the telepresence wheelchair.

Chapter 5

An Advanced Telepresence Wheelchair Based on WebRTC and 360-degree Video Communication

5.1 Introduction

In the preceding chapter, a telepresence wheelchair based on cross-platforms framework has been explored and has demonstrated the improvement of video streaming performance. The experimental results in the previous chapter pointed out that the prior attempts have achieved some degree of success in software and hardware development. The users can communicate and interact with the telepresence wheelchair with video communication and remote control in real-time from long distances in different environmental conditions. However, the limitations of the previously developed telepresence wheelchair are cumbersome tasks of additional software installations, the complexity of processing procedures, and handling time of third-party plugins to communicate with Skype. Moreover, the protocols of the existing platform are unpublicized. There still exist challenges for improving and dealing with the development of an independent assistive system. The telepresence wheelchair based on third party platform development does not seem to be an ideal target for a complete system, but its findings are beneficial to move to the next level. Therefore, it is highly desirable to develop an independent application to extend the advanced functionality of the system.

In this chapter, we present our research on an innovative approach to the real-time system. The previously proposed telepresence wheelchair system will be expanded and developed further by integrating Web Real-time Communication (WebRTC) and a dual-fisheye camera. The goal of the proposed development is to enhance the performance of the system

and improve a wide field of view surrounding the wheelchair to provide a safe wheelchair navigation. The investigation aims at providing an efficient way for assisted-living for people with disabilities. Specifically, this chapter introduces a telepresence wheelchair using web real-time communication technology with the purpose of enhancing video streaming and control signal transmissions in real-time. Based on the advantages of the emerging communication technologies, including the ability to stream video and data communication in real-time, a telepresence wheelchair system is developed for independent improvement to deal with the additional functionalities of the telepresence system.

The proposed system has been designed and developed by deploying Web Real-time Communication (WebRTC) and the JavaScript with Hyper Text Markup Language 5 (HTML5). To improve the video quality transmission, the adaptive rate control algorithm for video transmission is invoked. Furthermore, an efficient technique for streaming a full view of the real-world surrounding a wheelchair is proposed. The telepresence wheelchair which is capable of covering the whole scene with the wider field of view using a dual-fisheye camera is implemented. In addition, to improve the control reliability and quality of video transmission, we develop an estimation model for video quality of experience (QoE) based on neural networks. With the estimated QoE, the end users can estimate the transmission channel conditions such that they can navigate the telepresence wheelchair towards the coverage with strong wireless signals. Finally, experiments were carried out in real-world environments, and the wheelchair was controlled from a distance using the Internet browser, and results were assessed to evaluate the feasibility and efficiency of the designed system. A portion of the contributions in this chapter has also been published in the author's conference article (Ha, Chai & Nguyen 2017).

This chapter is organized as follows. After the introduction, the advanced information technology in Web Real-time Communication, including its architecture and practical implementation will be presented in Section 5.2. Section 5.3 describes an enhancing telepresence wheelchair with a full field of vision. The implementation of the combinations of the WebRTC and dual-fisheye camera of the proposed telepresence wheelchair approach is provided. Section 5.4 presents a QoE estimation model based on neural networks. Then, four experiments to evaluate the system performance will be presented in Section 5.5. The

first experiment is to demonstrate that the system using the WebRTC framework is able to make the most appropriate real-time communication. Then, the test is implemented with dual-fisheye video streaming to evaluate the performance of the 360-degree vision telepresence wheelchair system by quantitative measurements. The third experiment is to evaluate the performance of the QoE estimation model. The last experiment is to evaluate the reliability of the remote control telepresence wheelchair over WebRTC and 360-degree vision. Finally, a discussion can be found in the last part of this chapter.

5.2 Real-time Communication

5.2.1 Advanced Video Streaming over WebRTC

In the development of the emerging information technology, a so-called Web Real-time Communication (WebRTC) has offered a powerful technology that facilitates two-way video communication between browsers and has provided an efficient way for real-time communications (Atwah et al. 2015). WebRTC is a standard that defines a set of communication protocols and provides a free open source software package for developing real-time applications without the need of plugins or software installation.

Some research about the protocols and methods for developing real-time applications has been started since the first version of WebRTC was released by Google (Atwah et al. 2015). Since then, there has been a significant increase in the investigation into WebRTC for healthcare delivery. Much research has been developed to exploit WebRTC in healthcare applications to provide a better quality of life for the elderly and individuals with disabilities. Chiang et al. (2014) presented a video conferencing system based on WebRTC for seniors while Web shopping for older adults was also presented in (Hongo, Yamamoto & Yamazaki 2014). However, to the best of our knowledge, there has been no previous work which uses WebRTC for a telepresence wheelchair to assist wheelchair users for healthcare purposes.

The WebRTC has been developed and released by Google as an open source. It was standardized by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) (Kilinc & Andersson 2014; Singh, Lozano & Ott 2013). IETF is

standardizing protocols used in WebRTC, while JavaScript application programming interfaces (APIs) implemented in Web browsers is being standardized by W3C (Kilinc & Andersson 2014). WebRTC aims to enable real-time communications between Web browsers, without requiring any plugins. In our development, we have employed the following key components of WebRTC:

- Media Stream is the component which enables the browser to access media devices, such as microphones, webcams, and speakers. As shown in Figure 5.1, the Media Stream represents a synchronized stream of media data. Each Media Stream has inputs and outputs. The era of HTML5 has enabled direct hardware access to various devices and provides JavaScript APIs which can develop an interface with a system's hardware. `getUserMedia` is one such API which is offered as part of HTML5 enabling a browser to access a user's camera and microphone. The `getUserMedia()` function is requested to access the user to create and use a local Media Stream.

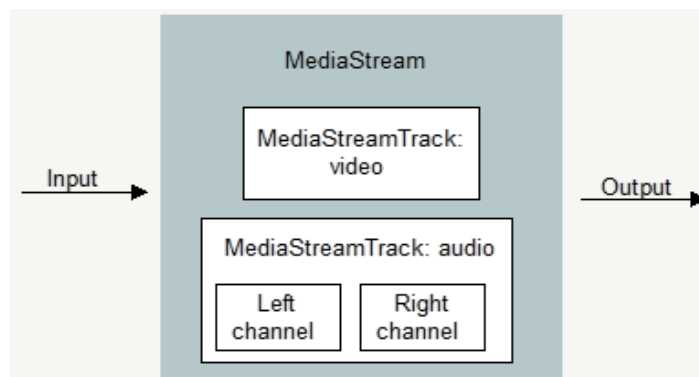


Figure 5.1: The MediaStream API component.

- `RTCPeerConnection` is the API which is a part of the WebRTC specification. An `RTCPeerConnection` is a peer connection which handles the peer-to-peer connection between the browsers by sharing the session object. It is used to represent a web real-time communication connection between the users and to handle efficient streaming of data between the two users. The `RTCPeerConnection` event call-back can be used to take care of dealing with the audio/video stream. It can be developed with Hypertext Markup Language, version 5 (HTML5) (Atwah et al. 2015).

- RTCDataChannel is the second primary API offered as part of WebRTC and is a channel providing the transport capability to the browser. The RTCDataChannel interface represents a bidirectional data exchange channel between two peers of a connection which includes an incoming and an outgoing stream. In other words, it is used to transport data directly from one peer to another.

Figure 5.2 below provides a WebRTC architecture diagram showing the role of RTCPeerConnection.

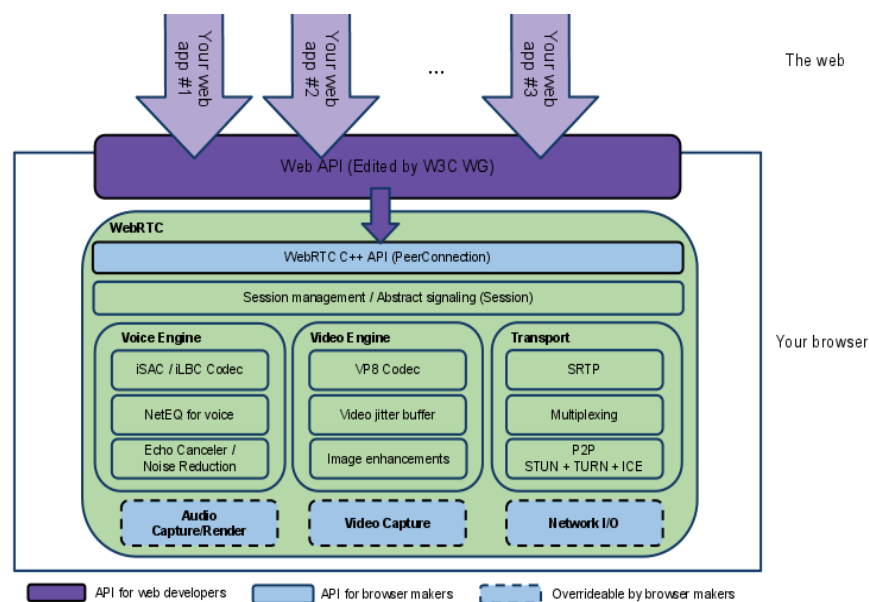


Figure 5.2: WebRTC architecture (WebRTC 2017)

The WebRTC architecture in Figure 5.2 indicates that the structure provides API for web developers, API for browser makers and overrideable by browser makers. Voice Engine is a framework for the audio processing to process a wideband and super-wideband audio codec for voice over internet protocol (VoIP) and streaming audio which is defined by IETF. A dynamic jitter buffer and error concealment algorithm are used for concealing the adverse effects of network jitter and packet loss to maintain voice quality. Video Engine is a framework video processing from camera to the network and from network to the screen. The codecs and video jitter buffer and image enhancement help to enhance the overall video. The protocols used by WebRTC do a very considerable amount of work to make real-time communication possible, even over unreliable networks.

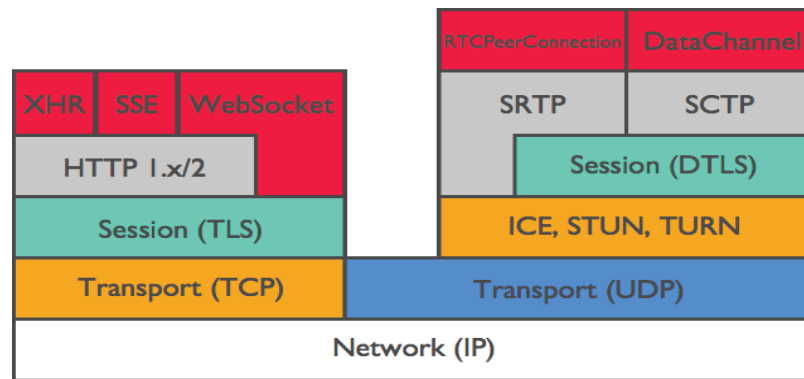


Figure 5.3: WebRTC Protocol Layer (Loreto & Romano 2012).

In WebRTC protocol as shown in Figure 5.3, WebRTC encrypts information using Datagram Transport Layer Security (DTLS). All data sent over WebRTC DataChannel is secured using DTLS. DTLS is a standardized protocol integrated into Internet browsers that support WebRTC and is one protocol consistently used in web browsers, email, and Voice over IP platforms to encrypt information. WebRTC communication is direct peer-to-peer (P2P) communication between two browsers, aided with signaling servers during the setup phase. Two browsers can connect with a P2P connection using Session Traversal Utilities (STUN), with a port mapping for Network Address Translation (NAT) traversal and Use Traversal Using Relays around NAT (TURN) as an intermediary as illustrated in Figure 5.4.

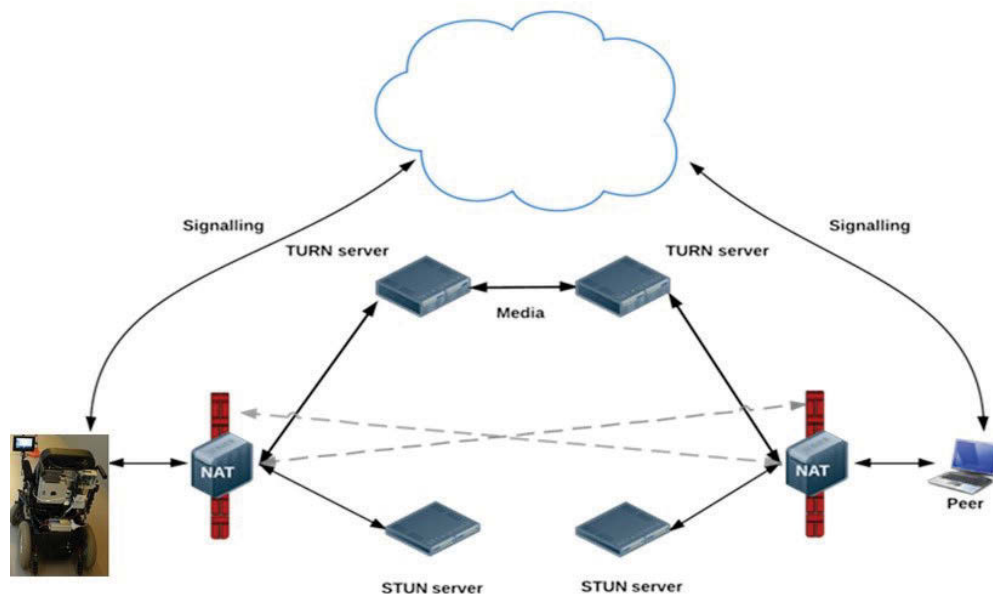


Figure 5.4: WebRTC using STUN, TURN, and Signalling (WebRTC 2017).

The signalling stage is a process to set up a bidirectional transmission of conversation. Signalling is the exchange of data information between caller and callee connections in the network to set up, notify, control, and terminate each connection when a connection is in progress. Also, the relevant information that browsers must exchange is the multimedia session description, which includes, the transport information as well as the media type and all associated media configuration parameters necessary to establish the media path.

To illustrate the procedure to make a connection between two users, consider the connection procedure of party (A) calls the other (B). The complete process is summarized in Figure 5.5 as follows: Each peer first establishes an IP address. Then, signaling data “channels” are dynamically created to detect peers and support peer-to-peer negotiations and session establishment. Once peers are connected to the same channel, the peers can communicate and negotiate session information.

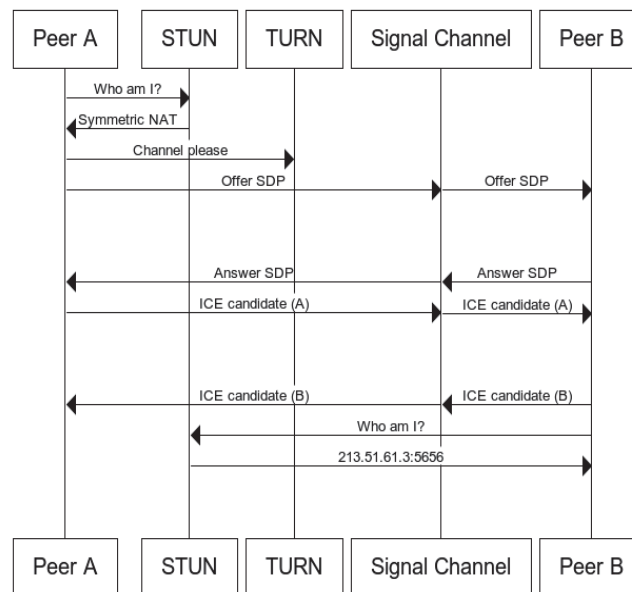


Figure 5.5: The signal of A and B connections (Mozilla 2017).

Peer A sends an “offer” using a signaling protocol and Session Description Protocol (SDP). Peer A waits to receive an “answer” from any receivers with the same “channel.” Once the response is received, the network session between the peers is then activated and established. Next, local data streams and the data channels are created, and multimedia data has finally transmitted both ways using bi-directional communication technology.

5.2.2 Prototype System Implementation and Deployment

In the present chapter, in order to perform a two-way audio and video communication, we develop a telepresence wheelchair using a more flexible web real-time communication framework. The architecture of the system is illustrated in Figure 5.6. We have developed a telepresence wheelchair system based on an electric-powered wheelchair platform modified with many add-on external modules. In order to achieve the desired features, the powered wheelchair is equipped with a Mac mini computer, control components, telecommunication interfaces, touchscreen display, and webcam.

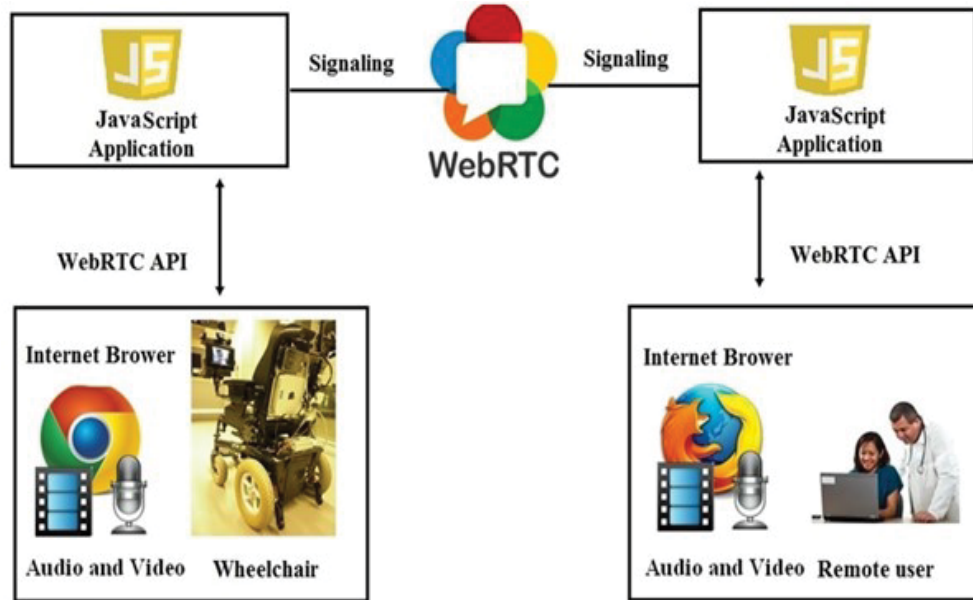


Figure 5.6: The WebRTC based telepresence wheelchair design.

For two-way video communications, a web page with the functions of establishing connections, acquiring, displaying a live video stream and controlling the wheelchair over the Internet from anywhere in real-time has been developed. To be able to achieve the remote interaction characteristics, a control interface with five buttons (move forward, go backward, turn right, turn left, and stop) has been designed to send the control signals from the remote user to the wheelchair over the WebRTC channels. These control signals are written and read by the application programming interfaces developed in JavaScript language and HTML5.

For telepresence systems which are capable of interactive video communication and remotely controllable from a distance, one of the most important design objectives is to transfer voice, video, control signal and data in real-time. The performance of data communications is carefully considered and thoroughly explored to understand the behavior of data transmission. There are many key factors affecting the performance of data transmission including bandwidth, throughput, round-trip-time, and frame rate. In order to evaluate the system streaming performance, these important factors are analyzed.

The bandwidth (W) is closely related to the maximum communication channel capacity in bits per second. In order to determine the maximum rate of transmission of data which can be sent over a link, the bandwidth of a communications link is measured. If the bandwidth is sufficient, the packet delay is very low, and interactive media applications work quite well in terms of user satisfaction. The channel capacity is given by the Shannon-Hartley theorem as follows:

$$C = W \log_2 \left(1 + \frac{S}{N} \right) \quad (5.1)$$

where S/N is signal-to-noise ratio. For a system transmitting at maximum capacity, the bandwidth efficiency of the system can be written as

$$\frac{C}{W} = \log_2 \left(1 + E_b \times \frac{C}{N_o \times W} \right) \quad (5.2)$$

where E_b is the average received energy per bit, N_o is the noise power density (Watts/Hz).

Obviously, the larger the ratio $\frac{C}{W}$, the greater the bandwidth efficiency is.

On the other hand, the throughput is the number of bits transmitted per unit time. A packet contains L bits, and the amount of time devoted to that packet is the actual transmission time (L/B) plus the propagation delay (d/V).

$$\text{Throughput} = \frac{L}{d/V + L/B} \quad (5.3)$$

The TCP-Friendly Rate Control (TFRC) bandwidth estimation model is used for bandwidth estimation algorithm, which relies on the p -value (the packet loss rate) (Kilinc & Andersson 2014).

$$T_{TFRC} = \frac{s}{RTT \sqrt{\frac{2p}{3}} + t_{RTO} \left(\sqrt[3]{\frac{3p}{8}} \right) p(1 + 32p^2)} \quad (5.4)$$

where s represents the average size of received packet size per second, RTT is round-trip-time and t_{RTO} is the value of retransmission timer. T_{TFRC} does not represent the actual WebRTC real-time video output rate but it represents an upper bound for it.

To deliver the video with improved quality, we adopted a Google Congestion Control (GCC) algorithm (Atwah et al. 2015). The sender-side uses the packet loss rate to estimate the sending rate (As) given by

$$As(i) = \begin{cases} \max\{S(i), As(i-1)(1 - 0.5p)\} & p > 0.1 \\ As(i-1) & 0.02 < p < 0.1 \\ 1.05(As(i-1) + 1\text{kbps}) & p < 0.02 \end{cases} \quad (5.5)$$

where $As(i)$ is the sender available bandwidth estimate at time i , $S(i)$ is the throughput, and p is the packet loss rate. The receiver-side estimates the receiving rate given by

$$Ar(i) = \begin{cases} \eta Ar(i-1) & \text{Increase} \\ Ar(i-1) & \text{Hold} \\ \alpha R(i) & \text{Decrease} \end{cases} \quad (5.6)$$

where $Ar(i-1)$ denotes the receiving rate estimate at time $(i-1)$, η is the receiving rate increase constant. $R(i)$ is the current incoming rate, and α represents the incoming rate decrease factor.

5.3 Integration of Emerging Technologies for an Advanced Telepresence wheelchair

5.3.1 Telepresence Wheelchair with 360-degree Vision

In chapter 4, to enhance the full field of view, we have used an array of cameras. Although the wide field of view has been improved, the use of an array of cameras results in a significant amount of data to be transmitted. In addition, the combination of images from multiple cameras to obtain the full view of the environment is challenging. To overcome these limitations, this section presents an innovative approach to a 360-degree vision telepresence wheelchair for healthcare applications. The study aims at improving a wide field of view surrounding the wheelchair to provide a safer wheelchair navigation and efficient assistant for the people with disabilities. A camera is mounted in front of the wheelchair to capture the dual-fisheye images, and these two parts of the field of view are stitched into a single seamless panorama, and then streamed over the Internet. We have adopted *a feature-based image stitching method* for panoramic reconstruction. Finally, to provide efficient data streaming, Web Real-time Communication is implemented.

The present work has developed an advanced system with a 360-degree view. For this reason, we have proposed a telepresence wheelchair in which a *dual-fisheye camera* is exploited to improve the wide field of view while web real-time communication technology (WebRTC) (WebRTC 2017) is employed to improve the video quality transmission. In particular, the Ricoh Theta S dual-fisheye camera is mounted in front of the wheelchair to provide coverage of the whole surrounding scene at the wheelchair location. As a result, a remote user is able to see a full view of 360-degree surroundings of the wheelchair for controlling the wheelchair more safely and efficiently. In order to improve the performance of the system, we developed and implemented a telepresence system using WebRTC. As previously mentioned, WebRTC is a free open source which defines a standardized set of communication protocols developing real-time applications including video conferencing, data transfer without the requirement of plugins or software installation. To stream the video and to define the communication paths, the signalling process is used to initiate the peer connection process between browsers.



Figure 5.7: A telepresence wheelchair equipped with a dual-fisheye camera.

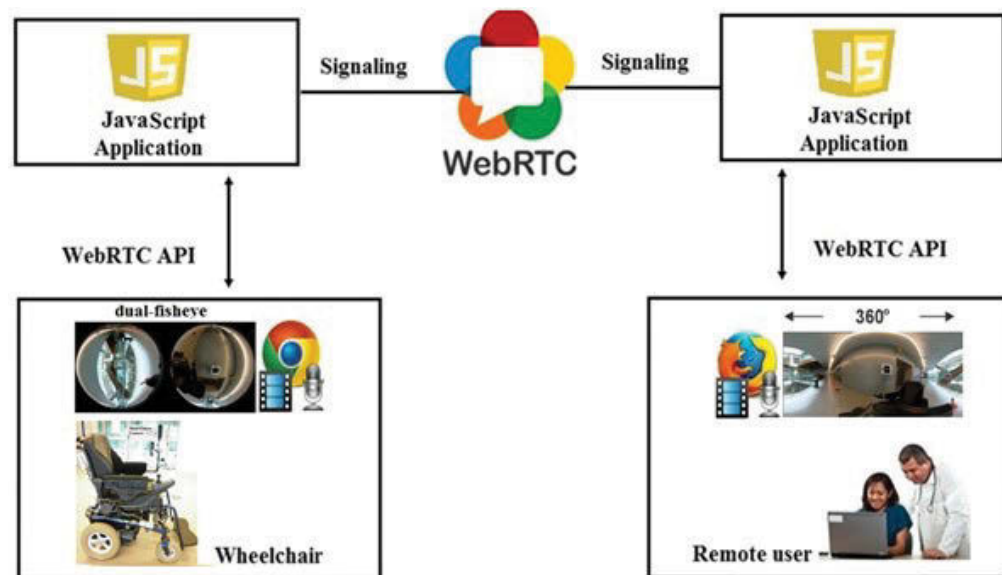


Figure 5.8: A 360-degree view telepresence using WebRTC architecture.

By developing an application programming interface and WebRTC enabled browser, real-time video communications can be set up. A 360-degree view telepresence using WebRTC architecture is illustrated in Figures 5.7 and 5.8. A remote user can observe the surrounding environment of the wheelchair in real-time over wireless internet connections. From the obtained full field of view, a method for remote control approach to navigating the wheelchair from a remote location to interact with the remote environment via must be developed.

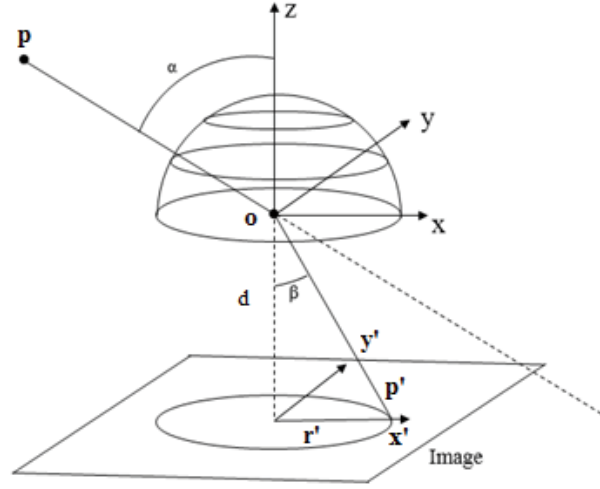


Figure 5.9: An equirectangular projection.

A dual-fisheye camera is used to obtain the 360-degree scene around the wheelchair. Now, we investigate the principle of the dual-fisheye camera in detail. A dual-fisheye camera is equipped with two fisheye lenses which can capture 180 degrees or larger field of view. There are various projection models proposed to represent a fisheye lens, such as equidistant projection, equip solid-angle projection, orthographic projection and stereographic projection (Schneider, Schwalbe & Maas 2009). In this study, an equidistant projection, also known as an equirectangular projection is used. Figure 5.8 shows an Equidistant projection model where (x, y, z) is a coordinate of an object in space and (x', y') is an image coordinate. Here, x and y -axes are parallel with x' and y' -axes, respectively. For the undistorted fisheye projection models, the equation projections are defined as follows (Schneider, Schwalbe & Maas 2009):

$$r' = d \times \alpha, \quad \tan \alpha = \frac{\sqrt{x^2 + y^2}}{z} \quad (5.7)$$

$$x' = \frac{d \times \arctan\left(\frac{\sqrt{x^2 + y^2}}{z}\right)}{\sqrt{\left(\frac{y}{x}\right)^2 + 1}} \quad (5.8)$$

$$y' = \frac{d \times \arctan \frac{\sqrt{x^2 + y^2}}{z}}{\sqrt{\left(\frac{x}{y}\right)^2 + 1}} \quad (5.9)$$

where d is the principal distance of the undistorted perspective image. α is the incidence angle. r' is the image radius. The perspective images of the object are de-projection. Based on (5.7)-(5.9), a conversion from fisheye images into perspective images can be done. By using real world image coordinates, fisheye projection equations to perspective image coordinates are mapped.

To assist a remote user in controlling and navigating the wheelchair more quickly, a panoramic view is developed by stitching two fisheye images. The panoramic image stitching involves two main processes, namely image registration, and the blending process, as shown in Figure 5.10.

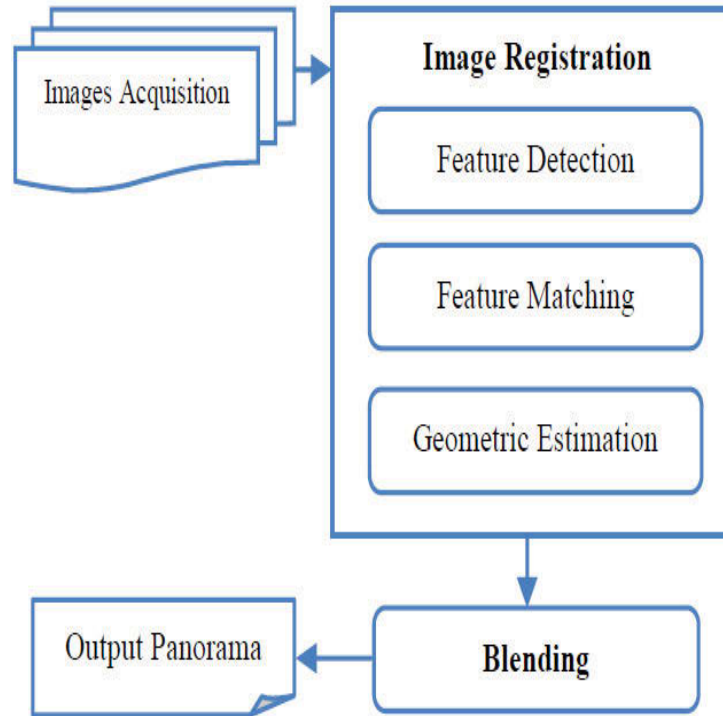


Figure 5.10: The block diagram of feature-based panoramic images stitching.

The registration step is to find the transformations to align two or more input images and the blending technique is used to smooth or blur the edges of a feature (Adel, Elmogy & Elbakry 2014).

In this work, a feature-based approach is used for image registration (Adel, Elmogy & Elbakry 2014). Particularly, the scale invariant feature transform (SIFT) is adopted, which offers scale and rotation invariant properties to extract the key image features. SIFT consists of four phases for computing the set of image features (Ali & Hussain 2012).

- Scale-space extrema detection: this stage extracts key points of interest based on the Gaussian function defined by

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (5.10)$$

where $*$ is the convolution operator. $G(x, y, \sigma)$ is a variable-scale Gaussian, and $I(x, y)$ is the input image. To detect stable keypoint locations in the scale-space, Difference of Gaussians $D(x, y, \sigma)$ which is the difference between two images in different scales is used. $D(x, y, \sigma)$ is defined as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (5.11)$$

- Key point localization: after removing extremas with low contrast, interest points are selected as key points. This is achieved by calculating from the 2×2 Hessian matrix given by

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (5.12)$$

- Orientation Assignment: each key-point is assigned one or more orientations based on local image gradient directions.
- Key point descriptor is performed on the image closest in scale to the key point's scale resulting in vectors SIFT keys which are used to match the features in image registration stage.

5.3.2 Remote Control Telepresence Wheelchair via WebRTC

In the preceding sections, we have presented a unique telepresence wheelchair system with 360-vision video communication. To establish a real-time 360-degree vision wheelchair interaction, both the remote client and the wheelchair must retrieve WebRTC-enabled web browser from a web application server. Through the Internet browser, the remote user and the wheelchair must establish a channel for video communication. A peer connection will be set from one to the other for exchanging media or data packets in real-time communications. The peer connection between the remote users typically employs the Secure Real-time Transport Protocol (SRTP) to carry real-time media channels and other protocols for real-time data interchange (IETF 2016).

However, WebRTC only supports video communication and provides the capability for modifying various aspects of the WebRTC API for developers to build independent applications. For instance, the users can make the modification related to speakers, microphones, video cameras and input data. WebRTC does not include a mechanism for exchanging the control signal. Therefore, in order to control the wheelchair from a remote location over WebRTC, a controller API must be developed to transfer the control signals from the remote user to the wheelchair.

In this research, the remotely controlling architecture via WebRTC for a telepresence wheelchair is depicted in Figure 5.11. As developed herein, a WebRTC media channel refers to a connection between two WebRTC clients for exchanging real-time media, while a WebRTC data channel relates to a connection for transferring data in one or more arbitrary formats. WebRTC data channel uses Stream control Transmission Protocols (SCTP) which allows it to deliver and retransmit data. Moreover, WebRTC data channel API supports a flexible set of data types such as strings and binary types. These types of data can be helpful when working with the transferring of a control signal from a remote user to the wheelchair. By using WebRTC data channel API, the control signal can be sent from a remote user to the wheelchair at the same time as the video communication. It is worth noting that a media channel and a data channel will be multiplexed and demultiplexed into a single peer connection between the WebRTC clients with the same User Datagram Protocol (UDP).

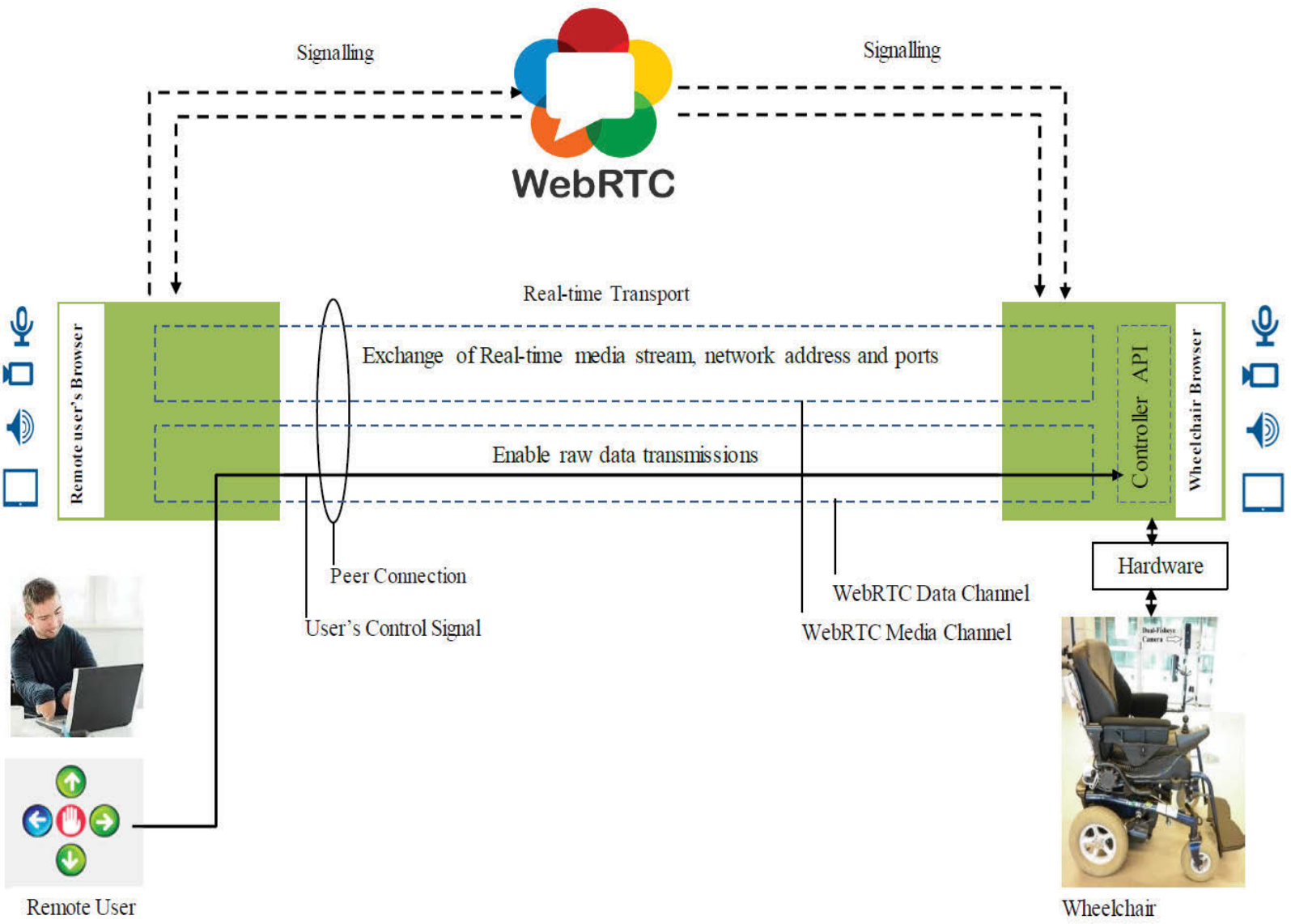


Figure 5.11: Remote control telepresence wheelchair based on WebRTC.

To transmit control signals we have developed an independent user-friendly interface with control buttons. The user interface permits a remote user to observe the surrounding environment of the wheelchair with audio and video communication and can navigate the wheelchair by using the control buttons at the same time and in real-time. In particular, to drive a wheelchair, a remote user as a WebRTC client uses an Internet browser to connect to the wheelchair via WebRTC. The control signal originating from the remote user will be sent via the WebRTC data channel. At the same time, responsive to receiving the control signal from the remote user via the WebRTC data channel, the controller API at the wheelchair browser will modify functionality associated with the wheelchair computer and transfer the signals to hardware.

In this study, the controller API module was designed and developed to interpret the control signals into the intended driving directions. In other words, the inputs of the controller API would be the data obtained from the remote user via the WebRTC data channel, and the outputs of the controller API would be the voltage values to control the wheel motors. The problem which needs to be solved here is that of how to develop a controller API which can communicate with the hardware. In this work, we use a USB-6008 for data acquisition and communication between WebRTC and hardware of the wheelchair platform. The idea behind this technique is straightforward: the required input signal which includes driving command of the wheelchair and the outputs will be used to control the motors. By doing so, the framework will exchange information between hardware and software during the navigation process.

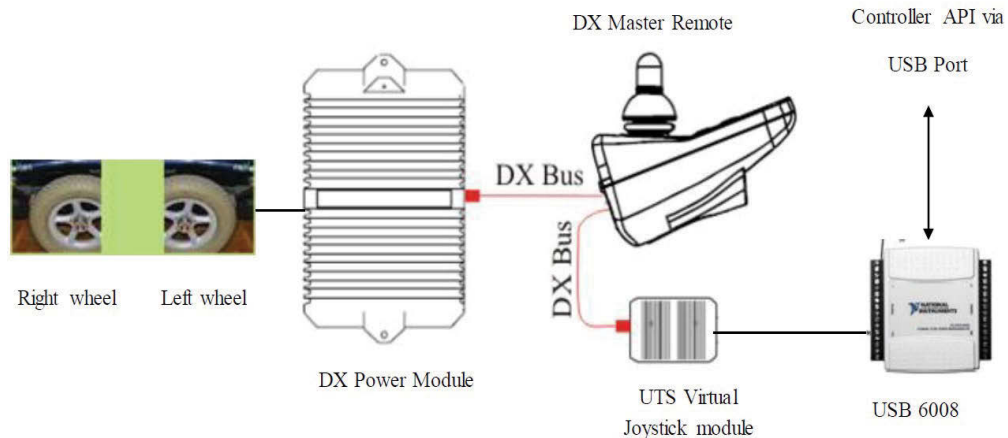


Figure 5.12: The connection of the control components.

Since the telepresence wheelchair has been designed based on the standard commercial electronic power wheelchair, therefore, to control the wheelchair, some modification and development have been done. The modification and connection of the hardware are described in Figure 5.12. The control signals generated by the WebRTC controller API would be transferred to the USB-6008 via a USB interface; after that, the USB 6008 produces the out- voltages and triggers the DX master remote via our design DX module – named Virtual joystick module. Finally, based on the received voltages values, the DX master would execute the DX power module to control the wheel motors accordingly.

5.4 QoE Estimation of Wireless Video Streaming Using Neural Networks

In a wireless network controlled telepresence system, the video transmission quality relies highly on the wireless network condition. Notably, the wireless signal fluctuations are affected by many factors and are difficult to forecast while the telepresence wheelchair is moving. The movement of the wheelchair into a weak wireless signal area can cause poor video streaming performance. Prediction of the quality of experience (QoE) is vital for a telepresence wheelchair in the short-term to react in time. However, the prediction of QoE while the wireless signal is not efficient to prevent loss of signal is a challenge. Thus, it is highly desirable to be able to predict the quality of service and, then, to properly control the wheelchair towards a wireless coverage area with strong signals.

The quality of experience evaluation of video can be either subjective or objective. However, subjective quality assessment methods for video are costly and time-consuming. Thus, objective evaluation is highly desirable to overcome these limitations by providing a mathematical calculation for the estimation of quality. Based on the availability of the original video signal, objective video quality assessment methods are categorized as full reference (intrusive) methods and free reference (non-intrusive) methods. Intrusive prediction models require access to the source, for example, Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM) whereas non-intrusive models do not and, hence, are more appropriate for real-time applications.

From the literature, there are several other artificial intelligent techniques (Machine Learning based) used to measure video quality, such as Artificial Neural Networks (Prasad et al. 2012), (Botia Valderrama et al. 2016), Random Neural Networks (Ghalut & Larijani 2014), and Fuzzy Systems (Pokhrel et al. 2013). However, very little work has been done on predicting video QoE considering the impacts of the fluctuations of wireless network conditions. Therefore, in this section, we develop a model based on artificial neural networks which are applied to predict the subjective quality metric of Mean Opinion Score (MOS) for QoE estimation of video streaming using WebRTC over wireless networks.

The aim of this section is to present an estimation model based on Neural Networks for objective, non-intrusive prediction of video quality over wireless streaming using WebRTC for video applications. To find an effective method to deal with uncertainty in the quality of service, we introduce an artificial neural network based quality estimation model in streaming media applications that can estimate the quality of the video. Accurate QoE estimation of the system is necessary for the remote user to control the wheelchair towards reliable wireless signal coverages to maximize the QoE to meet the needs of the real-time communication. The accuracy improvement of QoE prediction is a significant first step towards a more optimized feedback system that can efficiently control the system while moving the wheelchair into weak wireless signal coverage in the short-term.

- **Neural Network**

The neural network (the McCulloch-Pitts threshold neuron) was firstly introduced in 1943 (McCulloch & Pitts 1943). Neural networks are a system consisting of configurable parameters namely “weights” and “biases.” These parameters determine the neural network (NN) behavior and directly affect the accuracy of the neural network. The parameters of weights and biases are obtained during the training phase of model development. The iterative adjudication of the parameters in the NN is illustrated in Figure. 5.13. In the neural networks, inputs are multiplied by the weights and added to the biases to produce the target output. Based on a comparison of the output and the target, the weights and biases are iteratively adjusted until the network outputs match the targets.

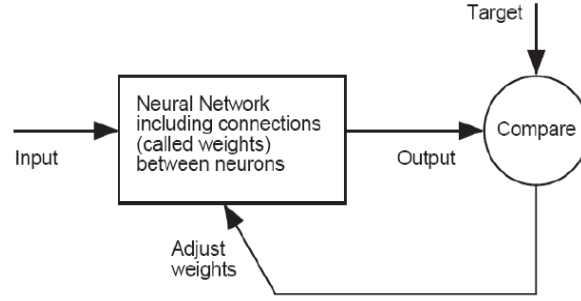


Figure 5.13: Neural Network.

- **One layer of neural networks**

A simple one-layer NN with R inputs and S neurons is shown in Figure 5.14 in which each component of the input vector p is connected to each neuron input through the weight matrix W defined as

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix} \quad (5.13)$$

where $w_{i,j}$ is the weight connected from the input j to the neuron i . The i -th neuron adds all its weighted inputs and bias to achieve its own scalar output $n(i)$. Then, the neuron layer outputs denoted by a column vector a can be computed by

$$a = f(Wp + b) \quad (5.14)$$

where p is the input vector, b is the bias vector and f is a transfer function.

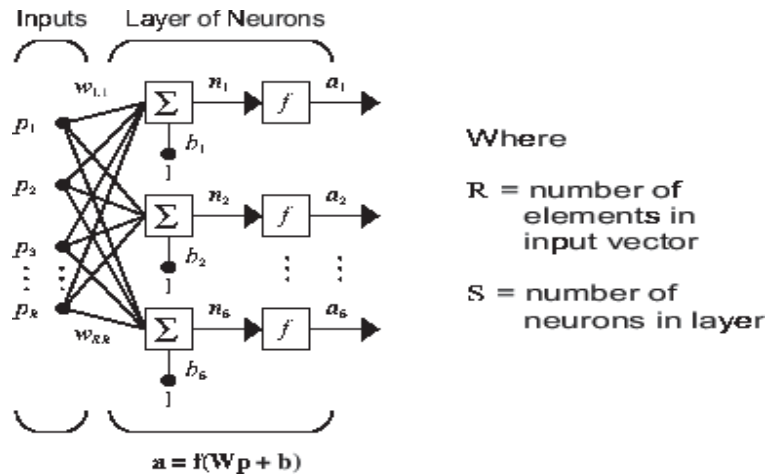


Figure 5.14: One layer of neurons (Matlab 2000).

The common transfer functions are logsig, tan-sigmoid and linear functions as shown in Figure 5.15.

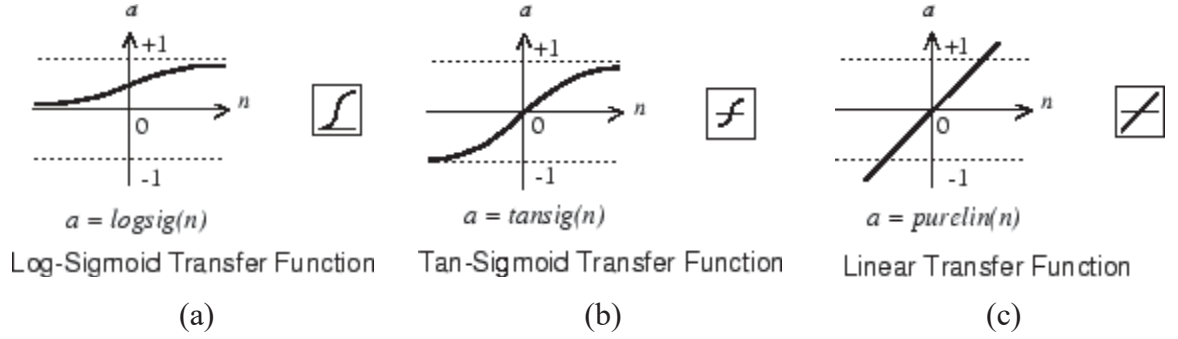


Figure 5.15: Typical transfer functions.

- **Multiple Layers of Neural Networks**

To allow the NNs to learn the nonlinear and linear relationships between the inputs and outputs, a feed-forward NN as Figure 5.16 with one or more hidden layers can be used. Each layer consists of a weight matrix W , a bias vector b , the output vector a . In the feed-forward NN, data from the inputs are transferred to the hidden layers and, then, to the output layer. The number of neurons at the inputs and outputs are determined by the specific applications. The number of hidden layers and the number of neurons in each hidden layer are mainly selected based on the developer's experience and the requirements of applications.

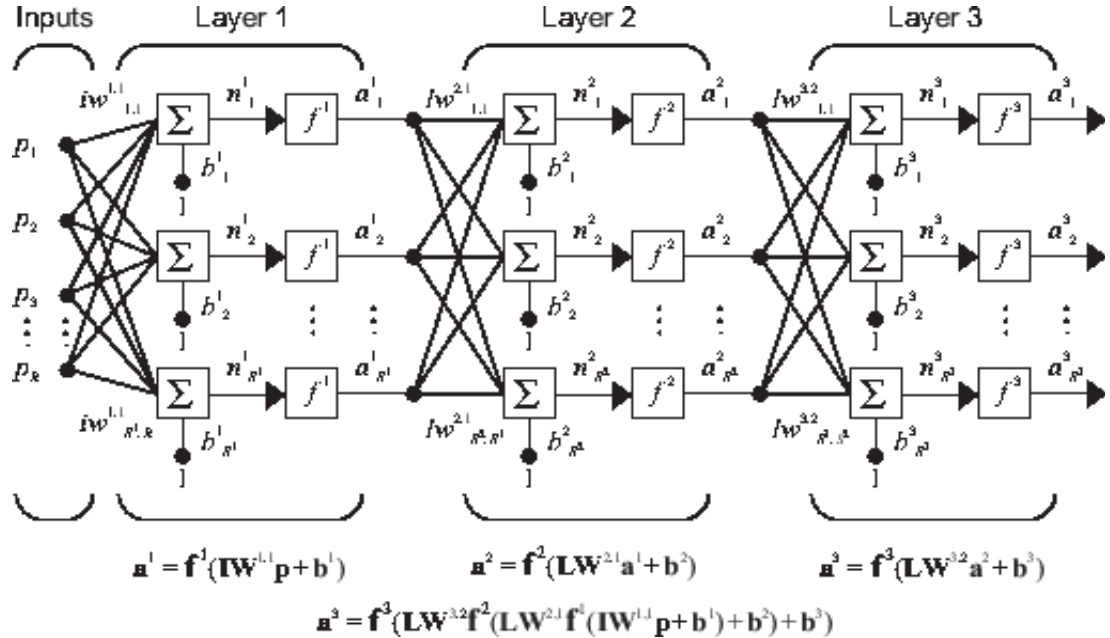


Figure 5.16: Multiple layers of Neural Networks (Matlab 2000).

If the last layer of a multilayer network has sigmoid neurons, the outputs of the network are restricted to a small range. If linear output neurons are used, the outputs can take on any value (H. Demuth 2000).

- **QoE Estimation Model Based on Neural Networks**

In this section, we introduce the QoE estimation based on the NNs as shown in Figure 5.17. By exploiting the NNs, the measuring of the quality of video can be done without accessing the original videos. After the NN is trained, the video quality evaluation can be performed in real time.

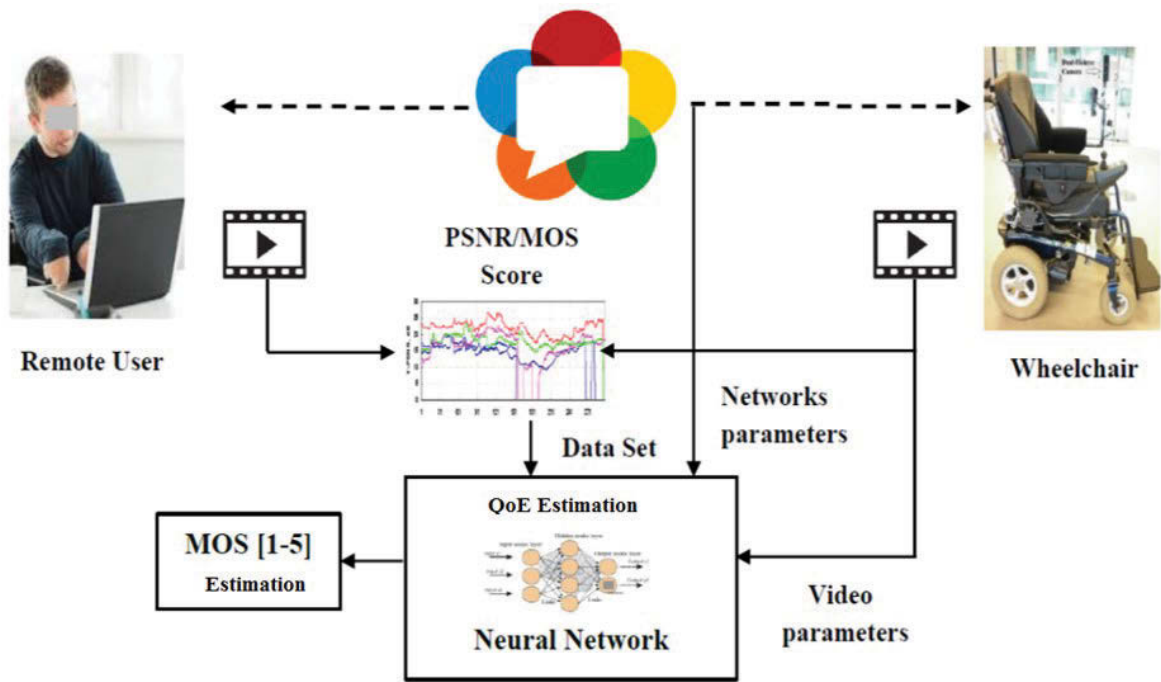


Figure 5.17: QoE Estimation model based on the Neural Network

This model uses a combination of objective parameters in the application and network layers, such as Sender Bit Rate (SBR), Jitter, Packet Loss Rate (PLR), and Round-Trip Time (RTT). The video quality was predicted in terms of the Mean Opinion Score (MOS). With the aim of estimation of video quality of experience, a neural network with a feed-forward three-layer topology is developed as shown in Figure 5.18.

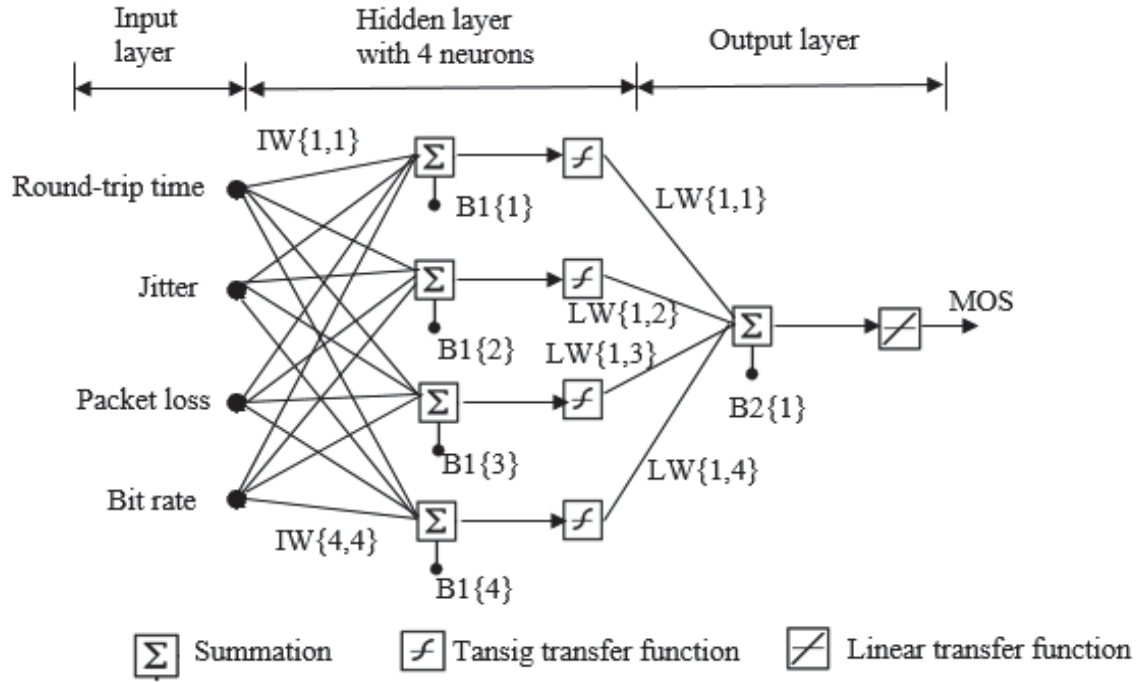


Figure 5.18: Block diagram of QoE estimation based on the Neural Network.

There are three layers in the network, one input layer, one hidden layer and one output layer. The input layer includes values of Sender Bit Rate (SBR), Jitter, Packet Loss Rate (PLR) and Round-Trip Time (RTT), and the output layer is the estimated QoE in terms of Mean Opinion Score (MOS). MOS has commonly been used as the subjective metric during human subject experiments. To evaluate the video quality, ITU defined a 5 level scale, called as ITU-5 point impairment scale, as shown in Table 5.1. To train the NN, the collection of MOS rankings in compliance with the ITU-T standard (Botia Valderrama et al. 2016; ITU-R 2012) during voting periods of human subject experiments was carried out.

Table 5.1: ITU 5-Point Quality Scale Measurement of MOS.

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very Annoying

As presented in (Zinner et al. 2010), (Botia Valderrama et al. 2016), there is a close relationship between the MOS, the PSNR, and SSIM. Table 5.2 shows that the PSNR and SSIM (as defined in Chapter 4) can be mapped to the corresponding MOS.

Table 5.2: Mapping the PSNR, SSIM to the MOS.

MOS	PSNR	SSIM	Quality
5	≥ 37	≥ 0.99	Excellent
4	32-37	0.95-0.99	Good
3	25-32	0.88-0.95	Fair
2	20-25	0.55-0.88	Poor
1	< 20	< 0.55	Bad

After training the NN, the weights and biases parameters in the NN are determined and, then, the closed form expressions for online prediction of user QoE can be obtained. The estimated QoE in terms of MOS is a function of online measurable network parameters (Jitter, SBR, PLR, and RTT) obtainable from wireless streams at monitoring points in the network. In order to develop the models, a series of well-designed objective and subjective test cases have been executed for the experiments. The ultimate goal of the objective and subjective testing is to come up with a set of user QoE training data from human subject experiments that can be fed to a neural network tool.

5.5 Experiments and results

5.5.1 Experiment 1: Telepresence Wheelchair based on WebRTC

A. Description

In this first experiment, we conducted three case studies, and the experiments consisted of 30 trials. We evaluated the performance of the proposed method by implementing WebRTC to stream videos from the wheelchair to a remote location over the Internet with three types of image resolution of 320×280 , 640×480 , and 1024×768 pixels, respectively. Ten trials were repeated for each resolution. For each experiment, the data transmission was observed during a period from 0 to 500-seconds.

The experiments were carried out at various locations where a remote user was a distance away from a wheelchair. The wheelchair was located in the Centre for Health Technologies in the University of Technology Sydney, and the remote user was located in other rooms. The remote user and wheelchair computer were configured with the Intel Core i7 CPU and 8GB RAM to have sufficient computing power. Window 7 (64bit), Internet browser (Firefox) was installed. The bandwidth management was taken into account to ensure the consistency of the system performance.

During the experiments, when the connection was established, the remote user could request video communication and control the wheelchair directly. At the remote site, a remote user can hear and observe the environments surrounding the wheelchair. The user interface and the video obtained from the remote user are shown as in Figure 5.19. Figure 5.19 illustrates the subjective quality assessments of the video communications by human perception. The results demonstrated that the remote user could obtain a high-quality video and could control the wheelchair with desired directions.

To further examine the system performance, we evaluated the other objective quality measures. The measurement data was recorded and collected carefully. A diagnostic packet and event recording related to the bandwidth estimation, packets sent, round-trip time and frame rates were measured and logged for analysis and evaluation.

B. Results

Figure 5.19 illustrates the user interface of the remote user in which the remote user could obtain a high-quality video and could control the wheelchair with desired directions from the control buttons while observing the surrounding wheelchair environments.

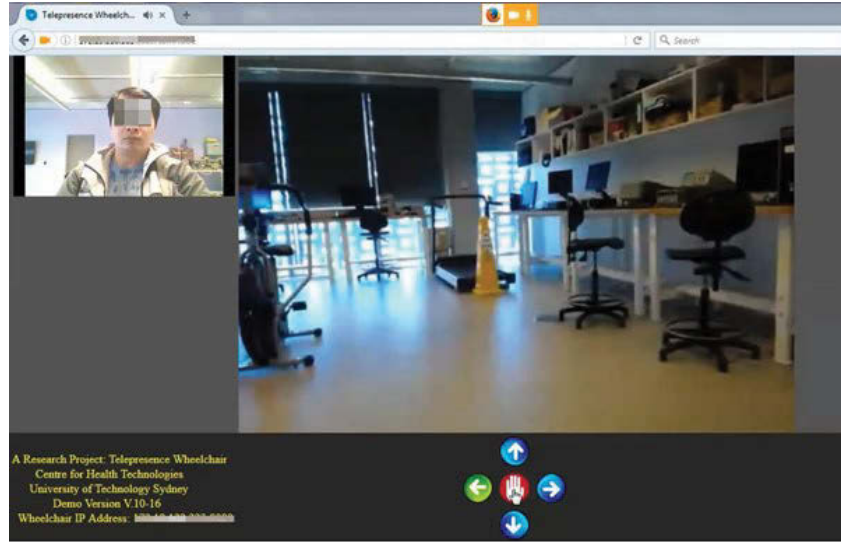


Figure 5.19: The user interface and the video obtained by the remote user.

Table 5.3: The average performance results of the 30-trials.

Platform	Resolution (pixels)	Avg. Encode (ms)	Avg. delay (ms)	Avg. throughput (kbps)
WebRTC	320×280	4.00±0.01	8.06±0.03	1641.635
	640×480	7.00±0.05	12.08±0.16	1617.374
	1024×768	11.00±0.02	18.01±0.23	1582.916

Table 5.3 provides the summary of results of the processing time of the encoded video, the delay and the throughput for each resolution in all trials. It can be seen from Table 5.3 that the mean (average) encoding time is increased when processing higher resolution videos. Delay during transmission may occur which in turn will result in increasing buffer size for temporary storage, causing longer latency. However, a higher resolution results in more data to be transmitted but does not affect the throughput of the streaming due to congestion control.

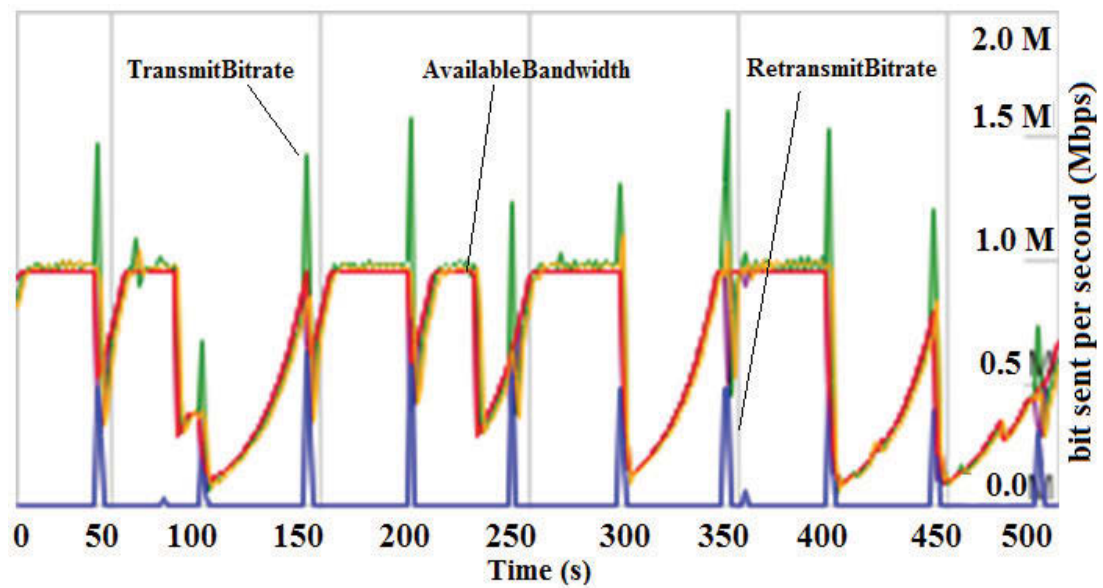


Figure 5.20: Bandwidth and bitrate of the system.

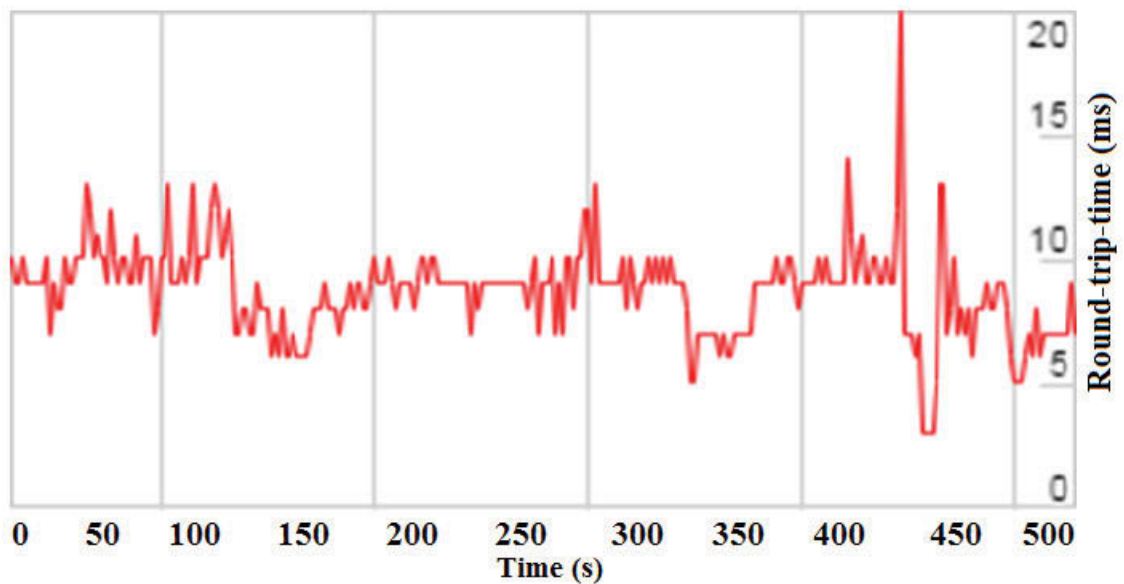


Figure 5.21: Round-trip time of telepresence based on WebRTC.

Figures 5.19-22 are the illustrations of the experimental results of streaming video at a resolution of 640×480 pixels. The results in Figure 5.20 show that the packets sending rate adapted with the available bandwidth and reached 1.5 Mbits per second. The results in Figure 5.21 indicate that the round-trip-time (RTT) fluctuated from 3 to 20 ms. In addition, it can be observed from Figure 5.22 that the streaming rate varied and reached a peak rate up to 30 fps.

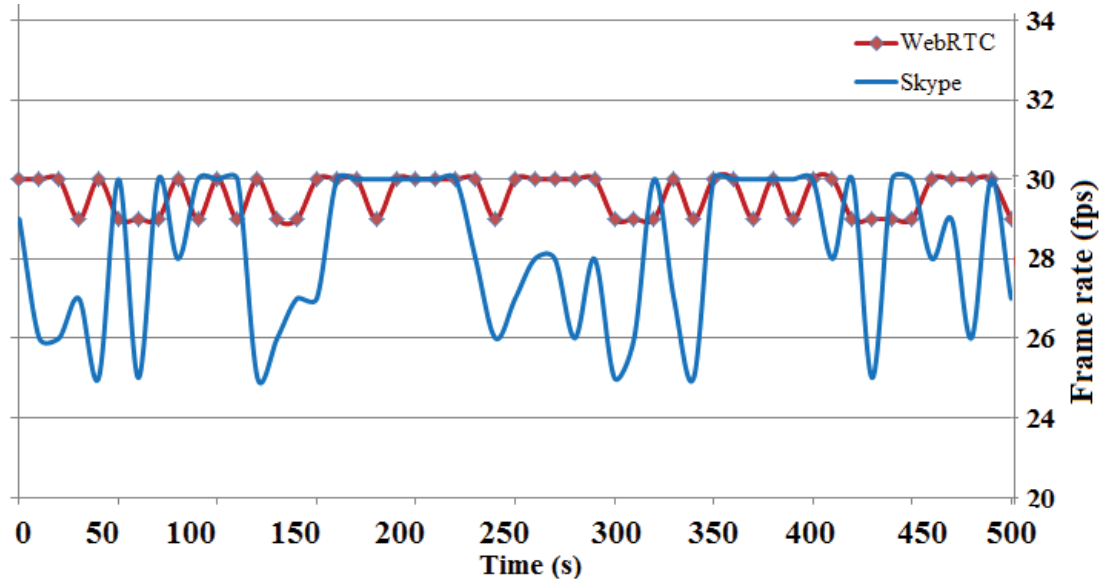


Figure 5.22: Frame rate sent (fps) of WeRTC versus Skype.

In comparison with (ETSI 2006), it is worth noting that the standard video frame rate is 25 fps (Europe) or 30 fps (North America). Therefore, the video streaming rate of the proposed system has reached the standard of real-time video communication. Regarding the round-trip time, it is worth noting that the RTT of less than 400 ms is considered as real-time communications (Ha, Nguyen & Nguyen 2016b). Obviously, the RTTs of WebRTC between two browsers in the experiments are under the RTT threshold value. The RTTs of the experiments are also significantly lower than the results of the recent telepresence robot study, which revealed that 40 ms delay was considered a good performance (Denojean-Maire et al. 2014a).

5.5.2 Experiment 2: An Advanced Telepresence Wheelchair with Wide Field View Using a Dual-Fisheye Camera

A. Description

The second experiment investigates the potential of enhancing user vision by using the dual-fisheye camera equipped with the telepresence wheelchair. The aims were to test the performance of images stitching, video streaming of the wheelchair from one area to another location over the Internet browser. For real-time implementation, we have implemented a complete image processing and stitching software by using web graphic library embedded in WebRTC. An application programming interface was developed for displaying video and transferring control signals.

The experiments were carried out with ten trials in an indoor scenario in the Centre for Health Technologies at the University of Technology Sydney. The wheelchair was set up at the balcony of our laboratory, and a remote user was located in a different location at a distance away. The remote user made a connection to the wheelchair over the Internet and controlled the wheelchair from the starting point through the obstacles and towards the end point in Figure 5.23. During the experiments, we recorded and extracted the processing information into log files for evaluation. All of the original images and received images were analyzed and simulated in MATLAB for performance evaluation.

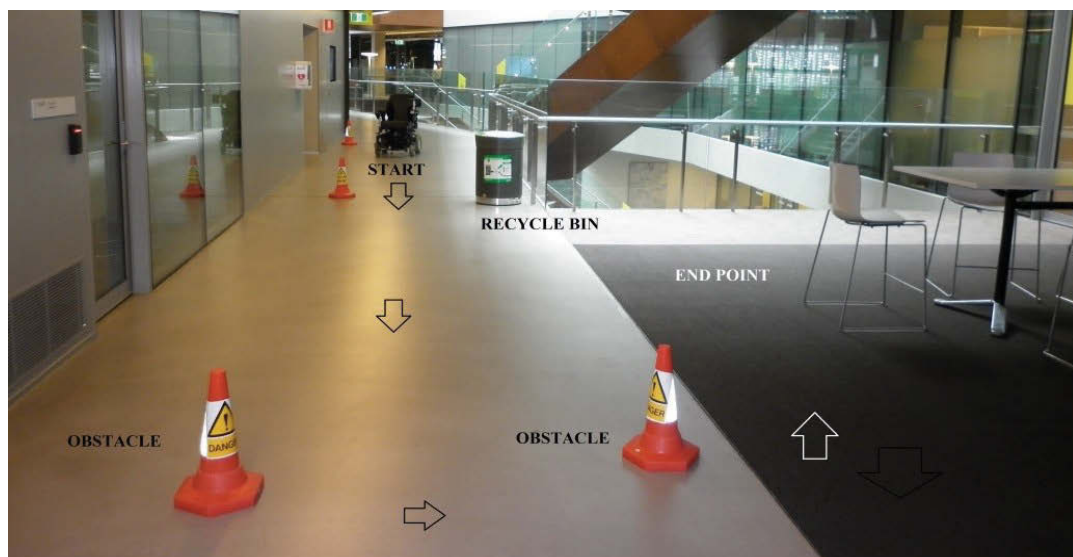


Figure 5.23: System performance test at the Centre for Health Technologies.

To evaluate and prove the effectiveness of the proposed approach, we conducted real-world panoramic video stitching and streaming from the wheelchair to a remote user via the Internet. At the remote site, a user was able to observe the wheelchair surrounding and control the wheelchair in real-time. The dual-fisheye camera used for development of the proposed telepresence wheelchair with a 360-degree vision is a Ricoh Theta S. This camera has a resolution of 1280×720 pixels with an input frame rate of 30 fps. The image processing was performed on an Intel Core i7 central processing unit (CPU), running on Windows 7 64-bit and 8 GB of RAM.

B. Results

Figures 5.24 and 5.25 provide the illustrative examples of the dual-fisheye images and the resultant panoramic view for visual evaluation. These figures also demonstrate a user-friendly application of programming interface with capabilities to display to the remote user in real-time the entire surrounding wheelchair environment that the camera can see. It also includes the control buttons as a remote control of the wheelchair for moving in different desired directions as the functioning of telepresence. During experiments, the wheelchair speed was set to 1 m/s to maintain reliable wireless connections. A remote user observed the surrounding of the wheelchair and controlled the wheelchair remotely from the start point to the end destination point accurately in real-time response.



Figure 5.24: Dual-fisheye video without stitching processing.

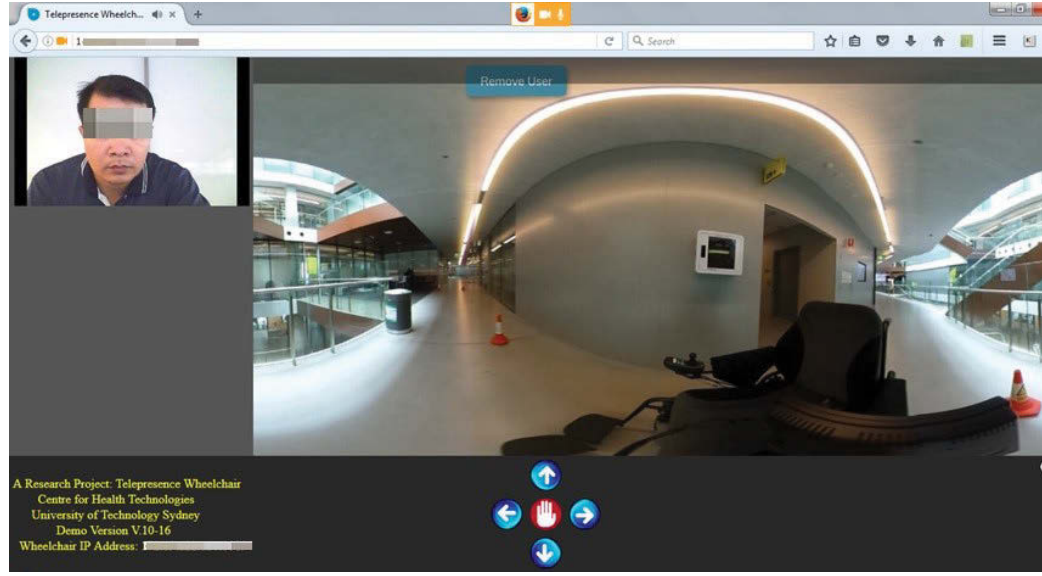


Figure 5.25: Spherical panoramic video obtained from a telepresence wheelchair and the control interface at the remote site.

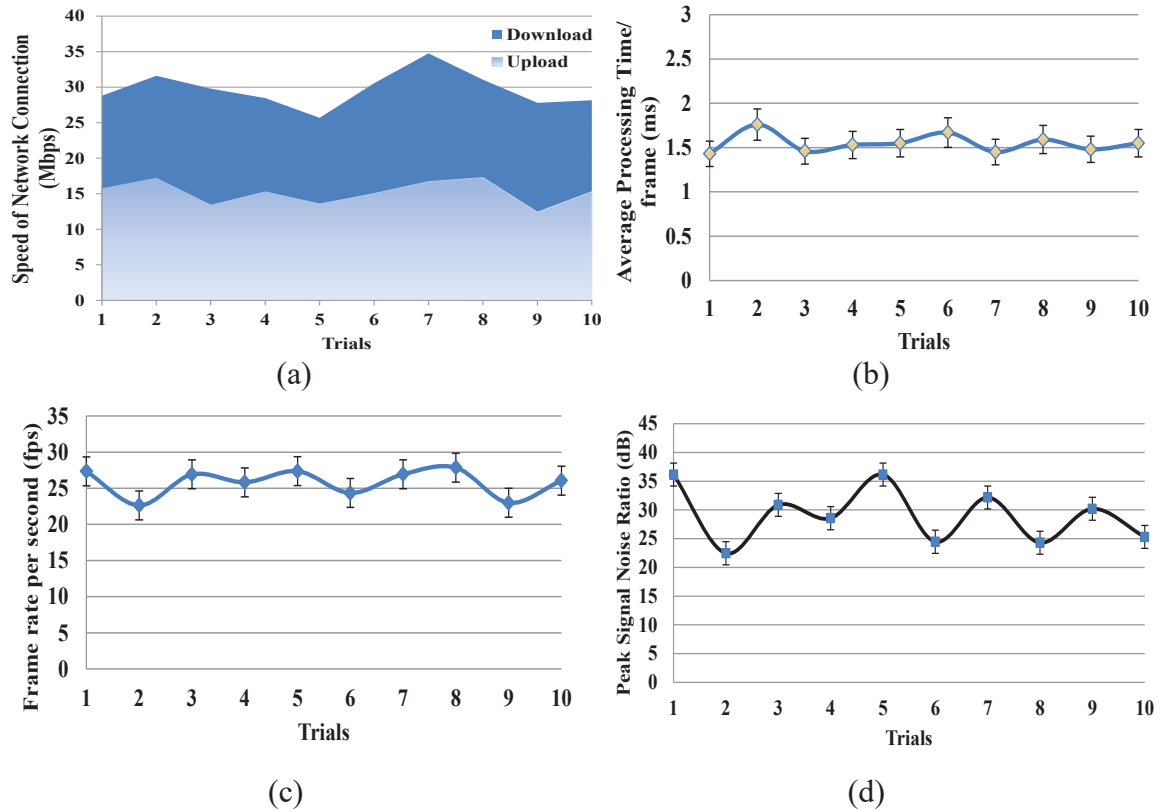


Figure 5.26: (a) The measurement of wireless network speed, (b) the average processing time per frame, (c) the frame rate and (d) the peak signal noise ratio of 10 experiments.

The video transmission performance results of 10 trials are illustrated in Figure 5.26. As can be seen from Figure 5.26 (a), the average download and upload speeds of the wireless networks were of 29.67 Mbps and 15.16 Mbps, respectively. The average of processing time per frame calculated over the experiments is 1.55 milliseconds in Figure 5.26 (b). It can be seen from Figure 5.26 (c) and (d) that the average frame rate of streaming the video with the resolution of 1280×720 pixels was 25.83 frames per second. The average peak signal to noise ratio (PSNR) was 29.06 dB. Note that the previous study of video streaming evaluation in (Yuwei, Deng & Nowostawski 2013) showed that the quality of the image is satisfactory for $25 \text{ dB} \leq \text{PSNR} \leq 32 \text{ dB}$ and good for $\text{PSNR} \geq 32 \text{ dB}$. In comparison with (Yuwei, Deng & Nowostawski 2013), it can be concluded that the proposed approach of a 360-degree vision for the telepresence wheelchair is feasible. Overall, the experimental results show that the streaming is smooth and seamless. The results demonstrate the effectiveness of our approach for the telepresence wheelchair with a wide field of view.

5.5.3 Experiment 3: QoE Estimation of Wireless Video Streaming Using Neural Networks.

A. Description

In this experiment, we developed a feed-forward neural network which consisted of multiple layers of neurons followed by an output layer of linear neurons. The model consists of an input layer, hidden layer and an output layer. The output layer is of the Purelin type which covers the entire MOS range. The modelling scheme chosen is a feed forward with a two-layer neural network. This gives the model the ability to approximate both linear and non-linear data functions. One hidden layer was used to perform the computation of weights and biases with considerations of the trade-off between performance and accuracy. It is known that the greater the number of hidden layers, the greater is the time taken by the model to compute the weights and bias. However, the weights and bias model the function very accurately.

The Tansig function (covers range -1 to 1) is used to facilitate accurate modeling of non-linear aspects of the modeled data. The linear function is used at the output layer such that the output can take the entire MOS values. In the training phase, it is important to define the number of Epochs which is the number of sweeps through all the records in the training

set. To have model accuracy, multiple sweeps are required. The number of Epochs affects both performance and accuracy of the model. With a large number of epochs, the model gets overtrained with given input set and does not correspond well to small fluctuations of input data. With a small number of epochs, model accuracy is reduced.

The type of training function used is the Trainlm (Levenberg-Marquardt), which is suitable in terms of accuracy for the size of the network in our problem. It cuts off the model training (number of epoch) before the model becomes overtrained with a given data set, and learns much faster than the normal train function training. The neural network toolbox available in MATLAB is used for developing our QoE model estimation. The proposed NN is presented in Figure 5.27. The parameters to be modeled are Network Jitter, Round-Trip-Time, Packet Loss, and Bit Rate. The resulting output is the Mean Opinion Score ranking with range (1 – 5).

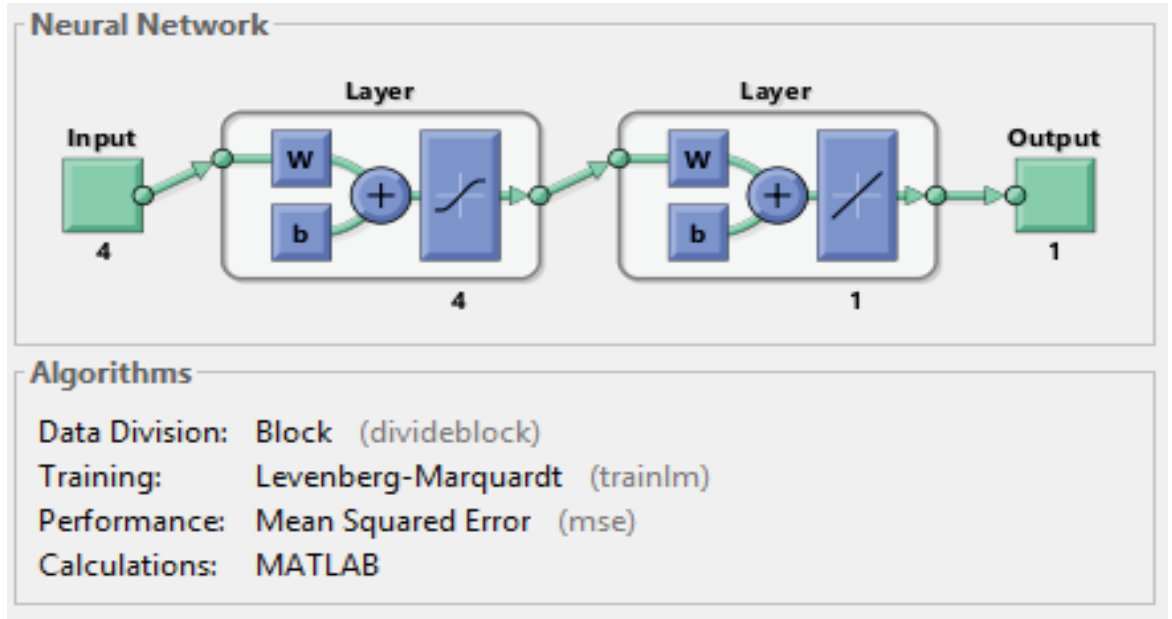


Figure 5.27: The proposed neural network for QoE estimation.

We collected the set of 300 cases in which MOS rankings are in compliance with the ITU-T standard. The NN was trained by using 100 cases while the other 100 cases were used for the NN validation and the remaining 100 cases were used to test the NN.

B. Results

Figure 5.28 indicates the performance progress for training, validation and test over epochs. It is worth noting that the number of epochs is the number of sweep through all the data records in the training set. The number of epochs is an important factor which determines the performance and accuracy of the NN model. It can be observed from Figure 5.28 that the best number of epochs is 12 where the validation performance reached a minimum.

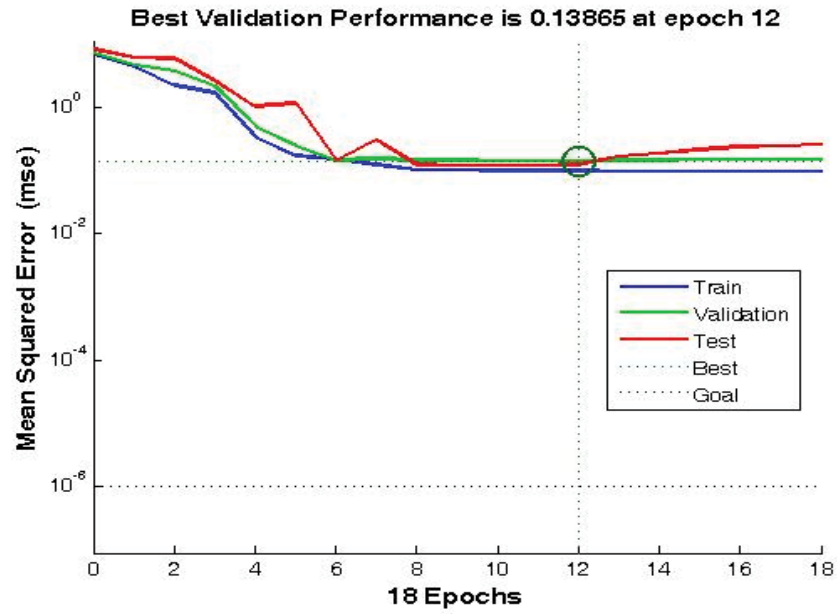


Figure 5.28: The performance progress over epochs.

To validate the NN, we provide a regression plot in Figure 5.29 which reveals the relationship between the MOS output of the NN and the MOS target. As can be seen in Figure 5.29, the training data shows a good fit while validation and test results indicate R values that greater than 0.9. We used the trained NN to estimate the QoE for 100 test cases of video streaming. The estimated MOS is presented in Figure 5.30 in comparison with the measured MOS. The correlation between the estimated MOS and measured MOS is about 94%. The simulation results demonstrated that the proposed scheme provides good predictive accuracy ($\sim 94\%$) between the measured and estimated values. The results achieved reasonable prediction values from the neural network models. This work can potentially help in predicting the quality of service (QoS) in order to react in time for video streaming of a telepresence wheelchair over wireless networks.

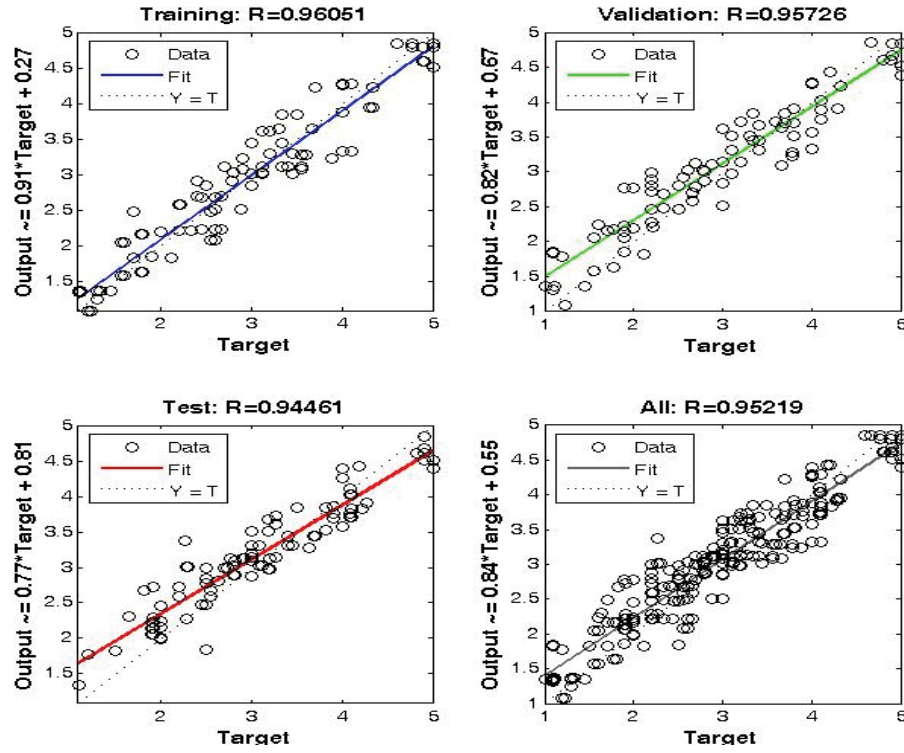


Figure 5.29: Regression plots for the training, validation, testing and all data.

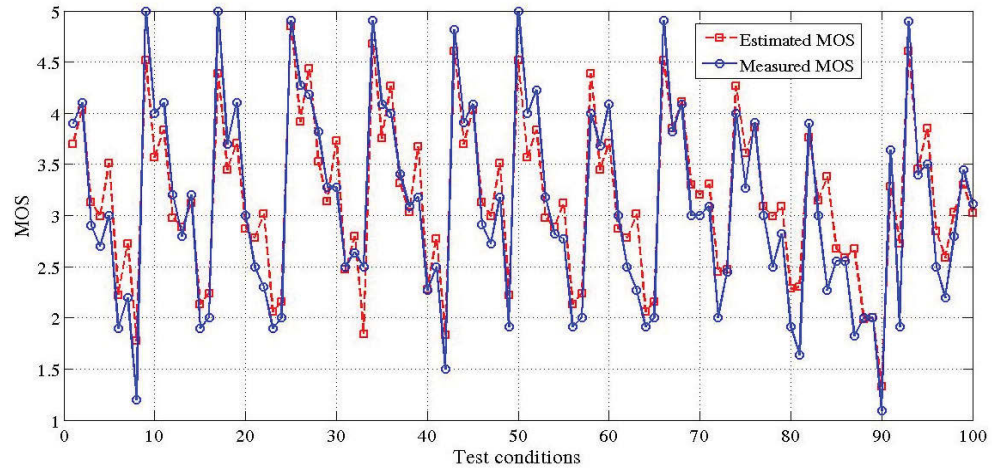


Figure 5.30: Estimated MOS based on ANN versus measured MOS.

The measured MOS in Figure 5.30 demonstrated that the video transmission quality in our telepresence wheelchair is satisfactory. The results show that good estimation accuracy was obtained from the proposed prediction model. This study should help in the development of an estimation of video quality of a telepresence wheelchair. Also, by observing the estimated MOS score, the remote user can navigate the wheelchair into the area which can guarantee the continuous connection and provide good performance.

5.5.4 Experiment 4: Evaluation of Remote Control Telepresence Wheelchair over WebRTC and 360-degree Vision.

The purpose of the experiment is to assess the improvement of performance and to further demonstrate the effectiveness of the remote controlling function of our design. The tests are carried out by remote controlling the wheelchair in three locations. The experiments were conducted both in indoor and outdoor environments. Ten experiments were conducted in each location. The average time spent for each test was two hours. The remote users were in a separate room where they could not see the paths and the targets, and they could only see them while driving the wheelchair through the telepresence function with the video stream. After being initialized and loaded with the necessary parameters, the telepresence wheelchair was ready to implement real-time experiments. The remote user used an Internet browser to connect and interact with the system to command the wheelchair. The remote user controlled the wheelchair by clicking on the control buttons while observing the surrounding environment. During the experiments, the remote users drove the telepresence wheelchair.

Firstly, the experiments were conducted in an indoor environment, at level 10 of Building 11, the University of Technology Sydney where some free spaces were available. The tests performed on the balcony where there is a wireless signal as shown in Figure 5.31. This environment was considered to be completely unknown to the remote user in each operation time. The remote user was far away from the wheelchair and was required to control the wheelchair to reach specified targets.

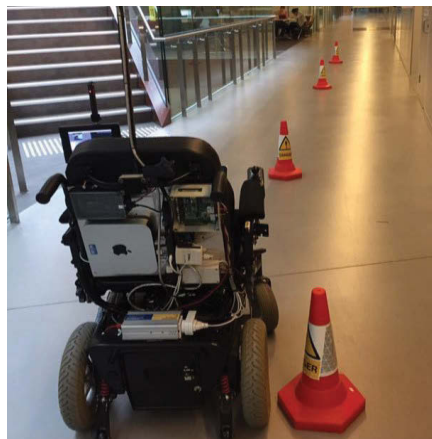


Figure 5.31: System performance test at the Centre for Health Technologies, UTS.

Figure 5.32 shows the environment and a trajectory produced by the wheelchair in the experiments. Starting from point A, the remote user controlled the wheelchair to move towards points B, C, D, and E respectively via the control of the remote user via a web browser. The wheelchair created a trajectory marked by the red dotted line. This trajectory is the result of the user commands during the navigation process. The results show that the remote user was easily able to move forward and control the wheelchair straight ahead.

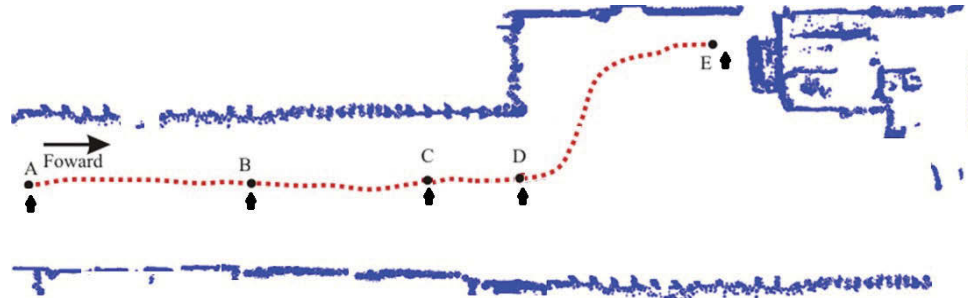


Figure 5.32: Trajectory of the wheelchair when the remote user controlled the wheelchair with straight ahead directions.

Secondly, the experiments were conducted in our laboratory's room. The test area is about 30 square meters. Objects have been placed in the test room to recreate a real home environment. The driving path is more challenging than that in the previous experiment. A path in a figure eight curve was drawn on the floor using a bright white dashed line as shown in Figure 5.33. It was marked to help the remote user to perform the task and to avoid obstacles in the room. The visual results of this experiment are illustrated in Figures 5.34 and 5.35. In this experiment, the remote user was asked to drive the wheelchair following the route which was drawn on the floor. During experiments, all trails in the testing were also video recorded, and the paths were traced and then used for evaluations.



Figure 5.33: Test path setup with the figure eight curve.

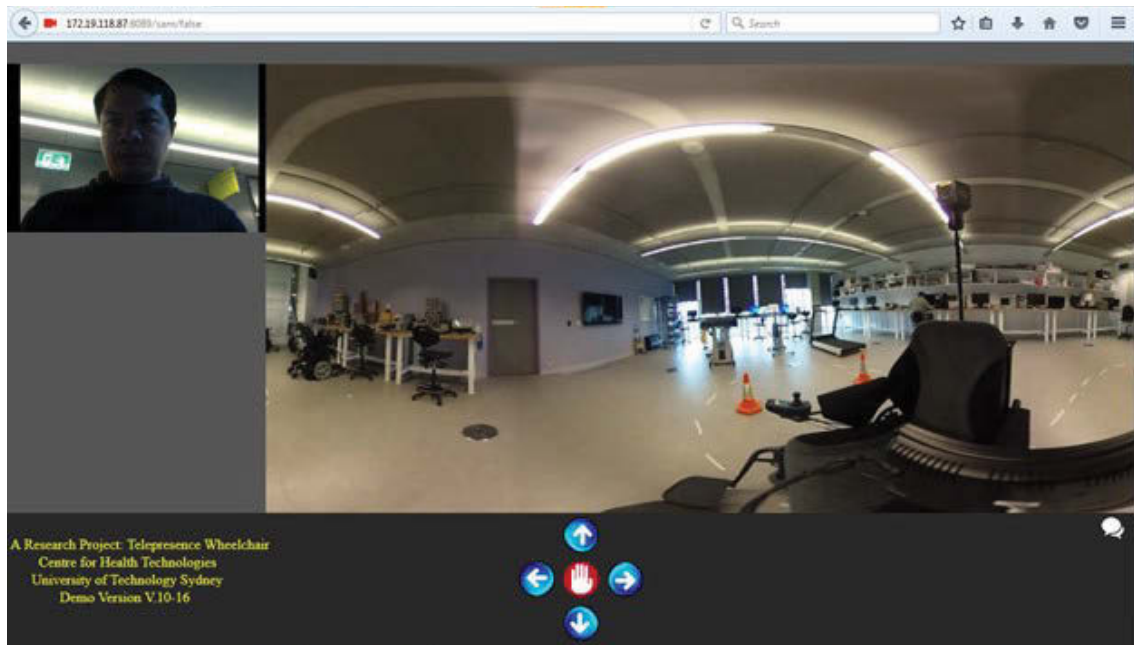


Figure 5.34: The full field of view obtained at the remote user.

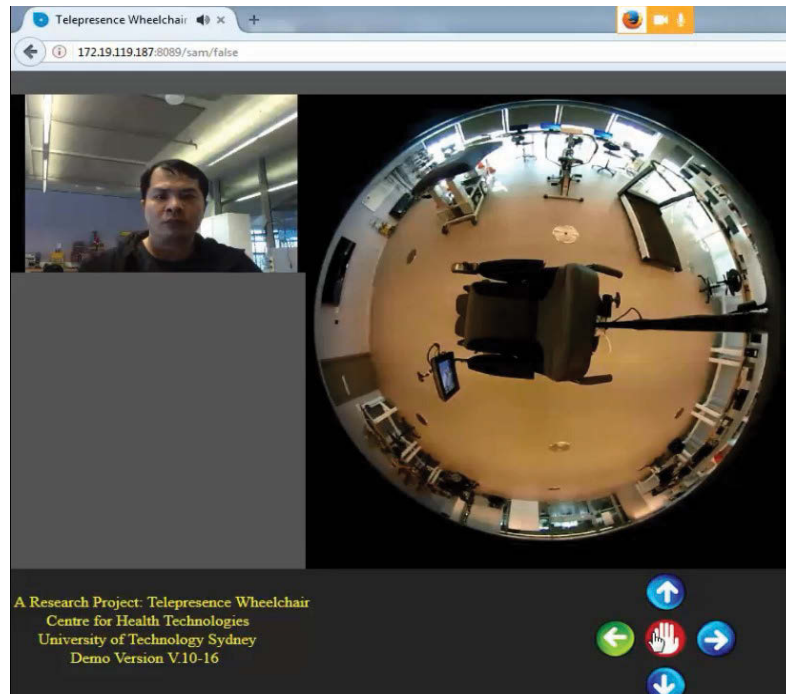


Figure 5.35: The full top-down view obtained at the remote user.

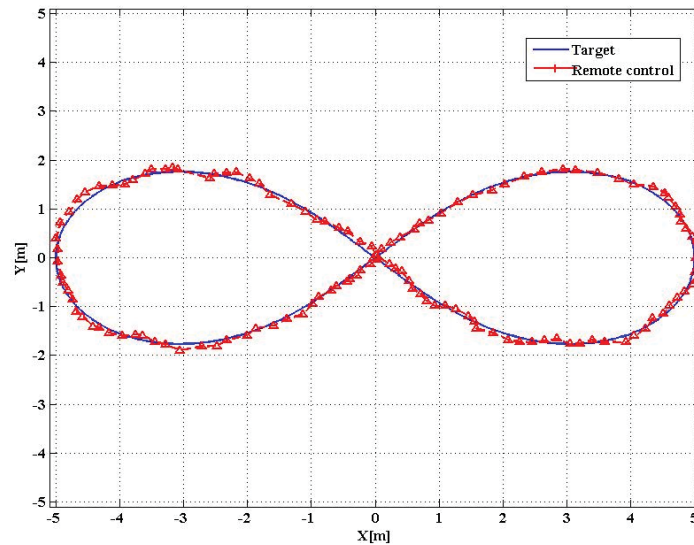


Figure 5.36: The trajectory of the remote control telepresence wheelchair and the desired path.

Experimental results of the remote controlling task in an indoor environment are shown in Figure 5.36. It can be observed from Figure 5.36 that the performance of telepresence wheelchair is very similar to the planning path.

The final experiment presents the remote control performance of the telepresence wheelchair in an outdoor environment as shown in Figure 5.37. The test was conducted for validating the effectiveness of the remote control in an outdoor environment. The tests were carried out at the campus of the University of Technology Sydney.

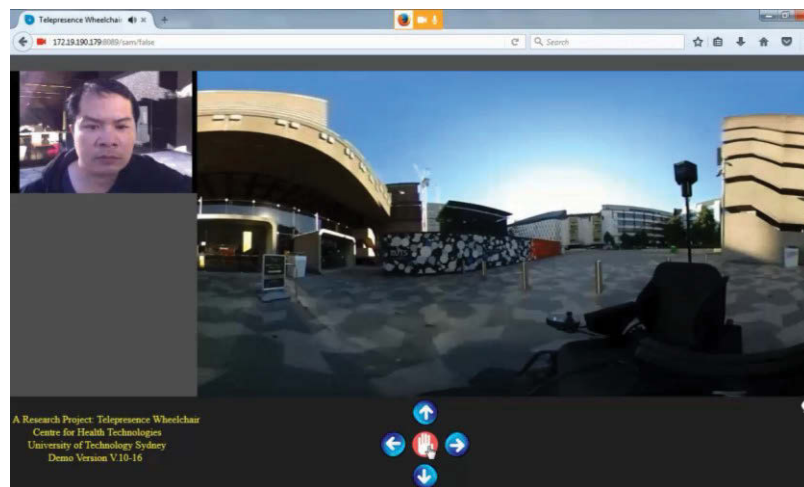


Figure 5.37: The telepresence wheelchair experiments in an outdoor environment.

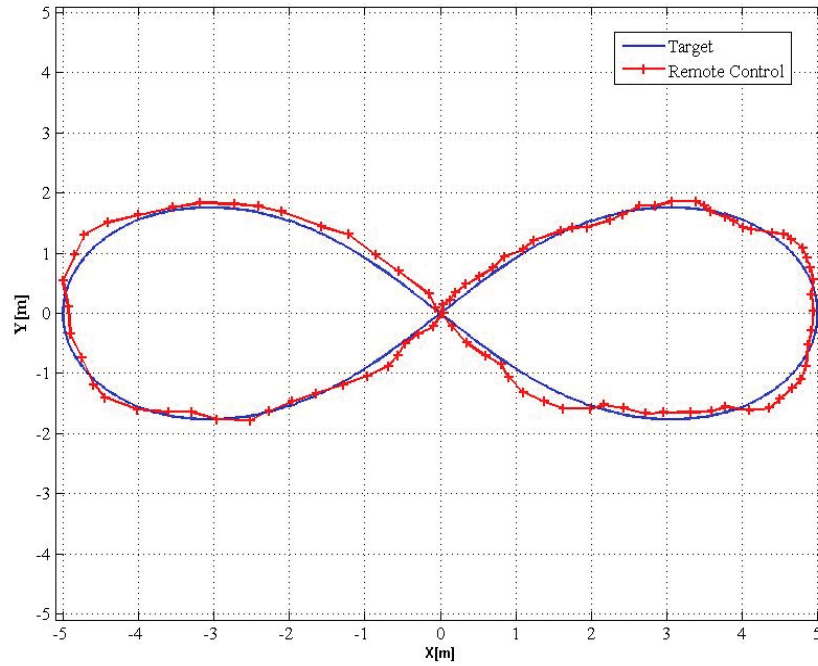


Figure 5.38: The trajectory of the remote control telepresence wheelchair in an outdoor environment at the campus of the University of Technology Sydney and the desired path.

It can be seen from Figures 5.36 and 5.38 that the trajectories made by two experiments are very similar to the planning paths with small errors. The results indicate that the wheelchair responds accurately to the control signal and the performance of the remote control is satisfactory under the design conditions with wireless coverage. These experimental results were found to be acceptable with the desired results.

5.6 Discussion

This chapter has presented the development of an advanced telepresence wheelchair which is able to provide a 360-degree field of vision using WebRTC. The complete hardware and software implementations of the telepresence wheelchair system, along with the emerging technology in the field of informatics and computer vision have been presented. The experiments were carried out to evaluate the system performance. The experimental results are reliable and positive compared to the standard recommendation and other recent approaches based on the traditional platforms. The results reveal the promise and potential of WebRTC for such a real-time telepresence wheelchair system.

The main findings of the first experiment (telepresence wheelchair based on WebRTC) are that the connection time of WebRTC is faster with less latency than the Skype platform (Corke, Findlater & Murphy 2012; Denojean-Mairet et al. 2014a; Ha, Nguyen & Nguyen 2016b). The WebRTC processing is done at the user level, which reduces the delay in processing and connection time. Moreover, the transmission rate of WebRTC is able to adapt to the available bandwidth of the network, whereas Skype platform adjusts the video resolution to adapt to the network condition (Corke, Findlater & Murphy 2012; Denojean-Mairet et al. 2014a; Ha, Nguyen & Nguyen 2016b). More importantly, WebRTC is open source and easily accessible. The convergence capabilities of the WebRTC technology can provide flexibility for further development and extension. However, the performance may be constrained by the hardware of the system. The video quality relies on the Internet connection bandwidth and the number of the participants who are conferring. Nevertheless, the development of the telepresence wheelchair using WebRTC has proven to be a potential alternative approach for telepresence systems with an open source and low-cost.

In the second experiment (an advanced telepresence wheelchair with wide field view using a dual-fisheye camera), due to the unique telepresence wheelchair equipped with a fisheye camera for healthcare applications, it is hard to make a comparison between our approaches to previous appropriate works. However, regarding visibility support, we found that using such a fisheye camera could bring considerable benefits for a telepresence wheelchair. The 360-degree view of the wheelchair's surroundings makes it easier to control and to navigate the wheelchair as compared to traditional camera approaches. During the experiments, we also found that there are correlations between the stitching processing time and the quality of received video. Longer processing time might affect the video quality. This can be overcome by improving the stitching techniques. Nevertheless, our novel approach has proven the effectiveness of WebRTC for a 360-degree vision telepresence wheelchair. It is worth noting that the use of a dual-fisheye camera along with real-time communication can deliver everything surrounding the wheelchair in 360 degrees. The state-of-the-art technologies would enable various interesting applications including healthcare support and remote monitoring.

In the third experiment (QoE estimation of wireless video streaming using neural networks), a QoE estimation model for wireless video streaming over WeRTC for a telepresence wheelchair is evaluated. The QoE estimation model was developed using neural network principles for video sequences streamed over different wireless network conditions. The model developed can be used for online QoE estimation given measurable network factors such as bit rate, jitter, round-trip-time, and packet loss rate. The estimated QoE is essential for the telepresence wheelchair navigation.

In the fourth experiment (evaluation of remote control telepresence wheelchair over WebRTC and 360-degree vision), the effectiveness of the 360-degree vision and remote control are also confirmed through the results of the experiments in Figures 5.32-38. Overall, the experimental results show the telepresence wheelchair is able to perform in various environments. In the case of traveling straight ahead, the remote user has easily piloted the wheelchair to the target. Also, the wheelchair was able to be remotely controlled to follow the figure eight curve in two different environmental conditions consisting of indoor and outdoor environments.

Considering the research of this chapter, it is clear that the development of the telepresence wheelchair using WebRTC is a promising concept in the field of assistive technology. The objective of this thesis is achieved by the proposed approach based on WebRTC and 360-degree vision. There are many potential usages of the system to integrate with other healthcare monitoring systems. For future works, the focus work could be on optimization the hardware and software such that the system can be embedded for accomplishing a commercial telepresence wheelchair system.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has presented the development of a telepresence wheelchair. The thesis has made the key contributions to design and develop an advanced assistive system for people with disabilities. From the original idea at the beginning, the research started by doing a literature review to explore the diversity of telepresence and its applications in the research field. Then, various technical solutions and possibilities for developing both software and hardware for the telepresence wheelchair have been investigated. The proposed approaches have mainly focused on two aspects including the video streaming over wireless communication with a full field of view and remote control in real-time based on standard Internet protocols, the ubiquitous streaming protocol, and the advanced real-time streaming strategy. These approaches are crucial for developing the video streaming and remote controlling of the telepresence wheelchair.

Through the literature review, the research directions of the development of the telepresence wheelchair were explored and introduced for innovative assistive technology systems. To take full advantage of the existing research reports, Chapter 2 has presented the benefits and the shortcomings of the current telepresence robot and wheelchair solutions to find the research gap which motives us to propose appropriate solutions to develop for an innovative telepresence wheelchair. An extensive literature review in Chapter 2 has shown that there exist various approaches to developing telepresence systems. However, almost all the existing telepresence systems equipped with traditional cameras are not flexible and are cumbersome when the systems are operated in the reverse direction. These findings have motivated us to propose the 360-degree vision telepresence wheelchair solution. In terms of

developing advanced functions for the telepresence wheelchair with 360-degree vision, the work done in this doctoral study is unique.

Chapter 3 in the thesis has introduced the investigation and implementation of the video streaming approach for the telepresence wheelchair. The design of a telepresence wheelchair has been presented relying on the electric powered wheelchair equipped with add-on devices and a digital camera named Ladybug3. This work has investigated a technical solution for the telepresence wheelchair system to deliver the surrounding images to a remote location over a wireless network. The main contribution of Chapter 3 is the design of a prototype of a telepresence wheelchair with wireless video streaming based on JPEG panoramic images. It has also provided concept, architecture and implementable framework for the early stage of this research. From the experiments, it is feasible to say that the telepresence wheelchair performance highly relies on the transmission protocols and the associated environments.

Experimental results in chapter 3 showed that the proposed method successfully streamed the video based on panoramic images over wireless communication with high-quality performance. The average testing results of the peak signal to noise ratio (PSNR) were 39.19 dB whereas the lowest PSNR performance of this approach was 29.78 dB and the highest PSNR performance was 43.71 dB. Comparing the performance of the proposed approach with the previous studies of video quality evaluation revealed the quality of the image is described as excellent for $\text{PSNR} \geq 37 \text{ dB}$ and is good for $32 \text{ dB} \leq \text{PSNR} < 37 \text{ dB}$ which confirmed that the outcome of wireless video streaming from the wheelchair platform has very considerable potential. The system performance evaluation of the proposal is meaningful for the overall success of the system implementation.

Chapter 4 has described the work for solving the streaming speed problems of the telepresence wheelchair. The major contribution of this chapter is to develop an effective video streaming approach using cross-platform frameworks. It commences with an implementation of a ubiquitous framework and multiple video streaming of the telepresence wheelchair. The ubiquitous framework based design is important for developing a telepresence wheelchair with a full field of view. Based on this framework, a

multiple video streaming and remote control strategy have been developed for the telepresence wheelchair system. Technically, the video streaming technique and the remote control method could be effectively integrated into the cross-platform framework.

In technical terms, the experiments showed that the Skype framework provided a very efficient solution for increasing the transmission speed and reducing control problems. Thanks to this technique, the design procedure was simplified. After being developed, a user-friendly control interface was implemented as a third party plugging for each independent subsystem. Real-time implementation of the telepresence wheelchair system revealed that the quality of video streaming performance of the scheme was increased significantly.

Real-time implementation of the telepresence wheelchair and the real-time results also showed that the ubiquitous framework based design successfully streams the video with high-speed and high-quality in real-time in various environmental conditions. The streaming rate in all scenarios was between 25 and 30 fps. The round-trip time varies from 3 to 271 milliseconds (ms) for local wireless connections. It is worth noting that the round trip time of less than 400 ms is considered as a real-time video conference. In addition, the wheelchair can successfully navigate at different distances with the average accuracy of 99.25 %. The real-time results also showed that the system could be controlled from the remote location successfully in various environmental conditions.

Moreover, Chapter 4 also involved the virtual reality experience of the telepresence wheelchair. The experiments were conducted by combining the virtual reality technology and a multiple video streaming. In the experiment using virtual reality, the overall average frame rate of video streaming was 29.11 ± 0.52 fps, and the average of peak signal to noise was 37.06 ± 0.67 dB. The wheelchair can be successfully navigated at various distances with the average accuracy of $97.72 \pm 0.91\%$. The experiments showed that the system was successfully controlled from a remote location, and the trajectory of the wheelchair was acceptable when compared with that of the joystick operation. The overall results in this chapter revealed that the proposed integration of technologies for developing an Internet-

enabled wheelchair is feasible. The telepresence wheelchair enables a remote user to control the mobility of the wheelchair without the help of others.

Chapter 5 has presented the development of the telepresence wheelchair with 360-degree vision. By exploiting the advancements in the field of emerging information technology and digital cameras, efficient technology integration for developing the telepresence wheelchair with 360-degree vision has been explored. The key idea in Chapter 5 is to develop an effective video streaming approach using WebRTC. The contribution is that the system does not only provide a prototype for a telepresence wheelchair for a healthcare assistant but also adapts and keeps up with the advancement of emerging information technology. Also, WebRTC technology contributes its advantages in dealing with real-time communication as well as developing an independent application. The results of this research have made a significant contribution to the area of telepresence. The resulting real-time data transferring using WebRTC also is significant in the field of information technology and telecommunications. Moreover, the integration of WebRTC technology and dual-fisheye video streaming led to a convergence of multimedia, robotics, human-machine interaction, and biomedical engineering in real-time applications. The advantage of this design is to provide the flexibility in control. A user-friendly interface and 360-degree field of view were designed to allow users to access the system and control the wheelchair easily and efficiently. The practical experiments indicated that the proposed technique is successfully streaming and navigating the wheelchair over wireless communication with excellent performance qualities. The demonstration of developing a full field of vision for a telepresence wheelchair is feasible. The effectiveness of the real-time 360-degree video streaming was confirmed via the real-time telepresence experiments in various environmental conditions.

The effectiveness of the WebRTC was confirmed via the real-time telepresence wheelchair experiments. The experimental results showed that the streaming video at a resolution of 640×480 pixels could reach a peak rate of 30 fps. The round-trip time significantly decreased and fluctuated from 3 to 20 ms. In addition, the average frame rate of streaming the 360-degree video with the resolution of 1280×720 pixels was 25.83 frames per second. The average peak signal to noise ratio (PSNR) was 29.06 dB. In various environmental

conditions, the results also demonstrated that the telepresence wheelchair with 360-degree vision could be very effective for the remote user to control the wheelchair while it follows the targets. In the case of traveling straight ahead, the remote user has easily piloted the wheelchair to the target. Also, the wheelchair was able to be remotely controlled to follow the figure eight curve in two different environmental conditions consisting of the indoor and outdoor environments. It can be concluded that the proposed approach of a 360-degree view of the telepresence wheelchair is feasible.

Furthermore, a QoE estimation model for wireless video streaming over WeRTC for a telepresence wheelchair was presented. The model was developed using neural network principles for multiple video sequences streamed and bit rates in different network conditions. The developed model can be used for online QoE estimation given measurable network factors which are necessary for the remote user to control the wheelchair towards reliable wireless signal coverage. Overall, the experimental results confirmed the effectiveness of our approach for the telepresence wheelchair with a full field of view.

Overall, an advanced telepresence wheelchair with a complete telepresence system with a wide field of view and remote navigation has been designed. Several techniques are involved in the process of completing the design in real-time and real hardware. Various hardware systems have been investigated from six integrated-cameras inside the Ladybug camera in Chapter 3, four cameras in Chapter 4 to the dual-fisheyes camera in Chapter 5. Moreover, we have studied the advanced streaming protocols of the telepresence wheelchair based on Internet protocol, cross-platform framework improved to independent and open source protocol. A number of experiments were conducted to evaluate the developed telepresence wheelchair system. The outcomes of the experimental results were positive. The overall performance results convince us that our telepresence wheelchair system is feasible and can be developed to become the next generation of the assistive system. Even though numerous methods have been introduced for developing the telepresence wheelchair system to tackle the challenging in real-time communication, there remain some interesting works to further explore in the future.

6.2 Future Work

Although the thesis has made significant improvements for the telepresence wheelchair, some further works beyond the current scope of the research efforts could be possibly researched in the future in terms of both hardware and software to further enhance the system performance. The first extended work would be to integrate obstacle avoidance algorithms into the telepresence wheelchair system. These will allow the wheelchair to adapt to various situations during wheelchair navigation and make the wheelchair safe and flexible when operated by the remote control. The second research direction would be suggested to improve the additional functionalities of the telepresence system by considering the transmission of feedback information and bio-signals to enable real-time health monitoring. The real-time health monitoring would be of fundamental importance and a practically very useful application.

In this doctoral study, the telepresence wheelchair with remote control strategy has been designed with user-friendly interface to accommodate people with various levels of disabilities. However, the experiments were only tested with able-body participants who played and acted as the disabled people. Despite the shortcomings of the tests, it is believed that the results are positive and reliable, and straightforward future works would be to conduct experiments involving people with disabilities in a hospital location or elderly care center over a period of time. The testing results could be then used as an important information source to refine the telepresence design.

Finally, in this thesis, we have accomplished the development of the telepresence wheelchair by using an onboard personal computer attached to the back of the chair. The use of the personal computer has made the design and implementation of the telepresence wheelchair more convenient and flexible. However, the onboard personal computer may result in the telepresence wheelchair with a high cost. Therefore, it would be better to convert the design into the embedded systems in order to design a low-cost telepresence wheelchair system for the commercial product.

Appendix A

Power Wheelchair Components

A.1 Motors



Figure A.1: Wheelchair gear-motors

A.2 Control Module -DX-RemG80

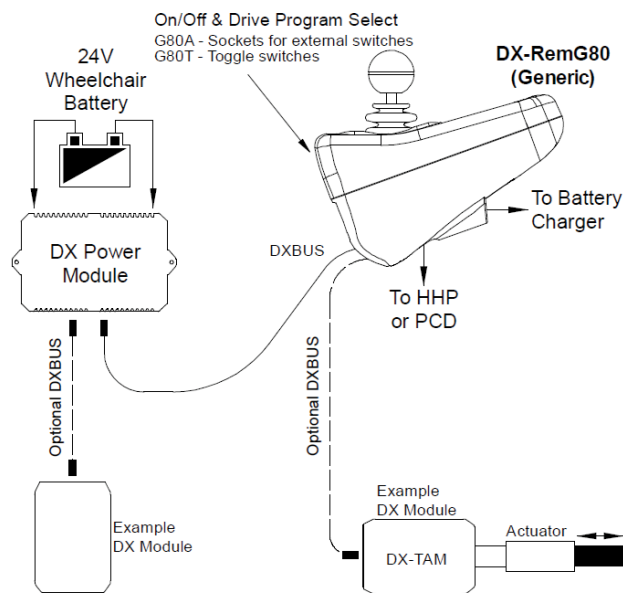


Figure A.2: DX System.

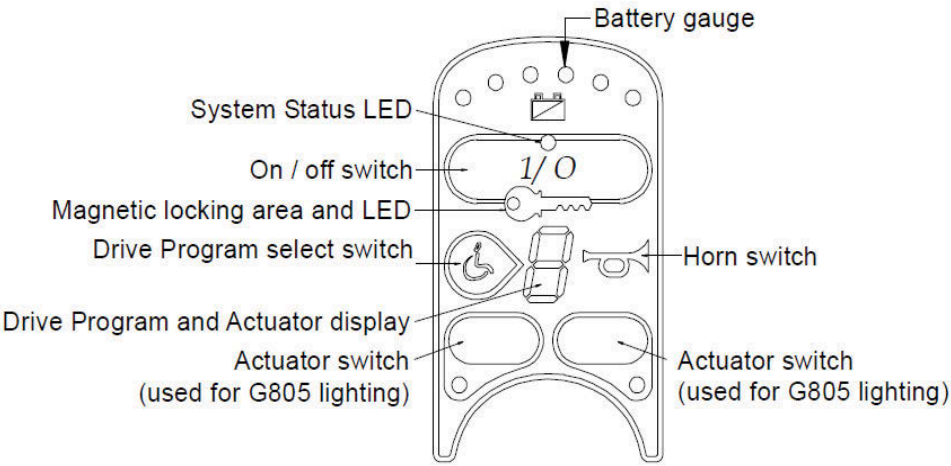


Figure A.3: Keypad of the wheelchair

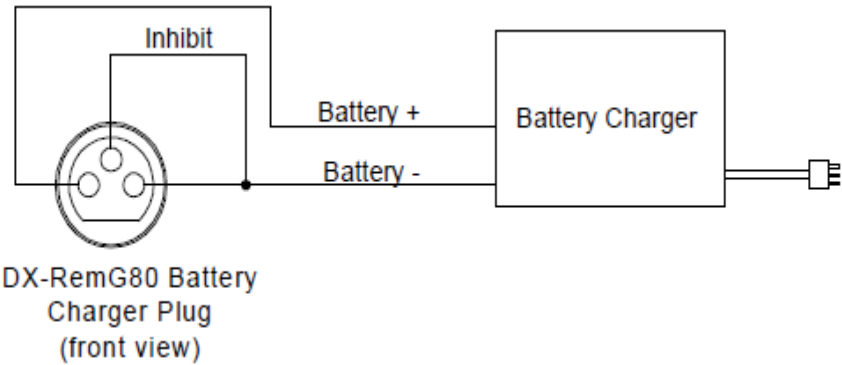


Figure A.4: Battery charging.

A. 3 NI USB 6008/6009

The National Instruments USB-6008/6009 devices provide eight single-ended analog input (AI) channels, two analog output (AO) channels, 12 DIO channels, and a 32-bit counter with a full-speed USB interface. The following table compares the NI USB-6008 and NI USB-6009 devices.

Table A.1. NI USB-6008 and NI USB-6009 Comparison

Feature	NI USB-6008	NI USB-6009
AI resolution	12 bits differential, 11 bits single-ended	14 bits differential, 13 bits single-ended
Maximum AI sample rate, single channel	10 kS/s	48 kS/s
Maximum AI sample rate, multiple channels (aggregate)	10 kS/s	48 kS/s
DIO configuration	Open collector	Each channel individually programmable as open collector or active drive

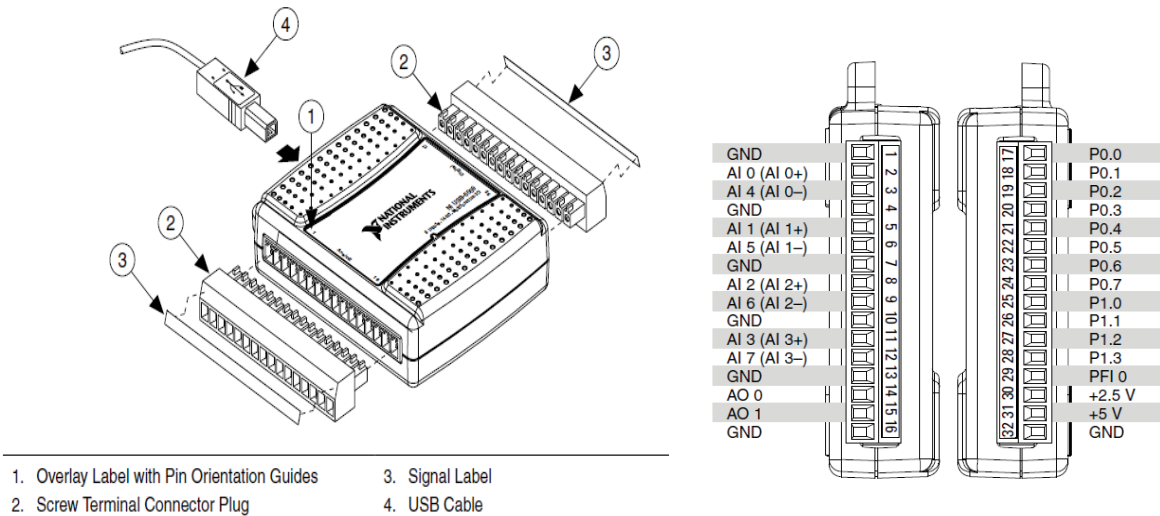


Figure A.5: Signal label application diagram and pin out

Table A.2. Signal Descriptions

Signal Name	Reference	Direction	Description
GND	-	-	Ground —The reference point for the single-ended analog input measurements, analog output voltages, digital signals, +5 VDC supply, and +2.5 VDC at the I/O connector, and the bias current return point for differential mode measurements.
AI <0..7>	Varies	Input	Analog Input Channels 0 to 7 —For single-ended measurements, each signal is an analog input voltage channel. For differential measurements, AI 0 and AI 4 are the positive and negative inputs of differential analog input channel 0. The following signal pairs also form differential input channels: AI <1, 5>, AI <2, 6>, and AI <3, 7>.
AO <0, 1>	GND	Output	Analog Output Channels 0 and 1 —Supplies the voltage output of AO channel 0 or AO channel 1.
P0.<0..7>	GND	Input or Output	Port 0 Digital I/O Channels 0 to 7 —You can individually configure each signal as an input or output.
P1.<0..3>	GND	Input or Output	Port 1 Digital I/O Channels 0 to 3 —You can individually configure each signal as an input or output.
PFI 0	GND	Input	PFI 0 —This pin is configurable as either a digital trigger or an event counter input.
+2.5 V	GND	Output	+2.5 V External Reference —Provides a reference for wrap-back testing.
+5 V	GND	Output	+5 V Power Source —Provides +5 V power up to 200 mA.

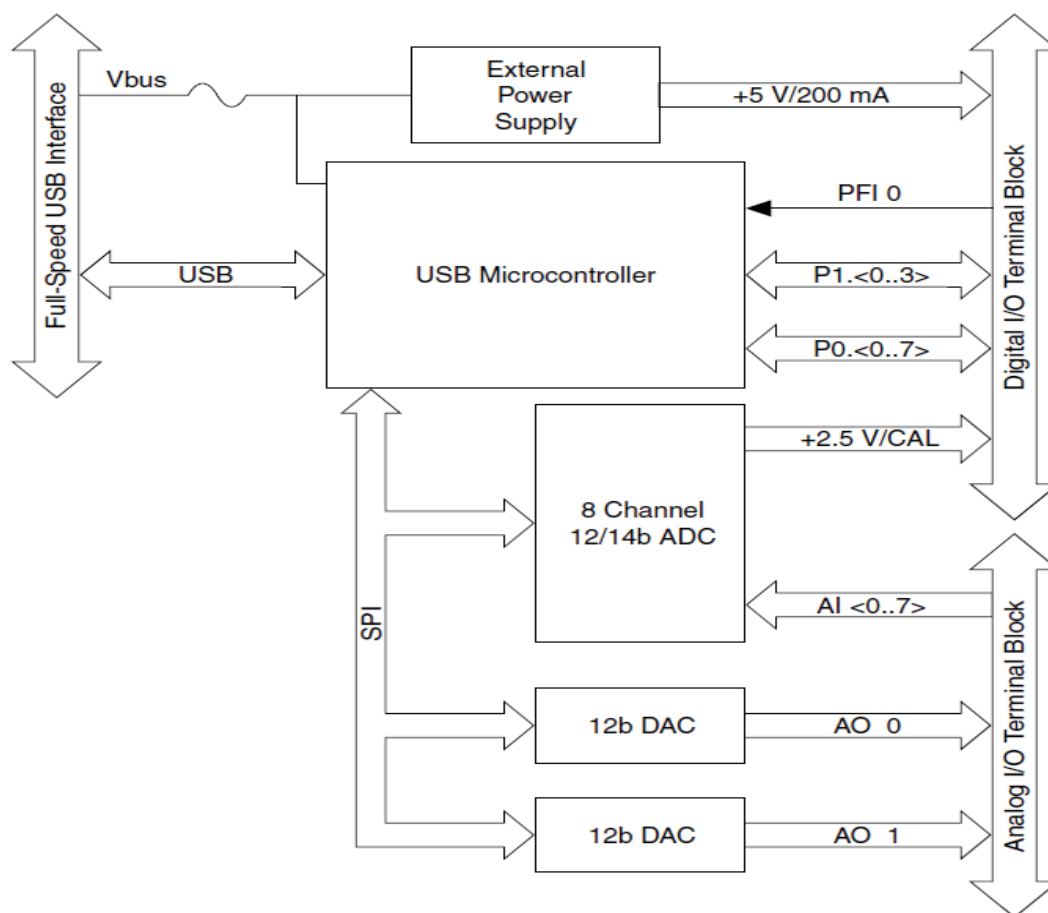


Figure A.6: NI USB-6008/6009 block diagram.

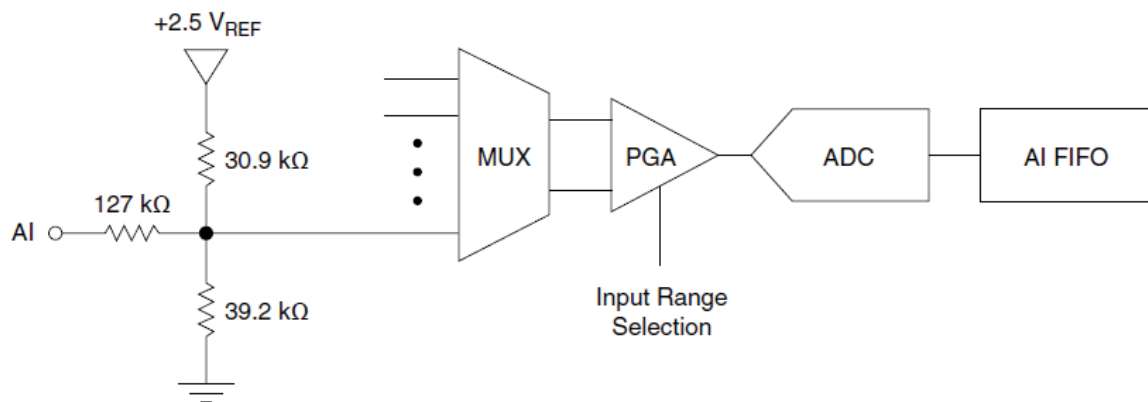


Figure A.7: NI USB-6008/6009 analog input circuitry.

Appendix B

Selected Software Code

B.1 Image Acquisition

```
// System Includes
//=====
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <assert.h>
#include <iostream>
#include <Windows.h>
#include <cstdio>
#include <ctime>
//=====
// PGR Includes
//=====
#include "ladybug.h"
#include "ladybuggeom.h"
#include "ladybugrenderer.h"
#include "ladybugstream.h"

//=====
// Macro Definitions
//=====

#define _HANDLE_ERROR \
    if( error != LADYBUG_OK ) \
    { \
        printf( "Error! Ladybug library reported %s\n", \
        ::ladybugErrorToString( error ) ); \
        assert( false ); \
        goto _EXIT; \
    }

#define IMAGES_TO_GRAB 10

//// The size of the stitched image
#define PANORAMIC_IMAGE_WIDTH 2048
#define PANORAMIC_IMAGE_HEIGHT 1024
int PANORAMIC_IMAGE_WIDTH;
int PANORAMIC_IMAGE_HEIGHT;

// handle CPU memory
static ULARGE_INTEGER lastCPU, lastSysCPU, lastUserCPU;
static int numProcessors;
static HANDLE self;
```



```

#define COLOR_PROCESSING_METHOD    LADYBUG_DOWNSAMPLE4    // The fastest color
method suitable for real-time usages
//#define COLOR_PROCESSING_METHOD    LADYBUG_HQLINEAR      // High quality method
suitable for large stitched images

//
// This function reads an image from camera
//
LadybugError
startCamera( LadybugContext context)
{
    // Initialize camera
    printf( "Initializing camera...\n" );
    LadybugError error = ladybugInitializeFromIndex( context, 0 );
    if (error != LADYBUG_OK)
    {
        return error;
    }

    // Get camera info
    printf( "Getting camera info...\n" );
    LadybugCameraInfo caminfo;
    error = ladybugGetCameraInfo( context, &caminfo );
    if (error != LADYBUG_OK)
    {
        return error;
    }

    // Load config file
    printf( "Load configuration file...\n" );
    error = ladybugLoadConfig( context, NULL);
    if (error != LADYBUG_OK)
    {
        return error;
    }

    // Set the panoramic view angle
    error = ladybugSetPanoramicViewingAngle( context, LADYBUG_FRONT_0_POLE_5);
    if (error != LADYBUG_OK)
    {
        return error;
    }

    // Make the rendering engine use the alpha mask
    error = ladybugSetAlphaMasking( context, true );
    if (error != LADYBUG_OK)
    {
        return error;
    }

    // Set color processing method.
    error = ladybugSetColorProcessingMethod( context, COLOR_PROCESSING_METHOD );
    if (error != LADYBUG_OK)
    {
        return error;
    }
}

```

```

// Configure output images in Ladybug library
printf( "Configure output images in Ladybug library...\n" );
error = ladybugConfigureOutputImages( context, LADYBUG_PANORAMIC );
if (error != LADYBUG_OK)
{
    return error;
}

error = ladybugSetOffScreenImageSize(
    context,
    LADYBUG_PANORAMIC,
    PANORAMIC_IMAGE_WIDTH,
    PANORAMIC_IMAGE_HEIGHT );
if (error != LADYBUG_OK)
{
    return error;
}

switch( caminfo.deviceType )
{
case LADYBUG_DEVICE_COMPRESSOR:
case LADYBUG_DEVICE_LADYBUG3:
case LADYBUG_DEVICE_LADYBUG5:
printf( "Starting Ladybug camera...\n" );
error = ladybugStart(
    context,
    LADYBUG_DATAFORMAT_COLOR_SEP_JPEG8);
break;

case LADYBUG_DEVICE_LADYBUG:
default:
printf( "Unsupported device.\n");
error = LADYBUG_FAILED;
}

return error;
}

double getCurrentValue(){
    FILETIME ftime, fsys, fuser;
    ULARGE_INTEGER now, sys, user;
    double percent;

    GetSystemTimeAsFileTime(&ftime);
    memcpy(&now, &ftime, sizeof(FILETIME));

    GetProcessTimes(self, &ftime, &ftime, &fsys, &fuser);
    memcpy(&sys, &fsys, sizeof(FILETIME));
    memcpy(&user, &fuser, sizeof(FILETIME));
    percent = (sys.QuadPart - lastSysCPU.QuadPart) +
        (user.QuadPart - lastUserCPU.QuadPart);
    percent /= (now.QuadPart - lastCPU.QuadPart);
    percent /= numProcessors;
    lastCPU = now;
    lastUserCPU = user;
}

```

```

        lastSysCPU = sys;

        return percent * 100;
    }
//=====
// Main Routine
//=====
int
main( int argc, char* argv[] )
{
    LadybugContext context = NULL;
    LadybugError error;
    unsigned int uiRawCols = 0;
    unsigned int uiRawRows = 0;
    int retry = 10;
    LadybugImage image;
    char savedPath[256] = ".";
    char prefixName[100] = "image";
    char width[5];
    char height[5];
    if (argc >=2) {
        strcpy(savedPath,argv[1]);
    }

    if (argc >=3) {
        strcpy(prefixName,argv[2]);
    }

    if (argc >=4) {
        strcpy(width,argv[3]);
        PANORAMIC_IMAGE_WIDTH = atoi(width);
    }

    if (argc >=5) {
        strcpy(height,argv[4]);
        PANORAMIC_IMAGE_HEIGHT = atoi(height);
    }
    // create ladybug context
    printf( "Creating ladybug context...\n" );
    error = ladybugCreateContext( &context );
    if ( error != LADYBUG_OK )
    {
        printf( "Failed creating ladybug context. Exit.\n" );
        return (1);
    }

    // Initialize and start the camera
    error = startCamera( context);
    _HANDLE_ERROR

    // Grab an image to inspect the image size
    printf( "Grabbing image...\n" );
    error = LADYBUG_FAILED;
    while ( error != LADYBUG_OK && retry-- > 0)

```

```

{
    error = ladybugGrabImage( context, &image );
}
_HANDLE_ERROR

// Set the size of the image to be processed
if (COLOR_PROCESSING_METHOD == LADYBUG_DOWNSAMPLE4 ||
    COLOR_PROCESSING_METHOD == LADYBUG_MONO)
{
    uiRawCols = image.uiCols / 2;
    uiRawRows = image.uiRows / 2;
}
else
{
    uiRawCols = image.uiCols;
    uiRawRows = image.uiRows;
}

// Initialize alpha mask size - this can take a long time if the
// masks are not present in the current directory.

printf( "Initializing alpha masks (this may take some time)...\n" );
error = ladybugInitializeAlphaMasks( context, uiRawCols, uiRawRows );
_HANDLE_ERROR

// handle CPU memory
SYSTEM_INFO sysInfo;
FILETIME ftime, fsys, fuser;

GetSystemInfo(&sysInfo);
numProcessors = sysInfo.dwNumberOfProcessors;

GetSystemTimeAsFileTime(&ftime);
memcpy(&lastCPU, &ftime, sizeof(FILETIME));

self = GetCurrentProcess();
GetProcessTimes(self, &ftime, &ftime, &fsys, &fuser);
memcpy(&lastSysCPU, &fsys, sizeof(FILETIME));
memcpy(&lastUserCPU, &fuser, sizeof(FILETIME));

_HANDLE_ERROR

// Process loop
printf( "Images Generation Loop...\n" );

bool exit = false;
while(exit == false)
{

    if (GetAsyncKeyState(VK_ESCAPE))
    {
        exit = true;
    }
}

```

```
        for (int i = 0; i < IMAGES_TO_GRAB; i++)
        {

            // printf( "Processing panoromic frame  %d ->", i);

            // Grab an image from the camera
            error = ladybugGrabImage(context, &image);
            _HANDLE_ERROR

            // Convert the image to 6 RGB buffers
            error = ladybugConvertImage(context, &image, NULL);
            _HANDLE_ERROR

            // Send the RGB buffers to the graphics card
            error = ladybugUpdateTextures(context, LADYBUG_NUM_CAMERAS, NULL);
            _HANDLE_ERROR

            // Stitch the images (inside the graphics card) and retrieve the output to
            the user's memory
            LadybugProcessedImage processedImage;
            error = ladybugRenderOffScreenImage(context, LADYBUG_PANORAMIC, LADYBUG_BGR,
            &processedImage);
            _HANDLE_ERROR

            // Save the output image to a file

            if(getCurrentValue() > 95)
                {exit = true;}

            // Save the output image to a file
            char pszOutputName[256];

            // get current time
            std::time_t rawtime;
            std::tm* timeinfo;
            char bufferTime [80];

            std::time(&rawtime);
            timeinfo = std::localtime(&rawtime);

            std::strftime(bufferTime,80,"%Y_%m_%d_%H_%M_%S",timeinfo);
            std::puts(bufferTime);

            sprintf( pszOutputName, "%s\\%s%s.jpg",savedPath,prefixName, bufferTime);

            printf( "Processing and Writing image:");

            // printf( "Processing and Writing image %s\n", pszOutputName);
            //Sleep(200);

            error = ladybugSaveImage( context, &processedImage, pszOutputName,
            LADYBUG_FILEFORMAT_JPG );
            _HANDLE_ERROR
        }
    }
```

```

    }

    printf("Done.\n");
_EXIT:

    // clean up
    //

    ladybugStop( context );
    ladybugDestroyContext( &context );

    // printf("<PRESS ESCAPE ANY KEY TO EXIT>");
    // _getch();
    return 0;
}

//=====

// Client

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace ClientReceiveImage
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private Socket socket = null;
            //int connectStatus = 0;
            //int total_received = 0;
            private void ConnectToServer()
            {
                int res = 1;
                try
                {
                    socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

```

```
// fill in the remote IP
IPAddress IP = IPAddress.Parse(txtServerIP.Text);
int port = int.Parse(txtPort.Text);
IPEndPoint IPE = new IPEndPoint(IP, port);

Console.WriteLine("started connection service ....");
// connect to the server
socket.Connect(IPE);
res = 2;
SetEnableButtonStop(true);
SetTextButtonStart("Start Connection");
IsStopReceive = false;
//Thread thread = new Thread(new ThreadStart(ReceiveData));
//thread.Start();
ReceiveData();

}
catch (SocketException ex)
{
    MessageBox.Show("Can not connect to server: " + txtServerIP.Text);
    res = 1;
}

if (res == 1)
{
    SetEnableButtonStop(true);
    SetTextButtonStart("Start Connection");
    PerformClickStop();
}
}

private const int defaultPackage = 100;
private int imageSize = 0;
private void IsAlive()
{
    while (true)
    {
        if (!socket.Connected)
        {
            MessageBox.Show("Disconnect from server: " + txtServerIP.Text);
            break;
        }
    }
}

private void ReceiveData()
{
    while (true)
    {
        List<byte[]> lstByte = new List<byte[]>();

        byte[] buffer = new byte[0];
        int index = 0;
        int packageSize = defaultPackage;
        bool beginReceive = false;
        int pos = 0;
        while (true)
```



```

        {
            byte[] tmp_buffer = new byte[packageSize + 1];

            if (imageSize != 0 && index != 0 && index + packageSize >
imageSize)
            {
                tmp_buffer = new byte[imageSize - index + 1];
            }
            if (!socket.Connected)
            {
                MessageBox.Show("Disconnect from server: " +
txtServerIP.Text);
                break;
            }

            // socket.ReceiveTimeout = 1000;

            socket.ReceiveTimeout = 2000;

            while (true)
            {
                if (IsStopReceive)
                {
                    break;
                }
                try
                {
                    int receiveLenght = 0;
                    if (!beginReceive)
                    {
                        socket.Receive(tmp_buffer, tmp_buffer.Length,
SocketFlags.None);
                    }
                    while (receiveLenght < tmp_buffer.Length &&
beginReceive)
                    {
                        byte[] receiveBuffer = new byte[packageSize + 1];
                        try
                        {
                            int receive = socket.Receive(receiveBuffer,
receiveBuffer.Length, SocketFlags.None);
                            Buffer.BlockCopy(receiveBuffer, 0, tmp_buffer,
receiveLenght, receive);
                            //receiveBuffer.CopyTo(tmp_buffer,
receiveLenght);

                            receiveLenght += receive;
                            //tmp_buffer = receiveBuffer;
                        }
                        catch { }
                    }

                    break;
                }
                catch { }
            }
        }
    }

```

```

        if (IsStopReceive)
        {
            break;
        }
        var msg = Encoding.Unicode.GetString(tmp_buffer);
        if (msg.IndexOf("BEGIN_IMAGE:") >= 0)
        {
            string split = msg.Replace("BEGIN_IMAGE:", "").Replace("\0",
""");

            string[] info = split.Split('|');
            imageSize = int.Parse(info[0]);
            packageSize = int.Parse(info[1]);
            buffer = new byte[imageSize];
            beginReceive = true;
            tmp_buffer = new byte[packageSize + 1];
            socket.Send(Encoding.Unicode.GetBytes("1"));
        }
        else if (msg.IndexOf("DISCONNECT:") >= 0)
        {
            PerformClickStop();
        }
        else if (msg.IndexOf("CLOSE:") >= 0)
        {
            MessageBox.Show("Server was closed, this application will be
closed too.");
            Application.Exit();
        }
        else if (beginReceive)
        {
            //tmp_buffer.CopyTo(buffer, index);
            pos++;
            lstByte.Add(tmp_buffer);
            index += tmp_buffer.Length-1;
            int percent = (int)(((double)index / imageSize) * 100);
            //SetProgressBar(percent);
            if (index >= imageSize)
            {
                buffer = BuidBufferImage(lstByte, imageSize, pos);
                packageSize = defaultPackage;
                socket.Send(Encoding.Unicode.GetBytes("3"));
                //SetProgressBar(100);
                break;
            }
            socket.Send(Encoding.Unicode.GetBytes("2"));
        }
    }
}
if (!IsStopReceive)
{
    Console.WriteLine("Receive success");
    MemoryStream ms = new MemoryStream();
    string a = string.Join("", buffer);
    ms.Write(buffer, 0, buffer.Length);
    Image img = Image.FromStream(ms);
    picShow.Image = img;
    //

```

```
        Thread.Sleep(10000);
        ms.Close();
        // total_received++;
        //SetProgressBar(total_received);
    }
    else
    {
        break;
    }
}
}
private byte[] BuidBufferImage(List<byte[]> lstBuffer, int size, int pos)
{
    byte[] buffer = new byte[size];
    int index = 0;
    for (int i = 0; i < pos; i++)
    {
        foreach (byte[] bytes in lstBuffer)
        {
            int first = bytes[0];
            if(first == i)
            {
                byte[] temp = new byte[bytes.Length-1];
                Buffer.BlockCopy(bytes, 1, temp, 0, temp.Length);
                temp.CopyTo(buffer, index);
                index += bytes.Length-1;
            }
        }
    }

    return buffer;
}
delegate void SetProgressBarCallback(int value);

//////////

delegate void ClickStopCallback();
private void PerformClickStop()
{
    if (this.btnStop.InvokeRequired)
    {
        ClickStopCallback d = new ClickStopCallback(PerformClickStop);
        this.Invoke(d, new object[] { });
    }
    else
    {
        this.btnStop.PerformClick();
    }
}
delegate void SetTextButtonStartCallBack(string value);
private void SetTextButtonStart(string value)
{
    if (this.btnStart.InvokeRequired)
    {
        SetTextButtonStartCallBack d = new
SetTextButtonStartCallBack(SetTextButtonStart);
        this.Invoke(d, new object[] { value });
    }
}
```

```
        }
        else
        {
            this.btnStart.Text = value;
        }
    }
    delegate void SetEnableButtonStopCallBack(bool value);
    private void SetEnableButtonStop(bool value)
    {
        if (this.btnStop.InvokeRequired)
        {
            SetEnableButtonStopCallBack d = new
SetEnableButtonStopCallBack(SetEnableButtonStop);
            this.Invoke(d, new object[] { value });
        }
        else
        {
            this.btnStop.Enabled = value;
        }
    }
    private bool IsStopReceive = false;
    private void btnStart_Click(object sender, EventArgs e)
    {
        btnStart.Enabled = false;
        btnStart.Text = "Connecting...";
        //btnStop.Enabled = true;
        Thread thread = new Thread(new ThreadStart(ConnectToServer));
        thread.Start();
    }

    private void btnStop_Click(object sender, EventArgs e)
    {
        btnStart.Enabled = true;
        btnStop.Enabled = false;
        IsStopReceive = true;
        picShow.Image = null;
        if (socket.Connected)
        {
            socket.Send(Encoding.Unicode.GetBytes("CLIENTSHUTDOWN:" +
socket.LocalEndPoint.ToString().Split(':')[0]));

            socket.Shutdown(SocketShutdown.Both);
            socket.BeginDisconnect(true, new AsyncCallback(DisconnectCallback),
socket);
        }
    }
    private void DisconnectCallback(IAsyncResult ar)
    {
        Socket client = (Socket)ar.AsyncState;
        client.EndDisconnect(ar);
    }
    private void Form1_Load(object sender, EventArgs e)
    {
    }
}
```

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        if (socket != null && socket.Connected)
        {
            socket.Send(Encoding.Unicode.GetBytes("CLIENTSHUTDOWN:" +
socket.LocalEndPoint.ToString().Split(':')[0]));
        }
    }
    catch { }
}
```

// Server

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Collections.ObjectModel;

namespace ServerTransferImage
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private string FolderPath = string.Empty;
            private const int packageSize = 20000;

            private const string extension = ".jpg";

            public string LocalIPAddress()
            {
                IPEndPoint host;
                string localIP = "";
                host = Dns.GetHostEntry(Dns.GetHostName());
                foreach (IPAddress ip in host.AddressList)
                {
```

```
        if (ip.AddressFamily == AddressFamily.InterNetwork)
        {
            localIP = ip.ToString();
            break;
        }
    }
    return localIP;
}

// Camera Activate

private void CallCamera()
{
    string pathFolder = Application.StartupPath +
    "\\..\\..\\..\\Camera\\CaptureImage";
    FolderPath = pathFolder;
    string prefixName = "image";
    Process p = new Process();
    string appPath = Application.StartupPath + "\\..\\..\\..\\Camera";
    p.StartInfo.FileName = string.Format("{0}\\Imageacquisition.exe",
appPath);
    p.StartInfo.Arguments = "\"" + pathFolder + "\" " + prefixName + " "
+ _width + " " + _height;
    //p.StartInfo.RedirectStandardOutput = true;
    p.StartInfo.UseShellExecute = true;
    p.StartInfo.CreateNoWindow = false;
    p.Start();
}

private string localIP = "";

private void Form1_Load(object sender, EventArgs e)
{
    localIP = LocalIPAddress();
    lblLocalIP.Text = "Server IP: " + localIP;

    // Choose Image Resolutions

    cbxResolution.Items.Add("512x256");
    cbxResolution.Items.Add("1024x512");
    cbxResolution.Items.Add("2048x1024");
    cbxResolution.Items.Add("3500x1750");
    cbxResolution.Items.Add("9600x1080");

    // Default

    cbxResolution.SelectedIndex = 2;
}

private Thread threadCamera = null;
private Thread threadSocket = null;
private Thread threadSendImage = null;
private void btnStart_Click(object sender, EventArgs e)
{
    isStop = false;
```

```

        btnStart.Enabled = false;
        btnStop.Enabled = true;
        //run camera

        threadCamera = new Thread(new ThreadStart(CallCamera));
        threadCamera.Start();

        //start socket

        threadSocket = new Thread(new ThreadStart(StartListening));
        threadSocket.Start();

        //start send image
        if (threadSendImage!=null)
        {
            threadSendImage.Abort();
        }
        threadSendImage = new Thread(new ThreadStart(ReadLoopImage));
        threadSendImage.Start();
    }
    private List<Socket> lstClientConnectionSocket = new List<Socket>();
    private Socket sListen = null;
    private void StartListening()
    {
        // Create a socket
        sListen = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

        // Fill IP
        IPAddress IP = IPAddress.Parse(localIP);
        int port = int.Parse(txtPort.Text);
        IPEndPoint IPE = new IPEndPoint(IP, port);

        // Binding
        sListen.Bind(IPE);

        // Monitor
        Console.WriteLine("Service is listening ...");
        sListen.Listen(2);
        //}
        // Loop to accept client connection requests
        while (true)
        {
            Socket clientSocket;
            try
            {
                clientSocket = sListen.Accept();
                if (!isStop)
                {
                    lstClientConnectionSocket.Add(clientSocket);
                    AddClientList(clientSocket.RemoteEndPoint.ToString().Split(':')[0]);
                }
            }
        }
    }

```

```
        catch
        {
        }
    }

    private delegate void AddClientListCallBack(string clientIP);
    private void AddClientList(string clientIP)
    {
        if (this.lstClient.InvokeRequired)
        {
            AddClientListCallBack d = new AddClientListCallBack(AddClientList);
            this.Invoke(d, new object[] { clientIP });
        }
        else
        {
            this.lstClient.Items.Add(clientIP);
        }
    }

    private delegate void DeleteClientListCallBack(string clientIP);
    private void DeleteClientList(string clientIP)
    {
        if (this.lstClient.InvokeRequired)
        {
            AddClientListCallBack d = new
AddClientListCallBack(DeleteClientList);
            this.Invoke(d, new object[] { clientIP });
        }
        else
        {
            for (int i = 0; i < this.lstClient.Items.Count; i++)
            {
                if (this.lstClient.Items[i].ToString() == clientIP)
                {
                    this.lstClient.Items.RemoveAt(i);
                    break;
                }
            }
        }
    }

    // Read images

    private void ReadLoopImage()
    {
        while (true)
        {
            if (!isStop && lstClientConnectionSocket.Count > 0)
            {
                bool clientDisconnect = false;
                int index = 0;
                for (int i = 0; i < lstClientConnectionSocket.Count; i++)
                {
                    Socket socket = lstClientConnectionSocket[i];
```



```

        byte[] tmp = new byte[100];
        socket.ReceiveTimeout = 1000;
        try
        {
            socket.Receive(tmp, tmp.Length, SocketFlags.None);
            string msg = Encoding.Unicode.GetString(tmp);
            if (msg.IndexOf("CLIENTSHUTDOWN:") >= 0)
            {
                msg = msg.Replace("\0", "");
                string clientIP = msg.Split(':')[1];
                DeleteClientList(clientIP);
                clientDisconnect = true;
                socket.Shutdown(SocketShutdown.Both);
                break;
            }
        }
        catch
        {
        }
        socket.ReceiveTimeout = 0;
        int res = ReadImageFromFolder(FolderPath, extension,
socket);

        if (res == 1)
        {
            try
            {
                IPEndPoint ip = (IPEndPoint)socket.RemoteEndPoint;
                string clientIP = ip.Address.ToString();
                DeleteClientList(clientIP);
                clientDisconnect = true;
                socket.Shutdown(SocketShutdown.Both);
                break;
            }
            catch { }
        }
        index++;
    }
    if (clientDisconnect)
    {
        lstClientConnectionSocket.RemoveAt(index);
    }
}
else if (lstClientConnectionSocket.Count <= 0)
{
    DeleteImagesInFolder(FolderPath, extension);
}
}

// Send image

private int SendOneImage(string path, Socket clientSocket)
{
    int res=0;

```

```
try
{
    // send the file
    byte[] buffer = ReadImageFile(path);
    string a = string.Join("", buffer);
    // send data to the client
    clientSocket.Send(Encoding.Unicode.GetBytes("BEGIN_IMAGE:" +
buffer.Length.ToString() + "|" + packageSize.ToString() + "|"));
    while (true)
    {
        byte[] tmp_buffer = new byte[2];
        try
        {
            clientSocket.Receive(tmp_buffer, tmp_buffer.Length,
SocketFlags.None);
        }
        catch (SocketException ex)
        {
            clientSocket.Connect(clientSocket.LocalEndPoint);
        }
        var msg = Encoding.Unicode.GetString(tmp_buffer);
        msg = msg.Replace("\0", "");
        if (msg == "1")
            break;
    }
    res = SendFile(path, clientSocket, buffer.Length);

    Console.WriteLine("Send success!");
}
catch (SocketException ex)
{
    res = 1;
}
catch (Exception ex)
{
    res = 2;
}
return res;
}

// Delete image

private void DeleteImagesInFolder(string path, string fileExtension)
{
    if (Directory.Exists(path))
    {
        List<string> listFileNames =
            new List<string>(Directory.GetFiles(path, "*" +
fileExtension));
        if (listFileNames.Count > 1)
        {
            List<FileInfo> lstFileInfo = new List<FileInfo>();
            foreach (var listFileName in listFileNames)
            {
                FileInfo info = new FileInfo(listFileName);
                lstFileInfo.Add(info);
            }
        }
    }
}
```

```
        List<FileInfo> sorted = lstFileInfo.OrderBy(o =>
o.CreationTimeUtc).ToList();
        //listFileNames.Sort();
        sorted.Reverse();

        foreach (FileInfo fullFilePath in sorted)
        {
            if (sorted.IndexOf(fullFilePath) == 1 && currentImage !=
fullFilePath.CreationTimeUtc)
            {
                if (fullFilePath.Exists)
                {
                    string filename = fullFilePath.FullName;
                    picShow.Load(filename);
                    currentImage = fullFilePath.CreationTimeUtc;
                }
            }
            else if (sorted.IndexOf(fullFilePath) == 0)
                continue;
            else
                deleteImages(fullFilePath.FullName);
        }
    }
}

private DateTime currentImage = new DateTime();
private int ReadImageFromFolder(string path, string fileExtension, Socket
clientSocket)
{
    int res = 0;
    List<string> listFileNames =
        new List<string>(Directory.GetFiles(path, "*" + fileExtension));
    if (listFileNames.Count > 1)
    {
        List<FileInfo> lstFileInfo = new List<FileInfo>();
        foreach (var listFileName in listFileNames)
        {
            FileInfo info = new FileInfo(listFileName);
            lstFileInfo.Add(info);
        }
        List<FileInfo> sorted = lstFileInfo.OrderBy(o =>
o.CreationTimeUtc).ToList();
        //listFileNames.Sort();
        sorted.Reverse();

        foreach (FileInfo fullFilePath in sorted)
        {
            if (sorted.IndexOf(fullFilePath) == 1 && currentImage !=
fullFilePath.CreationTimeUtc && clientSocket!=null)
            {
                if (fullFilePath.Exists)
                {
                    string filename = fullFilePath.FullName;
                    picShow.Load(filename);
                    //send picture
                    res = SendOneImage(filename, clientSocket);
                }
            }
        }
    }
}
```

```
        currentImage = fullPath.CreationTimeUtc;
        if (res == 1)
        {
            break;
        }
    }
}
else if (sorted.IndexOf(fullFilePath) == 0 &&
clientSocket!=null)
    continue;
else
    deleteImages(fullFilePath.FullName);
}
}
return res;
}
private void deleteImages(string filePath)
{
    try
    {
        File.Delete(filePath);
    }
    catch { }
}

private static byte[] ReadImageFile(String img)
{
    FileInfo fileInfo = new FileInfo(img);
    byte[] buf = new byte[fileInfo.Length];
    FileStream fs = new FileStream(img, FileMode.Open, FileAccess.Read);
    fs.Read(buf, 0, buf.Length);
    fs.Close();
    GC.RegisterForFinalize(fileInfo);
    GC.RegisterForFinalize(fs);
    return buf;
}
private int SendFile(String img, Socket clientSocket, int imageSize)
{
    try
    {
        FileStream fs = new FileStream(img, FileMode.Open, FileAccess.Read);
        int countSentSize = 0;
        int pos = 0;
        List<byte[]> lstBuffer = new List<byte[]>();
        while (true)
        {
            bool endoffile = false;
            byte[] buffer = new byte[packageSize];
            int space = packageSize, read, offset = 0;
            if (countSentSize + packageSize > imageSize)
            {
                buffer = new byte[imageSize - countSentSize];
                space = buffer.Length;
            }

            while (space > 0 && (read = fs.Read(buffer, offset, space)) > 0)
            {
```

```

        space -= read;
        offset += read;
    }
    // either a full buffer, or EOF
    if (space == 0 && offset == 0)
    {
        break;
    }
    // full buffer

    byte[] sendBuffer = new byte[buffer.Length + 1];
    sendBuffer[0] = (byte)pos;
    //Buffer.BlockCopy(buffer, 0, sendBuffer, 1, buffer.Length);
    buffer.CopyTo(sendBuffer, 1);
    lstBuffer.Add(sendBuffer);
    clientSocket.Send(sendBuffer, sendBuffer.Length,
SocketFlags.None);
    pos++;
    countSentSize += buffer.Length;
    while (true)
    {
        byte[] status = new byte[2];
        clientSocket.ReceiveTimeout = 0;
        clientSocket.Receive(status, status.Length,
SocketFlags.None);

        string msg = Encoding.Unicode.GetString(status);
        if (msg == "2")
        {
            break;
        }
        else if (msg == "3")
        {
            endoffile = true;
            break;
        }
    }
    if (endoffile)
    {
        break;
    }
}
return 0; //send ok
}
catch (SocketException ex)
{
    return 1; //socket error
}
catch (Exception ex)
{
    return 2; //unexpected
}
}

private bool isStop = false;
private void btnStop_Click(object sender, EventArgs e)
{
    btnStart.Enabled = true;

```

```

        btnStop.Enabled = false;
        sListen.Close();
        foreach (Socket socket in lstClientConnectionSocket)
        {
            socket.Send(Encoding.Unicode.GetBytes("DISCONNECT:"));
        }
        threadCamera.Abort();
        threadSocket.Abort();
        KillImageAcquistionProcess();
    }

    private int _width, _height = 0;
    private void cbxResolution_SelectedIndexChanged(object sender, EventArgs e)
    {
        string selectedText = cbxResolution.SelectedItem.ToString();
        if (!string.IsNullOrEmpty(selectedText))
        {
            string[] parts = selectedText.Split('x');

            _width = int.Parse(parts[0]);
            _height = int.Parse(parts[1]);
        }
    }

    private void Form1_FormClosing(object sender, FormClosingEventArgs e)
    {
        foreach (Socket socket in lstClientConnectionSocket)
        {
            socket.Send(Encoding.Unicode.GetBytes("CLOSE:"));
        }
    }

    private void KillImageAcquistionProcess()
    {
        Collection<Process> prs = new
Collection<Process>(Process.GetProcesses());
        foreach (Process pr in prs)
        {
            if (pr.ProcessName.ToLower().IndexOf("imageacquistion") >= 0)
            {
                pr.Kill();
                return;
            }
        }
    }
}

```

B.2 DAQ

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```
//using System.Threading.Tasks;
using NationalInstruments;
using NationalInstruments.DAQmx;

namespace Wheelchair
{
    public class DAQApp
    {
        public void btstop_Click()
        {
            double Voltagevalue0, Voltagevalue1;

            Voltagevalue0 = 2.5;
            Voltagevalue1 = 2.5;

            Task analogOut0 = new Task();
            Task analogOut1 = new Task();

            if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
            {
                string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
                string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

                AOChannel Chane1A00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "Chane1A00", 0, 5,
AOVoltageUnits.Volts);
                AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);
                writer0.WriteSingleSample(true, Voltagevalue0);

                AOChannel Chane1A01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "Chane1A01", 0, 5,
AOVoltageUnits.Volts);
                AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);
                writer1.WriteSingleSample(true, Voltagevalue1);
            }

        }

        public void btforward_Click()
        {
            double Voltagevalue0, Voltagevalue1;

            Voltagevalue0 = 2.77;
            Voltagevalue1 = 2.5;

            Task analogOut0 = new Task();
```

```
Task analogOut1 = new Task();

        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External)[1].ToString();

            AOChannel ChanelA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChanelA00", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

            writer0.WriteSingleSample(true, Voltagevalue0);
            AOChannel ChanelA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChanelA01", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

            writer1.WriteSingleSample(true, Voltagevalue1);
        }

    }

    public void btleft_Click()
    {
        double Voltagevalue0, Voltagevalue1;

        Voltagevalue0 = 2.5;
        Voltagevalue1 = 2.14;

        Task analogOut0 = new Task();
        Task analogOut1 = new Task();

        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.A0,
PhysicalChannelAccess.External)[1].ToString();
```



```

        AOChannel ChaneIA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChaneIA00", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

        writer0.WriteSingleSample(true, Voltagevalue0);

        AOChannel ChaneIA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChaneIA01", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

        writer1.WriteSingleSample(true, Voltagevalue1);
    }
}

public void btright_Click()
{
    double Voltagevalue0, Voltagevalue1;

    Voltagevalue0 = 2.5;
    Voltagevalue1 = 2.9;

    Task analogOut0 = new Task();
    Task analogOut1 = new Task();

    if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
    {
        string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
        string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

        AOChannel ChaneIA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChaneIA00", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

        writer0.WriteSingleSample(true, Voltagevalue0);
        AOChannel ChaneIA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChaneIA01", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

        writer1.WriteSingleSample(true, Voltagevalue1);
    }
}

```

```

    }

    public void btback_Click()
    {
        double Voltagevalue0, Voltagevalue1;

        Voltagevalue0 = 2.0;
        Voltagevalue1 = 2.5;

        Task analogOut0 = new Task();
        Task analogOut1 = new Task();

        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

            AOChannel ChanelA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChanelA00", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

            writer0.WriteSingleSample(true, Voltagevalue0);

            AOChannel ChanelA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChanelA01", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

            writer1.WriteSingleSample(true, Voltagevalue1);
        }
    }
}

```

```
//=====
```

```
// Control
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Wheelchair
{
    class Controller
    {
        PictureBox left, right, up, down, stop;

        public delegate void CMD(string data);
        public event CMD onCMD;
        public string mode = "Normal";

        public Controller(PictureBox left, PictureBox right, PictureBox up,
            PictureBox down, PictureBox stop)
        {
            this.left = left;
            this.right = right;
            this.up = up;
            this.down = down;
            this.stop = stop;
        }

        public void Reset(string mode)
        {
            switch (mode)
            {
                case "Master":
                    left.Image = global::Wheelchair.Properties.Resources.left;
                    right.Image = global::Wheelchair.Properties.Resources.right;
                    down.Image = global::Wheelchair.Properties.Resources.down;
                    up.Image = global::Wheelchair.Properties.Resources.up;
                    stop.Image = global::Wheelchair.Properties.Resources.stop;
                    break;

                case "Normal":
                    left.Image = global::Wheelchair.Properties.Resources.left_black;
                    right.Image =
global::Wheelchair.Properties.Resources.right_black;
                    down.Image = global::Wheelchair.Properties.Resources.down_black;
                    up.Image = global::Wheelchair.Properties.Resources.up_black;
                    stop.Image = global::Wheelchair.Properties.Resources.stop_black;
                    break;
            }
        }

        public void setDirection(string direction, bool flag)
        {
            switch (direction)
            {

```

```

        {
            case "Left":
                Reset(mode);
                left.Image =
global::Wheelchair.Properties.Resources.left_press;
                if (onCMD != null && flag) onCMD("cmdLEFT");
                break;
            case "Right":
                Reset(mode);
                right.Image =
global::Wheelchair.Properties.Resources.right_press;
                if (onCMD != null && flag) onCMD("cmdRIGHT");
                break;
            case "Stop":
                Reset(mode);
                if (onCMD != null && flag) onCMD("cmdSTOP");
                break;
            case "Up":
                Reset(mode);
                up.Image = global::Wheelchair.Properties.Resources.up_press;
                if (onCMD != null && flag) onCMD("cmdUP");
                break;
            case "Down":
                Reset(mode);
                down.Image =
global::Wheelchair.Properties.Resources.down_press;
                if (onCMD != null && flag) onCMD("cmdDOWN");
                break;
        }
    }

    public void DoUp(bool directionFlag)
    {
        DAQApp daqInstance = new DAQApp();
        daqInstance.btforward_Click();
        setDirection("Up", directionFlag);
    }

    public void DoLeft(bool directionFlag)
    {
        DAQApp daqInstance = new DAQApp();
        daqInstance.btleft_Click();
        setDirection("Left", directionFlag);
    }

    public void DoRight(bool directionFlag)
    {
        DAQApp daqInstance = new DAQApp();
        daqInstance.btright_Click();
        setDirection("Right", directionFlag);
    }

    public void DoDown(bool directionFlag)
    {
        DAQApp daqInstance = new DAQApp();
        daqInstance.btback_Click();
        setDirection("Down", directionFlag);
    }

```

```

    }

    public void DoStop(bool directionFlag)
    {
        DAQApp daqInstance = new DAQApp();
        daqInstance.btstop_Click();
        setDirection("Stop", directionFlag);
        stop.Image = global::Wheelchair.Properties.Resources.stop_press;
    }
}

//=====
// Plugin

using SKYPE4COMLib;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;

namespace Wheelchair
{
    public class SkypeEx
    {

        public SkypeEx()
        {
            if (parent == 0) parent = MyWIN32.FindWindow("tSkMainForm", null);
        }

        public int counter = 100;
        int parent, child1, child2, child3;

        Thread mThread;

        public string Checked(string contact)
        {
            SkypeHWND hwnd = new SkypeHWND();
            if (hwnd.GetHWND_child(0, parent, "TConversationForm", contact) == 0)
return "Unchecked";
            else return "Checked";
        }

        public string NewCall()
        {
            if (parent == 0) parent = MyWIN32.FindWindow("tSkMainForm", null);

```

```

        if (parent != 0)
        {
            MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_HOTKEY, "");
            MyWIN32.SetForegroundWindow((IntPtr)parent);
            SendKeys.Send("^%r");
            return "Call [DIALING]";
        }
        return "Call [DIALING FAILED]";
    }

    public string VideoToggle()
    {
        if (parent == 0) parent = MyWIN32.FindWindow("tSkMainForm", null);
        if (parent != 0)
        {
            SendKeys.Send("^%{PGUP}");
            MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, "");

            return "Video button toggled";
        }
        return "tSkMainForm not found";
    }

    public void AddGroup(List<string> Contact)
    {
        SkypeHWND hwnd = new SkypeHWND();

        if (parent == 0) parent = MyWIN32.FindWindow("tSkMainForm", null);
        if (parent != 0)
        {
            Debug.WriteLine("Clicked");
            MyWIN32.SetForegroundWindow((IntPtr)parent);
            MyWIN32.SendMessage((IntPtr)parent, MyWIN32.WM_LBUTTONDOWN,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(15, 15));
            MyWIN32.SendMessage((IntPtr)parent, MyWIN32.WM_LBUTTONUP,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(15, 15));
            SendKeys.Send("^n");
            SendKeys.Send("^+a");
            Thread.Sleep(1000);
            parent = MyWIN32.FindWindow("TSelectContactsPopup_AddPeople", "");
            child1 = hwnd.GetHWND_child(0, parent, "TPopupContainer", "");
            child2 = hwnd.GetHWND_child(0, child1, "TLabelledEdit", "");

            string total = String.Empty;
            foreach (string line in Contact)
            {
                MyWIN32.SendMessage((IntPtr)child2, MyWIN32.WM_SETTEXT, 0, "_" +
line);
                MyWIN32.sendKey((IntPtr)child2, Keys.Enter, false);
            }

            MyWIN32.SendMessage((IntPtr)child1, MyWIN32.WM_LBUTTONDOWN,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(156, 282));

```

```

        MyWIN32.SendMessage((IntPtr)child1, MyWIN32.WM_LBUTTONUP,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(156, 282));

    }
}

public string Scan(string username, string contact)
{
    SkypeHWND hwnd = new SkypeHWND();

    // Find contact -----
    if (parent == 0) parent = MyWIN32.FindWindow("tSkMainForm", null);
    child1 = hwnd.GetHWND_child(0, parent, "TSearchControl", "");
    child2 = hwnd.GetHWND_child(0, child1, "TAccessibleEdit", "");

    if (parent != 0 && child1 != 0 && child2 != 0)
    {
        MyWIN32.SendMessage((IntPtr)child2, MyWIN32.WM_LBUTTONDOWN,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(0, 0));
        MyWIN32.SendMessage((IntPtr)child2, MyWIN32.WM_LBUTTONUP,
(IntPtr)(long)0x0, MyWIN32.MakeLParam(0, 0));
        MyWIN32.SendMessage((IntPtr)child2, MyWIN32.WM_SETTEXT, 0, contact);
        MyWIN32.sendKey((IntPtr)child2, Keys.Down, false);
        Thread.Sleep(50);
        MyWIN32.sendKey((IntPtr)child2, Keys.Enter, false);
        Thread.Sleep(50);

        return contact;
    }
    else return "ERROR!";
}

public List<SkypeContact> GetContacts(String type, Skype skype)
{
    List<SkypeContact> Contacts = new List<SkypeContact>();
    string fullName = string.Empty;
    string displayName = string.Empty;

    for (int i = 0; i < skype.HardwiredGroups.Count; i++)
    {
        if (skype.HardwiredGroups[i + 1].Type == TGroupType.grpAllFriends)
        {
            for (int j = skype.HardwiredGroups[i + 1].Users.Count; j > 0; j-
-)
            {
                if (skype.HardwiredGroups[i +
1].Users[j].FullName.Contains("Echo"))
                {
                }
                else
                {

```

```

1].Users[j].FullName;
1].Users[j].DisplayName;
1].Users[j].FullName;
1].Users[j].Handle

        fullName = skype.HardwiredGroups[i +
        displayName = skype.HardwiredGroups[i +
        if (string.IsNullOrEmpty(fullName))
        {
            fullName = displayName;
        }
        if (string.IsNullOrEmpty(displayName))
        {
            fullName = skype.HardwiredGroups[i +
            displayName = fullName;
        }
        Contacts.Add(new SkypeContact()
        {
            FullName = fullName,
            DisplayName = displayName,
            Handle = skype.HardwiredGroups[i +
        }));
    }
}
}
return Contacts;
}

//=====
// Main Form
using SKYPE4COMLib;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using AForge.Video.FFMPEG;
using System.Net.NetworkInformation;
using SharpDX.DirectInput;
using InTheHand.Net;
using InTheHand.Net.Sockets;

namespace Wheelchair
{
    public partial class Form1 : Form

```



```

{

    #region form status
    string activeDirectionControlName;
    #endregion

    Controller mController;

    COM.COM SerialPort;
    COM.Constants mConstants;

    //-----
- Skype initiation
    Skype mSky;
    Call call;
    SkypeEx mSkyEx;

    bool loop = false;

    //List<SkypeContact> Handle_Name = new List<SkypeContact>();
    List<SkypeContact> Full_Name = new List<SkypeContact>();
    //List<SkypeContact> Display_Name = new List<SkypeContact>();
    //static string ContactType = "";
    //-----
- Skype initiation

    public static bool ConnectToInternet(int timeout_per_host_millis = 1000,
string[] hosts_to_ping = null)
    {
        bool network_available =
System.Net.NetworkInformation.NetworkInterface.GetIsNetworkAvailable();

        if (network_available)
        {
            string[] hosts = hosts_to_ping ?? new string[] { "www.google.com",
"www.facebook.com" };

            Ping p = new Ping();

            foreach (string host in hosts)
            {
                try
                {
                    PingReply r = p.Send(host, timeout_per_host_millis);

                    if (r.Status == IPStatus.Success)
                        return true;
                }
                catch { }
            }
        }

        return false;
    }

    private bool SkypeCheck()

```

```
{

    try
    {
        int counter;
        if (mSky.Client.IsRunning)
        {
            counter = mSky.ActiveCalls.Count; // trigger
            return true;
        }
        else return false;
    }

    catch (NullReferenceException err)
    {
        Debug.WriteLine("SkypeCheck error: " + err.Message);
        return false;
    }
    catch (COMException err)
    {
        Debug.WriteLine("SkypeCheck error: " + err.Message);
        return false;
    }
}

public Form1()
{
    mSky = new Skype();
    mSkyEx = new SkypeEx();

    InitializeComponent();

}

private void Form1_Load(object sender, EventArgs e)
{
    //Always on top
    this.TopMost = true;

    if (SkypeCheck())
    {
        mSky.Attach(8, true);
        mSky.MessageStatus += new
        _ISkypeEvents_MessageStatusEventHandler(mSky_MessageStatus);
        mSky.CallStatus += new
        _ISkypeEvents_CallStatusEventHandler(mSky_CallStatus);
        mSky.MessageHistory += new
        _ISkypeEvents_MessageHistoryEventHandler(mSky_MessageHistory);
        mSky.CallVideoStatusChanged += new
        _ISkypeEvents_CallVideoStatusChangedEventHandler(mSky_CallVideoStatusChanged);
        mSky.CallVideoReceiveStatusChanged += new
        _ISkypeEvents_CallVideoReceiveStatusChangedEventHandler(mSky_CallVideoReceiveStatusC
hanged);
    }
}
```

```

        mSky.CallVideoSendStatusChanged += new
_ISkypeEvents_CallVideoSendStatusChangedEventHandler(mSky_CallVideoSendStatusChanged
);
        bt_contact_scan.PerformClick();

    }
    else
    {
        MessageBox.Show("Please login Skype before starting Telepresence
Wheelchair", "Error!");
        if (System.Windows.Forms.Application.MessageLoop)
System.Windows.Forms.Application.Exit();
        else System.Environment.Exit(1);
    }

    UIInit();
}

void _SendSkypeCommand(string cmdToSend)
{
    Command cmdSkype = new Command();
    cmdSkype.Blocking = true;
    cmdSkype.Timeout = 2000;
    cmdSkype.Command = cmdToSend;
    Trace.WriteLineIf(true, string.Format("skype command sent '{0}'",
cmdToSend));
    mSky.SendCommand(cmdSkype);
}

void _hideSkypeWindows()
{
    _SendSkypeCommand("SET SILENT_MODE ON");
    _SendSkypeCommand("SET WINDOWSTATE HIDDEN");
}

private void UIInit()
{
    mController = new Controller(pb_left, pb_right, pb_up, pb_down,
pb_stop);
    mController.onCMD += MController_onCMD;

    cb_skype.Checked = true;
    //cb_COM.Checked = true;

    rtb_log.BackColor = Color.White;
    rtb_log.ForeColor = Color.Black;

    //cb_contact_type.Items.Add("Display Name");
    //cb_contact_type.Items.Add("Handle");
    //cb_contact_type.Items.Add("Full Name");
    //cb_contact_type.SelectedIndex = 0;
    //ContactType = "Full Name";
}

private void MController_onCMD(string data)

```

```

    {
        if (cb_debug.Checked) Update("RAW", "[Controller] " + data);

        //Off send message cmdLeft,...
        if (SkypeCheck() && cb_skype.Checked && cb_contact.Text != "")
mSkyEx.Send1((string)cb_contact.SelectedValue, data);

        //if (cb_COM.Checked && SerialPort != null && SerialPort.isConnected())
        //{
        //    SerialPort.Send(data);
        //    Update("Skype1", mConstants.values[0] + ": " + data);
        //    mController.setDirection(data, false);
        //}
    }

void mSky_CallVideoSendStatusChanged(Call pCall, TCallVideoSendStatus
Status)
{
}

void mSky_CallVideoReceiveStatusChanged(Call pCall, TCallVideoSendStatus
Status)
{
}

void mSky_CallVideoStatusChanged(Call pCall, TCallVideoStatus Status)
{
}

void mSky_CallStatus(Call pCall, TCallStatus Status)
{
    if (Status == TCallStatus.clsRinging && (pCall.Type ==
TCallType.cltIncomingP2P || pCall.Type == TCallType.cltIncomingPSTN))
    {
        try
        {
            Update("Anounc", "Incoming call from " +
pCall.PartnerDisplayName + "....");
            Update("Anounc", mSkyEx.VideoToggle());
            if (mSkyEx.Scan("Whatever",
pCall.PartnerDisplayName).Equals("ERROR!")) Update("Anounc", "Couldn't load contact
" + pCall.PartnerDisplayName + " ==!");
            else Update("Anounc", "Selected: " + pCall.PartnerDisplayName);
        }
        catch (Exception e)
        {
            Update("Anounc", "Exception: " + e.Message);
        }
    }

    else if (Status == TCallStatus.clsInProgress) { pCall.StartVideoSend();
}

```

```

        else if (Status == TCallStatus.clsBusy) { Update("Anounc", "Call
[BUSY]"); FinishCall();
} // MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }
        else if (Status == TCallStatus.clsCancelled) { Update("Anounc", "Call
[CANCELLED]"); FinishCall(); }
// MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }
        else if (Status == TCallStatus.clsRefused) { Update("Anounc", "Call
[REFUSED]"); FinishCall();
} // MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }
        else if (Status == TCallStatus.clsFinished) { Update("Anounc", "Call
[FINISHED]"); FinishCall();
} // MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }
        else if (Status == TCallStatus.clsFailed) { Update("Anounc", "Call
[FAILED]"); FinishCall(); }
// MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }
        else if (Status == TCallStatus.clsRinging) {
} // MyWIN32.SendMessage((IntPtr)MyWIN32.FindWindow("tSkMainForm", null),
MyWIN32.WM_SYSCOMMAND, MyWIN32.SC_CLOSE, ""); }

    }

    void mSky_MessageHistory(string Username)
    {
        // do nothing
    }

    void mSky_MessageStatus(ChatMessage pMessage, TChatMessageStatus Status)
    {
        if (cb_debug.Checked) Update("RAW", "MessageStatus triggered: " +
Status);
        bool trigger = false;
        string RAW = "";

        try
        {
            RAW = pMessage.Body.ToString();
            Debug.WriteLine(DateTime.Now.ToLocalTime() + ": " +
"Message Status - Message Id: " + pMessage.Id +
" - Chat Friendly Name: " + pMessage.Chat.FriendlyName +
" - Chat Name: " + pMessage.Chat.Name +
" - Converted Message Type: " +
mSky.Convert.ChatMessageTypeToText(pMessage.Type) +
" - Message Type: " + pMessage.Type +
" - Converted TChatMessageStatus Status: " +
mSky.Convert.ChatMessageStatusToText(Status) +
" - TChatMessageStatus Status: " + Status +
" - From Display Name: " + pMessage.FromDisplayName +
" - From Handle: " + pMessage.FromHandle);

            if (pMessage.Chat.Topic.Length > 0) Debug.WriteLine(" - Topic: " +
pMessage.Chat.Topic);

```

```
        if (pMessage.Body.Length > 0) Debug.WriteLine(" - Body: " +
pMessage.Body);
    }
    catch (COMException e)
    {
        trigger = true;
        Debug.WriteLine("Error: " + e.Message); // Invalid chat name - WTF?!
        Update("RAW", "Error: " + e.Source + "/" + e.Message);
    }
    finally
    {
        if (trigger) Update("Skype", "[D] " + RAW);
    }

    if (Status == TChatMessageStatus.cmsSending)
Update("Skype1", pMessage.Body.ToString());

    else if (Status == TChatMessageStatus.cmsSent) {;}

    else if (Status == TChatMessageStatus.cmsReceived)
    {

        if (cb_skype.Checked &&
pMessage.Body.ToString().Contains("[Connection REQ]"))
        {
            // send [ECHO-RESPONSE]
            Update("Skype1", "[Connection RESPONSE]" + " [" +
DateTime.Now.ToString() + "]");
        }

        //if (cb_COM.Checked && SerialPort != null &&
SerialPort.isConnected())
        //{
            //    SerialPort.Send(pMessage.Body.ToString());
            //    Update("Skype1", mConstants.values[0] + ": " +
pMessage.Body.ToString());
        //}

        if (mController.mode == "Normal")
        {
            switch (pMessage.Body.ToString())
            {

                case "cmdLEFT":
                    mController.DoLeft(false);
                    //mController.setDirection("Left", false);
                    break;
                case "cmdRIGHT":
                    mController.DoRight(false);
                    //mController.setDirection("Right", false);
                    break;
                case "cmdDOWN":
                    mController.DoDown(false);
                    //mController.setDirection("Down", false);
                    break;
            }
        }
    }
}
```

```

        case "cmdUP":
            mController.DoUp(false);
            //mController.setDirection("Up", false);
            break;
        case "cmdSTOP":
            mController.DoStop(false);
            //mController.setDirection("Stop", false);
            break;
    }

    }
    Update("Skype", pMessage.FromDisplayName + ": " +
pMessage.Body.ToString());
    }
}

private void bt_clear_Click(object sender, EventArgs e)
{
    rtb_log.Text = String.Empty;
}

public delegate void Received(string type, string rx);

private void Update(string type, string rx)
{
    Invoke((MethodInvoker)delegate
    {
        if (rtb_log.InvokeRequired) Invoke(new Received(Update), new
object[] { type, rx });

        else
        {
            Font font = new Font("Arial", 10, FontStyle.Bold);
            rtb_log.SelectionStart = rtb_log.Text.Length;
            rtb_log.ScrollToCaret();
            rtb_log.SelectionFont = font;

            switch (type)
            {
                case "RAW":
                    rtb_log.SelectionColor = Color.Gray;
                    break;
                case "Skype":
                    rtb_log.SelectionColor = Color.BlueViolet;
                    break;
                case "Anounc":
                    rtb_log.SelectionColor = Color.Red;
                    break;
                case "Skype1":
                    rtb_log.SelectionColor = Color.Green;
                    break;
            }
            rtb_log.SelectedText = rx + "\n";
            rtb_log.ScrollToCaret();
        }
    });
}
}

```

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (!this.Focus()) this.Activate();
}

private void clearToolStripMenuItem_Click(object sender, EventArgs e)
{
    rtb_log.Text = String.Empty;
    rtb_log.ScrollToCaret();
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (cb_controller.Checked) cb_controller.Checked = false;
    if (System.Windows.Forms.Application.MessageLoop)
        System.Windows.Forms.Application.Exit();
    else System.Environment.Exit(1);
}

private void LoadCt()
{
    try
    {
        Update("Anounc", "Scanning... [" + DateTime.Now.ToString() + "]");
        Full_Name = mSkyEx.GetContacts("Full Name", mSky);

        Invoke((MethodInvoker)delegate
        {
            bt_contact_scan.Enabled = false;
            bt_call.Enabled = false;
            bt_hang.Enabled = false;
            cb_contact.Text = "";
            cb_contact.Tag = true;
            cb_contact.DataSource = Full_Name.OrderBy(c =>
                c.FullName).ToList();
            cb_contact.Tag = null;

            if (Full_Name.Count > 0) Update("Anounc",
                cb_contact.Items.Count + " Contact(s) added" + " [" + DateTime.Now.ToString() +
                "]");
            else Update("Anounc", "No Contact found" + " [" +
                DateTime.Now.ToString() + "]");

            bt_contact_scan.Enabled = true;
            bt_call.Enabled = true;
            bt_hang.Enabled = true;
        });
    }
    catch (COMException err)
    {
        Update("Anounc", "LoadCt error: " + err.Message);
    }
}

private void bt_contact_scan_Click(object sender, EventArgs e)
{

```

```
        if (SkypeCheck() )
        {
            Thread mThread = new Thread(new ThreadStart(LoadCt));
            mThread.Priority = ThreadPriority.AboveNormal;
            mThread.IsBackground = true;
            mThread.Name = "Load Skype";
            mThread.Start();

            bt_contact_scan.Enabled = false;
            bt_call.Enabled = false;
            bt_hang.Enabled = false;
        }
        else Update("Anounc", "Skype client not found!");
    }

    private void bt_call_Click(object sender, EventArgs e)
    {
        if (SkypeCheck() )
        {
            if (cb_contact.SelectedIndex > -1 && mSky.ActiveCalls.Count == 0)
            {
                //string temp = ((SkypeContact)cb_contact.SelectedItem).Handle;
                //Thread VidCall = new Thread(new ThreadStart(() =>
                this.VideoCall(temp));
                //VidCall.Start();
                Update("Anounc", mSkyEx.NewCall());
            }
        }
        else Update("Anounc", "Skype client not found!");
    }

    private void FinishCall()
    {
        try
        {
            if (mSky.ActiveCalls.Count > 0)
            {
                mSky.ActiveCalls[1].Finish();
            }
        }
        catch (COMException e)
        {
            Update("Anounc", "FinishCall error: " + e.Message);
        }
    }

    private void VideoCall(string id)
    {
        try
        {
            call = mSky.PlaceCall(id);
        }
        catch (COMException e)
        {
            Update("Anounc", "VideoCall error: " + e.Message);
        }
    }
}
```

```

    }
}

private void bt_hang_Click(object sender, EventArgs e)
{
    if (SkypeCheck() ) FinishCall();
    else Update("Anounc", "Skype client not found!");
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    System.Environment.Exit(1);
}

private void cb_contact_SelectedIndexChanged_1(object sender, EventArgs e)
{
    if (cb_contact.Tag != null)
    {
        cb_contact.SelectedIndex = -1;
        return;
    }
    if (!cb_contact.Text.Equals("") &&
mSkyEx.Checked(((SkypeContact)cb_contact.SelectedItem).Handle).Equals("Unchecked"))
    {
        string result = mSkyEx.Scan("Whatever",
(string)cb_contact.SelectedValue);
        if (result.Equals("ERROR!")) Update("Anounc", "ERROR OCCUR!");
        else Update("Anounc", "Selected: " + cb_contact.Text);
    }
    //else Update("Anounc", "Already deployed: " + cb_contact.Text);
}

private void COMToolStripMenuItem_Click(object sender, EventArgs e)
{
    mConstants = new COM.Constants();
    COM.COMUI Form = new COM.COMUI(mConstants);

    if (SerialPort != null && SerialPort.isConnected())
    {
        DialogResult dialog = MessageBox.Show(SerialPort.connectedPort() + "
is opened, close it?", "Warning ==!", MessageBoxButtons.OKCancel);
        if (dialog == DialogResult.OK)
        {
            SerialPort.Disconnect();
        }
        else if (dialog == DialogResult.No)
        {
            // do nothing
        }
    }
    else

```

```

        {
            Form.ShowDialog();
            if (cb_debug.Checked)
            {
                Update("RAW", "COM Port info.");
                Update("RAW", "values[0]: " + mConstants.values[0]);
                Update("RAW", "values[1]: " + mConstants.values[1]);
                Update("RAW", "values[2]: " + mConstants.values[2]);
                Update("RAW", "values[3]: " + mConstants.values[3]);
                Update("RAW", "values[4]: " + mConstants.values[4]);
                Update("RAW", ">_Initiating...");
            }

            if (mConstants.values[0] != null)
            {
                SerialPort = new COM.COM(mConstants.values);
                SerialPort.mReceiver += SerialPort_mReceiver;
                SerialPort.mSignal += SerialPort_mSignal;
                SerialPort.Connect();
            }
            else Update("Anounc", "No COM Port found!");
        }
    }

    void SerialPort_mSignal(string data)
    {
        Update("Anounc", data);
    }

    void SerialPort_mReceiver(string data)
    {
        Update("Skype", mConstants.values[0] + ": " + data);
        if (SkypeCheck() && cb_skype.Checked && cb_contact.Text != "")
            mSkyEx.Send1((string)cb_contact.SelectedValue, data);
    }

    private void cb_controller_CheckedChanged(object sender, EventArgs e)
    {
        if (cb_controller.Checked)
        {
            mController.Reset("Master");
            mController.mode = "Master";
        }
        else
        {
            mController.Reset("Normal");
            mController.mode = "Normal";
            mController.setDirection("Stop", true);
        }
    }

    private void timer1_Tick_1(object sender, EventArgs e)
    {
    }

```

```
private void pb_up_Click(object sender, EventArgs e)
{
    if (mController.mode == "Master" &&
!string.Equals(activeDirectionControlName, pb_up.Name))
    {
        activeDirectionControlName = pb_up.Name;
        mController.DoUp(true);
    }
}

private void pb_left_Click(object sender, EventArgs e)
{
    if (mController.mode == "Master" &&
!string.Equals(activeDirectionControlName, pb_left.Name))
    {
        activeDirectionControlName = pb_left.Name;
        mController.DoLeft(true);
    }
}

private void pb_right_Click(object sender, EventArgs e)
{
    if (mController.mode == "Master" &&
!string.Equals(activeDirectionControlName, pb_right.Name))
    {
        activeDirectionControlName = pb_right.Name;
        mController.DoRight(true);
    }
}

private void pb_down_Click(object sender, EventArgs e)
{
    if (mController.mode == "Master" &&
!string.Equals(activeDirectionControlName, pb_down.Name))
    {
        activeDirectionControlName = pb_down.Name;
        mController.DoDown(true);
    }
}

private void pb_stop_Click(object sender, EventArgs e)
{
    if (mController.mode == "Master" &&
!string.Equals(activeDirectionControlName, pb_stop.Name))
    {
        activeDirectionControlName = pb_stop.Name;
        mController.DoStop(true);
    }
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    Update("Anounc", "Clicked");
    if (mController.mode == "Master")
    {
        switch (e.KeyCode)
```

```
        {
            case Keys.Up:
                mController.setDirection("Up", true);
                break;

            case Keys.Left:
                mController.setDirection("Left", true);
                break;

            case Keys.Right:
                mController.setDirection("Right", true);
                break;

            case Keys.Down:
                mController.setDirection("Down", true);
                break;

            case Keys.Space:
                mController.setDirection("Stop", true);
                break;
        }
    }

    private void cb_btserver_CheckedChanged(object sender, EventArgs e)
    {
    }

    private void cb_skype_CheckedChanged(object sender, EventArgs e)
    {
    }

}
}
```

```
//=====
```

B.3 WebRTC Control

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
//using System.Threading.Tasks;
using NationalInstruments;
using NationalInstruments.DAQmx;

namespace WebControl
{
    public class DAQApp
    {
        public void btstop_Click()
        {
            double Voltagevalue0, Voltagevalue1;

            Voltagevalue0 = 2.5;
            Voltagevalue1 = 2.5;

            Task analogOut0 = new Task();
            Task analogOut1 = new Task();

            try
            {
                if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
                {
                    string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
                    string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

                    AOChannel Chane1A00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "Chane1A00", 0, 5,
AOVoltageUnits.Volts);
                    AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);
                    writer0.WriteSingleSample(true, Voltagevalue0);

                    AOChannel Chane1A01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "Chane1A01", 0, 5,
AOVoltageUnits.Volts);
                    AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);
                    writer1.WriteSingleSample(true, Voltagevalue1);
                }
            }
            catch
            { };
        }

        public void btforward_Click()
    }
}

```

```

    {
        double Voltagevalue0, Voltagevalue1;

        Voltagevalue0 = 2.77;
        Voltagevalue1 = 2.5;

        Task analogOut0 = new Task();
        Task analogOut1 = new Task();

        try
        {
            if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
            {
                string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
                string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

                AOChannel ChanelA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChanelA00", 0, 5,
AOVoltageUnits.Volts);
                AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

                writer0.WriteSingleSample(true, Voltagevalue0);
                AOChannel ChanelA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChanelA01", 0, 5,
AOVoltageUnits.Volts);
                AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

                writer1.WriteSingleSample(true, Voltagevalue1);
            }
        }
        catch
        { };
    }

    public void btleft_Click()
    {
        double Voltagevalue0, Voltagevalue1;

        Voltagevalue0 = 2.5;
        Voltagevalue1 = 2.14;

        Task analogOut0 = new Task();
        Task analogOut1 = new Task();

        try
        {

```

```

        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

            AOChannel ChannelA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChanelA00", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

            writer0.WriteSingleSample(true, Voltagevalue0);

            AOChannel ChannelA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChanelA01", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

            writer1.WriteSingleSample(true, Voltagevalue1);
        }
    }
    catch
    { };
}

public void btright_Click()
{
    double Voltagevalue0, Voltagevalue1;

    Voltagevalue0 = 2.5;
    Voltagevalue1 = 2.9;

    Task analogOut0 = new Task();
    Task analogOut1 = new Task();

    try
    {
        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 = DaqSystem.Local.GetPhysicalC 4
`ec hannels(PhysicalChannelTypes.AO, PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

```



```

        AOChannel ChaneIA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChaneIA00", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

        writer0.WriteSingleSample(true, Voltagevalue0);
        AOChannel ChaneIA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChaneIA01", 0, 5,
AOVoltageUnits.Volts);
        AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

        writer1.WriteSingleSample(true, Voltagevalue1);
    }
}
catch
{ };
}

public void btback_Click()
{
    double Voltagevalue0, Voltagevalue1;

    Voltagevalue0 = 2.0;
    Voltagevalue1 = 2.5;

    Task analogOut0 = new Task();
    Task analogOut1 = new Task();

    try
    {
        if (DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External).ToList().Count > 0)
        {
            string devA00 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[0].ToString();
            string devA01 =
DaqSystem.Local.GetPhysicalChannels(PhysicalChannelTypes.AO,
PhysicalChannelAccess.External)[1].ToString();

            AOChannel ChaneIA00 =
analogOut0.AOChannels.CreateVoltageChannel(devA00, "ChaneIA00", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer0 = new
AnalogSingleChannelWriter(analogOut0.Stream);

            writer0.WriteSingleSample(true, Voltagevalue0);

            AOChannel ChaneIA01 =
analogOut1.AOChannels.CreateVoltageChannel(devA01, "ChaneIA01", 0, 5,
AOVoltageUnits.Volts);
            AnalogSingleChannelWriter writer1 = new
AnalogSingleChannelWriter(analogOut1.Stream);

```

```
        writer1.WriteSingleSample(true, Voltagevalue1);
    }
}
catch
{ };
}
}
}
```

Bibliography

ABS, 2016, Disability, Ageing and Carers, Australia: Summary of Findings, 2015<<http://www.abs.gov.au/ausstats/abs@.nsf/0/C258C88A7AA5A87ECA2568A9001393E8?Opendocument>>, viewed 13 Feb 2017.

Ackerman, E. 2014, New Beam+ Telepresence System, IEEE Spectrum, 3 Park Avenue, New York, <<http://spectrum.ieee.org/automaton/robotics/home-robots/suitable-technologies-introduces-new-beam-telepresence-system>>, viewed 22 May 2014.

Adalgeirsson, S.O. & Breazeal, C. 2010, 'MeBot: A robotic platform for socially embodied telepresence', 2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 15-22.

Adel, E., Elmogy, M. & Elbakry, H. 2014, 'Image Stitching based on Feature Extraction Techniques: A Survey', International Journal of Computer Applications vol. 99, p. 8.

Aggarwal, R., Vohra, A. & Namboodiri, A.M. 2016, 'Panoramic Stereo Videos with a Single Camera', 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3755-63.

Ali, S. & Hussain, M. 2012, 'Panoramic image construction using feature based registration methods', 2012 15th International Multitopic Conference (INMIC), pp. 209-14.

Argyriou, L., Economou, D., Bouki, V. & Doumanis, I. 2016, 'Engaging Immersive Video Consumers: Challenges Regarding 360-Degree Gamified Video Applications', 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), pp. 145-52.

Atwah, R., Iqbal, R., Shirmohammadi, S. & Javadtalab, A. 2015, 'A Dynamic Alpha Congestion Controller for WebRTC,' 2015 IEEE International Symposium on Multimedia (ISM), pp. 132-5.

Baruffa, G. & Frescura, F. 2015, 'Adaptive Error Protection for the Streaming of Motion JPEG 2000 Video over Variable Bit Error Rate Channels', 2015 IEEE International Symposium on Multimedia (ISM), pp. 124-7.

Bolas, M., Iliff, J., Hoberman, P., Burba, N., Phan, T., McDowall, I., Luckey, P. & Krum, D.M. 2013, 'Open virtual reality,' 2013 IEEE Virtual Reality (VR), pp. 183-4.

Borvorntanajanya, K., Thiuthipsakul, P., Chalongwongse, S., Moonjaita, C. & Suthakorn, J. 2016, 'Development of differential suspension wheeled system for telepresence robot in rural hospital area,' 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1046-51.

Botia Valderrama, D.J., Luis, G. G., & mez, N. 2016, 'Nonintrusive Method Based on Neural Networks for Video Quality of Experience Assessment,' Advances in Multimedia, vol. 2016, p. 17.

Bourhis, G., Moumen, K., Pino, P., Rohmer, S. & Pruski, A. 1993, 'Assisted navigation for a powered wheelchair,' Proceedings of IEEE Systems Man and Cybernetics Conference - SMC, pp. 553-8 vol.3.

Cain, W., Bell, J. & Cheng, C. 2016, 'Implementing Robotic Telepresence in a Synchronous Hybrid Course,' 2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT), pp. 171-5.

Calin, D. 2014, Beam Plus: Telepresence Robot for Home, Into Robotics, viewed 22 May 2014, <<http://www.intorobotics.com/beam-plus-telepresence-robot-home/>>.

Chai, R., Ling, S.H., Hunter, G.P., Tran, Y. & Nguyen, H.T. 2014, 'Computer Interface Classifier for Wheelchair Commands Using Neural Network With Fuzzy Particle Swarm Optimization', *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 5, pp. 1614-24.

Chen, Y.-S., Lu, J.-M. & Hsu, Y.-L. 2013, 'Design and Evaluation of a Telepresence Robot for Interpersonal Communication with Older Adults', in *Inclusive Society: Health and Wellbeing in the Community, and Care at Home: 11th International Conference on Smart Homes and Health Telematics, ICOST 2013, Singapore, June 19-21, 2013. Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 298-303.

Chiang, C.Y., Chen, Y.L., Tsai, P.S. & Yuan, S.M. 2014, 'A Video Conferencing System Based on WebRTC for Seniors', *Trustworthy Systems and their Applications (TSA), 2014 International Conference on*, pp. 51-6.

Coradeschi, S., Loutfi, A., Kristoffersson, A., Cortellessa, G. & Eklundh, K.S. 2011, 'Social robotic telepresence,' *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pp. 5-6.

Corke, P., Findlater, K. & Murphy, E. 2012, 'Skype : a communications framework for robotics %U <http://eprints.qut.edu.au/57790>', in D. Carnegie & W. Browne (eds), *Proceedings of the 2012 Australasian Conference on Robotics and Automation*, Australian Robotics & Automation Association, Wellington, New Zealand.

Craig, D.A., Nguyen, H.T. & Burchey, H.A. 2006, 'Two Channel EEG Thought Pattern Classifier', *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pp. 1291-4.

Denojean-Mairet, M., Tan, Q., Pivot, F. & Ally, M. 2014a, 'A Ubiquitous Computing Platform - Affordable Telepresence Robot Design and Applications', *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*, pp. 793-8.

Denojean-Mairet, M., Tan, Q., Pivot, F. & Ally, M. 2014b, 'A Ubiquitous Computing Platform - Affordable Telepresence Robot Design and Applications', 2014 IEEE 17th International Conference on Computational Science and Engineering, pp. 793-8.

Dongmahn, S., Suhyun, K., JaeWook, Y., Hogun, P. & Heedong, K. 2012, 'Immersive Panorama TV service system,' IEEE International Conference on Consumer Electronics (ICCE), , pp. 201-2.

ETSI, E.T.S.I. 2006, Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Review of available material on QoS requirements of Multimedia Services.

Fowdur, T.P. & Soyjaudah, K.M.S. 2011, 'Wireless JPEG Image Transmission Using Multiple Diversity Combining and Unequal Error Protection', Wireless Personal Communications, vol. 59, no. 2, pp. 275-96.

Freeman, R.L. 2005, Fundamentals of Telecommunications, 2nd Edition, Wiley-IEEE Press.

Gao, Y., Chen, H., Li, Y. & Liu, Y. 2016, 'Autonomous WiFi-relay control with mobile robots,' 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), pp. 198-203.

Ghalut, T. & Larijani, H. 2014, 'Non-intrusive method for video quality prediction over LTE using random neural networks (RNN),' 2014 9th International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP), pp. 519-24.

Gonzalez-Jimenez, J., Galindo, C. & Ruiz-Sarmiento, J.R. 2012, 'Technical improvements of the Giraff telepresence robot based on users' evaluation,' IEEE Conference on RO-MAN, pp. 827-32.

Gonzalez, J.D.T., Linton, G., Wai-keung, F. & Barwell, R. 2012, 'ON internet based supermedia enhanced telepresence via cellular data network,' IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 2750-5.

Grangetto, M., Magli, E. & Olmo, G. 2003, 'Error sensitivity data structures and retransmission strategies for robust JPEG 2000 wireless imaging', IEEE Transactions on Consumer Electronics, vol. 49, no. 4, pp. 872-82.

Gurler, C.G., Gorkemli, B., Saygili, G. & Tekalp, A.M. 2011, 'Flexible Transport of 3-D Video Over Networks', Proceedings of the IEEE, vol. 99, no. 4, pp. 694-707.

H. Demuth, M.B. 2000, 'Matlab Neural Network Toolbox User's Guide.'

Ha, V.K.L., Chai, R. & Nguyen, H.T. 2017, 'Real-Time WebRTC-Based Design for a Telepresence Wheelchair,' Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE, Jeju, Korea, 2017, pp. 2676-2679.

Ha, V.K.L., Nguyen, T.N. & Nguyen, H.T. 2015, 'Real-time transmission of panoramic images for a telepresence wheelchair,' Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, pp. 3565-8.

Ha, V.K.L., Nguyen, T.N. & Nguyen, H.T. 2016a, 'Real-time Video Streaming with Multi-Camera for a Telepresence Wheelchair,' The 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016, Phuket, Thailand.

Ha, V.K.L., Nguyen, T.N. & Nguyen, H.T. 2016b, 'A Telepresence Wheelchair Using Cellular Network Infrastructure in Outdoor Environments,' Engineering in Medicine and Biology Society (EMBC), 2016 38th Annual International Conference of the IEEE, pp. 5352-5.

Hasegawa, K. & Nakauchi, Y. 2013, 'Telepresence robot conveying pre-motions for avoiding speech collisions in teleconference,' 2013 IEEE RO-MAN, pp. 350-1.

Herring, S.C. 2013, 'Telepresence Robot for Academics,' paper presented to the ASIST, Montreal, Quebec, Canada, 1-6 November.

Hongo, N., Yamamoto, H. & Yamazaki, K. 2014, 'Web shopping support system for elderly people using WebRTC,' 16th International Conference on Advanced Communication Technology, pp. 934-40.

Horn, O. 2012, 'Smart wheelchairs: Past and current trends', 2012 1st International Conference on Systems and Computer Science (ICSCS), pp. 1-6.

Hsiao, Y.M., Chen, C.H., Lee, J.F. & Chu, Y.s. 2012, 'Designing and implementing a scalable video-streaming system using an adaptive control scheme', IEEE Transactions on Consumer Electronics, vol. 58, no. 4, pp. 1314-22.

Huang, K.C., Chien, P.Y., Chien, C.A., Chang, H.C. & Guo, J.I. 2014, 'A 360-degree panoramic video system design', Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test, pp. 1-4.

IETF 2016, RTCWEB. <https://tools.ietf.org/html/draft-ietf-rtcweb-overview-15>, viewed 13 Feb 2017.

ITU-R 2012, 'Methodology for the subjective assessment of the quality of television pictures,' Jan 2012.

ITU-T-G.1010, I.T.U.R. 2001, Transmission systems and media, digital systems and networks, Switzerland Geneva.

ITU-T 1993, Recommendation T.81 Digital compression and coding of continuous-tone still images - Requirements and guidelines, ITU-T.

Iturrate, I., Antelis, J.M., Kubler, A. & Minguez, J. 2009, 'A Noninvasive Brain-Actuated Wheelchair Based on a P300 Neurophysiological Protocol and Automated Navigation', *Robotics, IEEE Transactions on*, vol. 25, no. 3, pp. 614-27.

Jaju, A., Banerji, A. & Pal, P.K. 2013, 'Development and evaluation of a telepresence interface for teleoperation of a robot manipulator,' *Ubiquitous Robots and Ambient Intelligence (URAI)*, 2013 10th International Conference on, pp. 90-5.

Joveluro, P., Malekmohamadi, H., Fernando, W.A.C. & Kondozi, A.M. 2010, 'Perceptual Video Quality Metric for 3D video quality assessment', *3DTV-Conference: The True Vision - Capture, Transmission, and Display of 3D Video (3DTV-CON)*, 2010, pp. 1-4.

Katsura, S. & Ohnishi, K. 2006, 'Semiautonomous Wheelchair Based on Quarry of Environmental Information,' *IEEE Transactions on Industrial Electronics*, vol. 53, no. 4, pp. 1373-82.

Khoswanto, P.S.a.H. 2014, 'Open protocol framework for telepresence robot,' *ARPJ Journal of Engineering and Applied Sciences* vol. 9, p. 4.

Kilinc, C. & Andersson, K. 2014, 'A Congestion Avoidance Mechanism for WebRTC Interactive Video Sessions in LTE Networks', *Wireless Personal Communications*, vol. 77, no. 4, pp. 2417-43.

Kim, H., Campos, T.d. & Hilton, A. 2016, 'Room Layout Estimation with Object and Material Attributes Information Using a Spherical Camera,' 2016 Fourth International Conference on 3D Vision (3DV), pp. 519-27.

Kim, K.T. & Lee, S.W. 2016, 'Towards an EEG-based intelligent wheelchair driving system with vibro-tactile stimuli', 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 002382-5.

Kodera, S., Fujihashi, T., Saruwatari, S. & Watanabe, T. 2014, 'Multi-view video streaming with mobile cameras,' 2014 IEEE Global Communications Conference, pp. 1412-7.

Kreher, R. & Gaenger, K. 2010, LTE Signaling, Troubleshooting, and Optimization John Wiley & Sons, Ltd.

Kristoffersson, A., Coradeschi, S. & Loutfi, A. 2013, 'A Review of Mobile Robotic Telepresence', *Advances in Human-Computer Interaction*, vol. 2013, p. 17.

Ktistakis, I.P. & Bourbakis, N.G. 2017, 'Assistive Intelligent Robotic Wheelchairs', *IEEE Potentials*, vol. 36, no. 1, pp. 10-3.

Leeb, R., Tonin, L., Rohm, M., Desideri, L., Carlson, T. & Millán, J.d.R. 2015, 'Towards Independence: A BCI Telepresence Robot for People With Severe Motor Disabilities,' *Proceedings of the IEEE*, vol. 103, no. 6, pp. 969-82.

Lepage, P., Létourneau, D., Hamel, M., Brière, S., Corriveau, H., Tousignant, M. & Michaud, F. 2016, 'Telehomecare telecommunication framework; From remote patient monitoring to video visits and robot telepresence', 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 3269-72.

Li, H., Kutbi, M., Li, X., Cai, C., Mordohai, P. & Hua, G. 2016, 'An egocentric computer vision based co-robot wheelchair,' 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1829-36.

Li, Z., Zhao, S., Duan, J., Su, C.Y., Yang, C. & Zhao, X. 2017, 'Human Cooperative Wheelchair With Brain;Machine Interaction Based on Shared Control Strategy', IEEE/ASME Transactions on Mechatronics, vol. 22, no. 1, pp. 185-95.

Lian, Y., Zhang, W. & Jiang, J. 2012, 'The architecture of the remote control system oriented to 4G networks', 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), pp. 1386-90.

Linck, S., Mory, E., Bourgeois, J., Dedu, E. & Spies, F. 2006, 'Video quality estimation of DCCP streaming over wireless networks,' 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06), pp.1-8.

Liu, L., Chen, W. & Wang, J. 2015, 'Experimental Study on mapping and localization algorithm of intelligent wheelchair in spacious and dynamic environments,' 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 871-6.

Loreto, S. & Romano, S.P. 2012, 'Real-Time Communications in the Web: Issues, Achievements, and Ongoing Standardization Efforts,' IEEE Internet Computing, vol. 16, no. 5, pp. 68-73.

Ltd, P.T.P. 2014, Mobile network guide improving mobile signal, Australia, viewed 30 Jan 2016, <<http://www.mobilenetworkguide.com.au/pdf/Mobile-Network-Guide-Improving-Mobile-Signal.pdf>>.

Lyons, K.R. & Joshi, S.S. 2013, 'Paralyzed subject controls telepresence mobile robot using novel sEMG brain-computer interface: Case study,' Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on, pp. 1-6.

MacCormick, J. 2013, 'Video chat with multiple cameras,' paper presented to the Proceedings of the 2013 conference on Computer supported cooperative work companion, San Antonio, Texas, USA.

Martinez-Garcia, E.A., Gallegos, E. & Jaichandar, K.S. 2012, 'Telepresence by deploying an avatar robot with brain-robot interfacing,' Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on, pp. 144-9.

Martini, M.G., Istepanian, R.S.H., Mazzotti, M. & Philip, N.Y. 2010, 'Robust Multilayer Control for Enhanced Wireless Telemedical Video Streaming,' IEEE Transactions on Mobile Computing,, vol. 9, no. 1, pp. 5-16.

Mathur, A.S. 2015, 'Low-cost virtual reality for medical training,' Virtual Reality (VR), 2015 IEEE, pp. 345-6.

McCulloch, W. & Pitts, W. 1943, 'A logical calculus of the ideas immanent in nervous activity', The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115-33.

Mozilla 2017, WebRTC connectivity, viewed 10th June 2017, <https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity>.

Mulla, A., Baviskar, J., Khare, S. & Kazi, F. 2015, 'The Wireless Technologies for Smart Grid Communication: A Review,' 2015 Fifth International Conference on Communication Systems and Network Technologies, pp. 442-7.

Muller, K., Merkle, P. & Wiegand, T. 2011, '3-D Video Representation Using Depth Maps', Proceedings of the IEEE, vol. 99, no. 4, pp. 643-56.

N, M.T., Rosman, R. & Sarmawi, D.S. 2011, 'Design and analysis of wireless controller panel using RF module's for robotic wheelchair,' 2011 IEEE Symposium on Industrial Electronics and Applications, pp. 376-81.

Nguyen, A.V., Nguyen, L.B., Su, S. & Nguyen, H.T. 2012, 'Development of a Bayesian neural network to perform obstacle avoidance for an intelligent wheelchair', Engineering in

Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, pp. 1884-7.

Nguyen, A.V., Su, S. & Nguyen, H.T. 2011, 'Development of a Bayesian recursive algorithm to find free-spaces for an intelligent wheelchair,' Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE, pp. 7250-3.

Nguyen, H.T., King, L.M., Knight, G. & IEEE 2004, 'Real-time head movement system and embedded Linux implementation for the control of power wheelchairs,' Proceedings of the 26th Annual International Conference of the Ieee Engineering in Medicine and Biology Society, Vols 1-7, vol. 26, pp. 4892-5.

Nguyen, J.S., Nguyen, T.H. & Nguyen, H.T. 2009, 'Semi-autonomous wheelchair system using stereoscopic cameras,' Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pp. 5068-71.

Nguyen, J.S., Su, S.W. & Nguyen, H.T. 2010, 'Spherical vision cameras in a semi-autonomous wheelchair system,' Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, pp. 4064-7.

Nguyen, J.S., Su, S.W. & Nguyen, H.T. 2013, 'Experimental study on a smart wheelchair system using a combination of stereoscopic and spherical vision,' 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 4597-600.

Nguyen, J.S., Tuan Nghia, N., Tran, Y., Su, S.W., Craig, A. & Nguyen, H.T. 2012, 'Real-time performance of a hands-free semi-autonomous wheelchair system using a combination of stereoscopic and spherical vision', Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, pp. 3069-72.

Oculus 2016, Oculus Rift, viewed 16 Feb 2016, <<https://www.oculus.com/en-us/rift/>>.

Oh-Hun, K., Seong-Yong, K., Young-Geun, K. & Dong-Soo, K. 2010, 'Telepresence robot system for English tutoring,' IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), pp. 152-5.

Oh, K.J., Yea, S., Vetro, A. & Ho, Y.S. 2009, 'Depth Reconstruction Filter and Down/Up Sampling for Depth Coding in 3-D Video', IEEE Signal Processing Letters, vol. 16, no. 9, pp. 747-50.

Oleksii, I. 2012, 'Using PSNR parameter as a measurement tool in real-time estimation of multimedia information in WiMAX systems,' Modern Problems of Radio Engineering Telecommunications and Computer Science (TCSET), 2012 International Conference on, pp. 283-.

Orebaugh, A., Ramirez, G., Burke, J. & Pesce, L. 2006, Wireshark \& Ethereal Network Protocol Analyzer Toolkit (Jay Beale's Open Source Security), Syngress Publishing.

Pan, J., Appia, V. & Bovik, A.C. 2016, 'Virtual top-view camera calibration for accurate object representation,' 2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), pp. 21-4.

Pavón-Pulido, N., López-Riquelme, J.A., Pinuaga-Cascales, J.J., Ferruz-Melero, J. & Santos, R.M.d. 2015, 'Cybi: A Smart Companion Robot for Elderly People: Improving Teleoperation and Telepresence Skills by Combining Cloud Computing Technologies and Fuzzy Logic', 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, pp. 198-203.

Pesce, M. 2003, Programming Microsoft DirectShow for digital video and television, Microsoft Press, Redmond, Wash.

Peter Corke, K.F.a.E.M. 2012, 'Skype: a communications framework for robotics', paper presented at the Proceedings of Australasian Conference on Robotics and Automation, New Zealand, 3-5 Dec 2012.

Point Grey Research, I. 2014, Ladybug3 viewed 2 June 2014, <http://www.ptgrey.com/products/ladybug3/ladybug3_360_video_camera.asp>.

Pokhrel, J., Wehbi, B., Morais, A., Cavalli, A. & Allilaire, E. 2013, 'Estimation of QoE of video traffic using a fuzzy expert system,' 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC), pp. 224-9.

Prasad, Chandrasekaran, P., Trueb, G., Howes, N., Ramnath, R., Yu, D., Liu, Y., Xiong, L. & Yang, D. 2012, 'Multi-Resolution Multimedia QoE Models for IPTV Applications', International Journal of Digital Multimedia Broadcasting, vol. 2012, pp. 1-13.

Rifai, C., Sai Ho, L., Hunter, G.P., Tran, Y. & Nguyen, H.T. 2013, 'Classification of wheelchair commands using brain computer interface: comparison between able-bodied persons and patients with tetraplegia', Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE, pp. 989-92.

Samsung 2016, Samsung Gear VR viewed 16 Feb 2016, <<http://www.samsung.com/global/galaxy/wearables/gear-vr/>>.

Schneider, D., Schwalbe, E. & Maas, H.G. 2009, 'Validation of geometric models for fisheye lenses', ISPRS Journal of Photogrammetry and Remote Sensing, vol. 64, no. 3, pp. 259-66.

Schultz, R.J., Nakajima, R. & Nomura, J. 1991, 'Telepresence mobile robot for security applications', Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on, pp. 1063-6 vol.2.

Sebastian, J., Jun-Ming, L. & Yen-Liang, H. 2013, 'Robotic concept design for dementia care,' Advanced Robotics and Intelligent Systems (ARIS), 2013 International Conference on, pp. 169-73.

Seo, D. & Jung, I. 2011, 'Network-adaptive autonomic transcoding algorithm for seamless streaming media service of mobile clients', Multimedia Tools and Applications, vol. 51, no. 3, pp. 897-912.

Sharma, S., Rajeev, S.P. & Devearux, P. 2015, 'An immersive collaborative virtual environment of a university campus for performing virtual campus evacuation drills and tours for campus safety,' Collaboration Technologies and Systems (CTS), 2015 International Conference on, pp. 84-9.

Shen, J., Xu, B., Pei, M. & Jia, Y. 2016, 'A low-cost tele-presence wheelchair system', 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2452-7.

Shigang, L. 2006, 'Full-View Spherical Image Camera', 18th International Conference on Pattern Recognition (ICPR'06), vol. 4, pp. 386-90.

Singh, V., Lozano, A.A. & Ott, J. 2013, 'Performance Analysis of Receive-Side Real-Time Congestion Control for WebRTC', 2013 20th International Packet Video Workshop, pp.1-8.

Siwach, G. & Esmailpour, A. 2014, 'LTE Security potential vulnerability and algorithm enhancements', Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on, pp. 1-7.

Skype, M. 2015, viewed 10 Oct 2015, <https://www.skype.com/en/>.

Suitable 2014, Suitable Technologies, Inc., Palo Alto, CA, viewed 22 May 2014, <<https://www.suitabletech.com/beampro/>>.

Tanaka, F., Takahashi, T., Matsuzoe, S., Tazawa, N. & Morita, M. 2013, 'Child-operated telepresence robot: A field trial connecting classrooms between Australia and Japan,' Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 5896-901.

Tanaka, R., Kurabe, K., Kihal, M.E., Ichimi, M. & Tatsuno, K. 2015, 'Improvement on an obstacle avoidance in telepresence robot', 2015 IEEE/SICE International Symposium on System Integration (SII), pp. 634-9.

Tatsuno, K., Kawai, T., Nako, M., Yasuda, Y., Fukuta, T. & Murata, H. 2010a, 'Development of a remote visitor robot system — Attending a remote conference and visiting to an aged care center', Proceedings of SICE Annual Conference 2010, pp. 3616-7.

Tatsuno, K., Kawai, T., Nako, M., Yasuda, Y., Fukuta, T. & Murata, H. 2010b, 'Development of a remote visitor robot system; Attending a remote conference and visiting to an aged care center', SICE Annual Conference 2010, Proceedings of, pp. 3616-7.

Terrazas Gonzalez, J.D., Linton, G., Wai-keung, F. & Barwell, R. 2012, 'ON internet based supermedia enhanced telepresence via cellular data network', IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, pp. 2750-5.

Tsui, K.M., Norton, A., Brooks, D.J., McCann, E., Medvedev, M.S. & Yanco, H.A. 2013, 'Design and development of two generations of semi-autonomous social telepresence robots', 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA), pp. 1-6.

Vajda, T., Márton, L., Szántó, Z. & Pirooska, H. 2014, 'The effect of JPEG compression on network controller designed for teleoperation system', 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 283-7.

WebRTC 2017, Web Real-Time Communications, viewed 25 Jan 2017, <<https://webrtc.org/>>.

WHO, W.H.O. 2016, From walking canes to wrist alarms: WHO's global survey on assistive technologies, viewed 16 Feb 2016, <<http://www.who.int/features/2016/assistive-technologies/en/>>.

Wireshark 2016, 'Network protocol analyzer', viewed 26 Feb, <<https://www.wireshark.org/>>.

Woods, B. & Watson, N. 2003, 'A Short History of Powered Wheelchairs', *Assistive Technology*, vol. 15, no. 2, pp. 164-80.

Wu, B.F., Jen, C.L. & Huang, T.W. 2011, 'Intelligent Radio Based Positioning and Fuzzy Based Navigation for Robotic Wheelchair with Wireless Local Area Networks', 2011 First International Conference on Robot, Vision and Signal Processing, pp. 61-4.

Xiaolin, W., Cheng, S. & Zixiang, X. 2001, 'On packetization of embedded multimedia bitstreams', *IEEE Transactions on Multimedia*, , vol. 3, no. 1, pp. 132-40.

Xu, Y., Deng, J.D., Nowostawski, M. & Purvis, M.K. 2015, 'Optimized routing for video streaming in multi-hop wireless networks using analytical capacity estimation', *Journal of Computer and System Sciences*, vol. 81, no. 1, pp. 145-57.

Yi, Z., Yu-chao, Q. & Yuan, L. 2010, 'An information collection system of network intelligence wheelchair based on ZigBee Wireless Sensor Networks', 2010 2nd International Conference on Industrial and Information Systems, vol. 1, pp. 140-3.

Yuwei, X., Deng, J.D. & Nowostawski, M. 2013, 'Quality of service for video streaming over multi-hop wireless networks: Admission control approach based on analytical capacity

estimation', Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on, pp. 345-50.

Zhang, R., Li, Y., Yan, Y., Zhang, H., Wu, S., Yu, T. & Gu, Z. 2016, 'Control of a Wheelchair in an Indoor Environment Based on a Brain-Computer Interface and Automated Navigation', IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 24, no. 1, pp. 128-39.

Zhang, R., Wang, Q., Li, K., He, S., Qin, S., Feng, Z., Chen, Y., Song, P., Yang, T., Zhang, Y., Yu, Z., Hu, Y., Shao, M. & Li, Y. 2017, 'A BCI-based Environmental Control System for Patients with Severe Spinal Cord Injuries,' IEEE Transactions on Biomedical Engineering, vol. PP, no. 99, pp. 1959-71.

Zinner, T., Hohlfeld, O., Abboud, O. & Hossfeld, T. 2010, 'Impact of frame rate and resolution on objective QoE metrics,' 2010 Second International Workshop on Quality of Multimedia Experience (QoMEX), pp. 29-34.