Real-Time Sign Language Video Communication over Cell Phones

Jaehong Chon

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2011

Program Authorized to Offer Degree: Department of Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Jaehong Chon

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

_____

Eve A. Riskin

_____

Richard E. Ladner

Reading Committee:

_____

Eve A. Riskin

_____

Richard E. Ladner

_____

Hui Liu

Date: _____

Signature_____

Date_____

University of Washington

Abstract

Real-Time Sign Language Video Communication over Cell Phones

Jaehong Chon

Co-Chairs of the Supervisory Committee:
Professor Eve A. Riskin
Department of Electrical Engineering

Professor Richard E. Ladner
Department of Computer Science and Engineering

The goal of the MobileASL (American Sign Language) research project is to enable sign language video communication over the U.S. cellular network. The performances of mobile phones and wireless networks are two major factors in enabling video calling over wireless networks. Two challenges are limited processing power and short battery life. The wireless channel is especially vulnerable when trying to ensure real-time high-quality video because of packet loss which causes errors, jitter, delay and latency.

In this research, I implement the MobileASL system to enable access on the go. This required speeding up the encoder, enhancing video quality, and extending battery life. Second, I propose a new method to provide better video quality at the decoder when packet loss occurs using feedback-based error-tracking intra-block refreshment. Furthermore, I show the benefits of intra-block refreshment by comparing the video quality and packet size.

Increasing intelligibility for sign language video communication requires many different considerations such as image quality, frame rate, video resolution, and network status. First, I investigate how the quality is different depending on video resolution as bit rates are increased. Next, I calculate intelligibility based on temporal resolution (frame rate) and spatial resolution (bit rate). This intelligibility is verified with different video sequences and used to choose the best encoding parameter settings for different bit rates.

Finally, I summarize two field studies conducted in 2010 and 2011 with the MobileASL

system I built on Windows Mobile and Android. Results showed that participants took advantage of the mobility provided by MobileASL, but that use was limited by low battery life.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

3G: third generation cellular standards (UMTS, HSDPA/HSUPA, EV-DO)

4G: fourth generation cellular standards (HSPA+, LTE, Mobile WiMAX)

AMERICAN SIGN LANGUAGE (ASL): the primary sign language of the Deaf in the United States

ANDROID: operating system for mobile devices such as smartphones and tablet computers

BIT RATE: the number of bits that are conveyed or processed per unit of time

FIELD STUDY: evaluating the usability of the system

FRAMES PER SECOND (FPS): unit of measure of the frame rate of a video

FRAME RATE: the rate at which frames in a video are shown, measured in frames per second (fps)

H.264: the latest IEEE standard for video compression

HSPA: High Speed Packet Access (HSDPA and HSUPA)

INTER-FRAME CODING: encoding a frame using information from other frames

INTELLIGIBILITY: the quality of language that is comprehensible

INTRA-FRAME CODING: encoding a frame using information within that frame

KILOBITS PER SECOND (KBPS): unit of measure of bandwidth

LONG TERM EVOLUTION (LTE): standard for wireless communication of high-speed data

MACROBLOCK: a 16×16 square area of pixels

MEAN OPINION SCORE (MOS): numerical indication of the perceived quality from the users' perspective

NETWORK ADDRESS TRANSLATION (NAT): process of modifying IP address information in IP packet headers

MOTION VECTOR: a vector applied to a macroblock indicating the portion of the reference frame it corresponds to

PEAK SIGNAL TO NOISE RATIO (PSNR): a measure of the quality of an image

QUANTIZATION PARAMETER (QP): quantizer step size, a way to control macroblock quality

REAL-TIME: a processing speed fast enough so that there is no delay in the video

REGION OF INTEREST (ROI): an area of the frame that is specially encoded

SINGLE INSTRUCTION MULTIPLE DATA (SIMD): same operation on multiple data simultaneously

SMARTPHONE: high-end mobile phone that combines the functions of a personal digital assistant and a mobile phone

TRANSMISSION CONTROL PROTOCOL (TCP): reliable, ordered delivery of a stream of bytes from one to another

USER DATAGRAM PROTOCOL (UDP): data delivery without transmission channel

VIDEO COMPRESSION: reducing the amount of data in digital video

VIDEO RESOLUTION: a pixel count in digital frame in video

X264: an open source implementation of H.264

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors, Eve Riskin and Richard Ladner. I appreciate all their contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. Eve is an excellent mentor who continuously helped me to see life and research in their full path. Richard has been very approachable, and his suggestions and feedback has enlightened me through his wide knowledge of signal processing. Their advice on dealing with many research problems, finding a job, and managing family made my Ph.D. possible. I am also grateful to my other committee members, Jake Wobbrock, Hui Liu, and Beth Kolko for their constructive comments.

My work would not have been possible without the collaborations from my colleagues Rahul Vanam, Neva Cherniavsky, Sam Whittle, Anna Cavendar, Joy Kim, Jessica Tran, and Katie O'Leary. I thank Rahul Vanam for the research collaboration in H.264 encoding parameter settings and his friendship. I thank Neva Cherniavsky and Anna Cavendar for the wonderful experience of user study for region-of-interest video encoder and variable frame rate. I thank Sam Whittle for his invaluable help on feedback-based intra refresh when packet loss occurs. I thank Joy Kim for her help in both MobileASL software development in the server and the client and in proof reading this document. I thank Jessica Tran and Katie O'Leary for our successful collaboration on the MobileASL field studies. I also thank the rest of the MobileASL group at the University of Washington, including both graduate and undergraduate students.

Finally, I am very grateful to my parents and two daughters, Heewon Chon, Chaewon Chon, and especially to my wife, Seunghee Park. My wife supported me to go back to the university to pursue graduate studies in my life after nine years at Samsung. Without her love, support, and encouragement, I would not have been able to complete my graduate studies.

DEDICATION

to my dear wife and two daughters, Seunghee, Heewon, and Chaewon

Chapter 1

INTRODUCTION

Since the late 1990s, there has been a tremendous growth in mobile video applications. Being able to stream TV over mobile networks or share and stream live video through cell phones are two good examples of the impact this growth has had. The Austrian wireless company '3' claimed that streamed entertainment content constituted 45-55% of their mobile network traffic in the 2nd quarter of 2009 [36]. According to Cisco, mobile data traffic will grow at a compound annual growth rate of 108 percent between 2009 and 2014, reaching 3.6 exabytes (10^{18} bytes) per month by 2014. Almost 66 percent of the world's mobile data traffic will be video by 2014 [30] (see Figure 1.1). It is obvious that video is becoming one of the most important mobile applications.

One of the most promising applications of mobile video is video chat. Applications of video chat include distance eduction (classes could be taken remotely, and research collaboration between institutes could be made easier); telemedicine and telenursing for diagnosis and consulting; telecommuting to work; giving remote testimonies in court; and general video communication through chat programs such as Skype and Microsoft Live Messenger. The applications described above save time and money, and provide flexibility. There are several solutions available for these services over broadband networks.

There is another important application for video chat: sign language communication. Video communication is especially useful for ASL (American Sign Language) speakers, who currently use video chat programs for computers and video phone setups in the phone to remotely communicate in sign language. VRS (Video Relay Services) in the U.S. provide communication over video phones at home or office with hearing people in real-time through a sign language interpreter. There are efforts to provide the mobility from VRS providers such as Sorenson and ZVRS. Sorenson has SIPRelay which is a service for deaf and hard-of-hearing individuals to place and receive text-based relay calls using mobile devices [11],

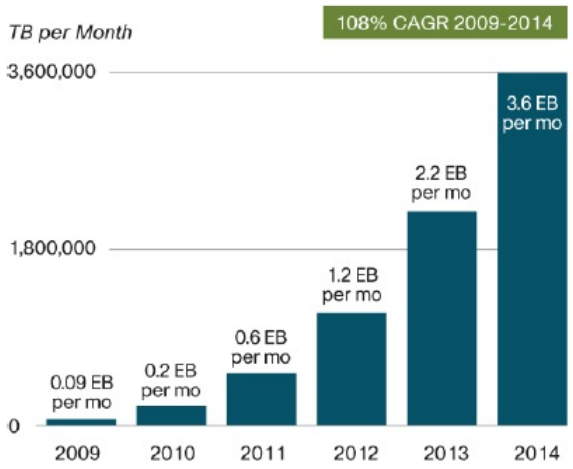


Figure 1.1: Cisco forecasts 3.6 Exabytes per Month of Mobile Data Traffic by 2014 [30].

however, it is not a video-based relay service on mobile devices. Recently ZVRS released mobile VRS software for the latest mobile devices such as the iPhone and Android phones.

The goal of the MobileASL research project at the University of Washington [50] and Cornell University [48] is to enable real-time sign language video communication over the U.S. cellular network; my goal in this research is to build an entire system for MobileASL over current cellular networks, including methods to protect against data loss over cellular networks. MobileASL is the first experimental mobile phone for ASL users in the U.S and is not available to the public yet. For members of the Deaf community, the use of such a phone will potentially greatly affect how they communicate with each other, Compared to texting and e-mail. There is clearly a need for video calls on mobile devices.

In order to allow mobile video communication, a phone must have a video camera on the same side as the screen. Older cell phones do not have a camera set up this way, but many latest smartphones do. The Royal Bank of Canada estimated that smartphone sales will surpass worldwide PC sales by the end of 2011, and shipments of both will approach 400 million a year. Figure 1.2 illustrates this growth of smartphone sales; more smartphones are expected to be sold in 2011 than PCs for the first time [57]. The time is ripe to add value to smartphones, especially for ASL speakers, by utilizing them for video chat.

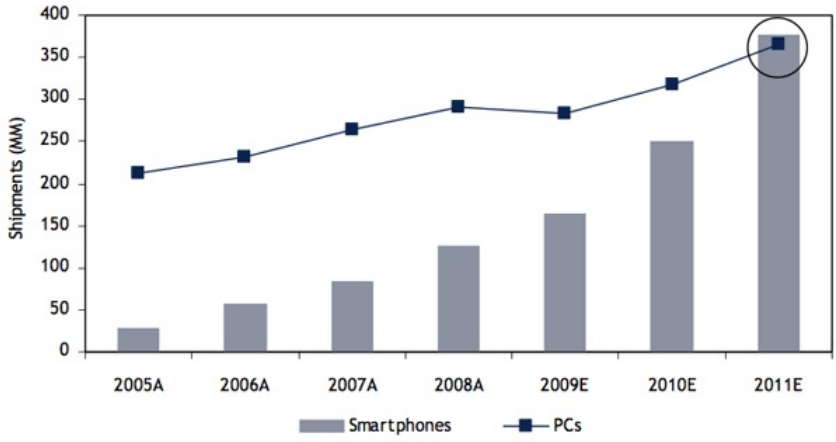


Figure 1.2: Estimate of Smartphone and PC Sales. In 2011, more smartphones will be sold than PCs [57].

1.1 Smartphone Market Trend

After the first iPhone was released in 2007, the mobile phone market changed tremendously. Before the iPhone, some companies such as Microsoft (Windows Mobile), Nokia (Symbian), and RIM (Blackberry) were developing operating systems for smartphones. However, these operating systems were neither convenient nor intuitive to use because they used small text and non-touch screen interfaces (see Figure 1.3).

Android, a smartphone OS developed by Google, was released after the iPhone in late 2007, and mainly focused on providing a comfortable user interface and easy development for application authors. Figure 1.4 shows the first iPhone and the first Android device released in 2007. Unlike previous smart phones, the iPhone and Android phones allowed users to experience a wider range of increasingly effective and entertaining applications than just checking e-mail.

Android was listed as the best-selling smartphone platform worldwide in Q4 2010 by Canalys [18]. Figure 1.5 shows the latest smartphone market share numbers: Apple is flat, while Google is going strong [20]. The total overall market share for Apple iOS and Google Android alone is 60.2%. Therefore, I decided to examine the possibility of providing sign language video communication for these popular platforms.

Figure 1.3: Main Screen of Windows Mobile 6 and Blackberry.

Figure 1.4: The first iPhone and Android released in 2007.

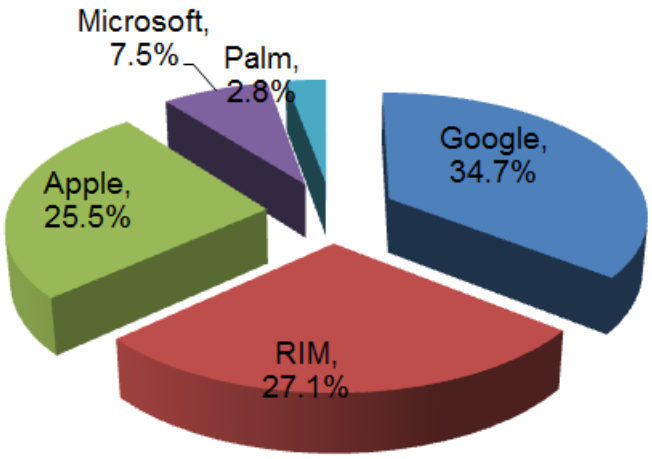


Figure 1.5: Top Smartphone Platforms in March 2011. Total U.S. Smartphone Subscribers Ages 13+.

1.2 Challenges

Many PC-based video chat applications like Skype and Microsoft Live Messenger provide both high video quality and frame rate. However, achieving the same quality on a mobile phone is challenging because of slower processor speeds, which results in lower frame rate and limited battery life, even on the latest smartphones, and video degradation over wireless networks due to block errors, jitter, delay, and latency. I propose enhancing mobile video performance by speeding up the encoder through assembly optimization, H.264 parameter optimization, and other optimizations.

Unlike video streaming services, which utilize buffering to prevent lower video quality, mobile video calls run in real-time and so the video quality must recover quickly when packet loss occurs. I address packet loss, which is the main cause for lower video quality over wireless networks [23], using feedback-based recovery.

1.3 Contributions

MobileASL is the experimental mobile phone for ASL users to enable real-time sign language video communication over the U.S. cellular network. The performances of mobile phones

(limited computational power) and wireless networks (packet loss) are the major challenges to providing two-way video communication on cellular phones. Video compression is very computationally expensive. The overall encoding time widely varies depending on encoding parameters and video resolution.

Chapter 3 describes the original implementation of MobileASL and shows how to achieve for real-time sign language video communication on the Windows Mobile 6 platform for the HTC TyTN-II hand set as follows:

- Speed up key components in H.264 with ARMv6 SIMD instruction set
- Design network protocol for peer-to-peer communication behind local router
- Provide the contact list and logging with MobileASL server.

Another factor that affects compressed video quality is packet loss because of error propagation due to inter-frame prediction. In Chapter 4, I present an effective algorithm to overcome the effects of packet loss and maintain intelligibility for sign language communication on mobile devices using feedback-based intra block refresh encoding. The contributions are as follows:

- Reduce the packet size and recover the quality quickly utilizing skip blocks in a predicted frame when packet loss occurs
- Analyze intra block refresh with 3G cellular network simulator.

Intelligibility of sign language video communication is affected by many things such as image quality, the frame rate, the video resolution, the packet loss ratio, and delay. In Chapter 5, I discuss how video resolution affects the quality depending on the bit rates. Next, I present a model to evaluate the relationship between frame rates and bit rates that evaluate the intelligibility for different video sequences. Finally, I show the bandwidth in the 3G cellular network is time varying. The details are as follows:

- Analyze effect of spatial resolution on frame rate and quality

- Propose an intelligibility model that accounts for frame rates and bit rates
- Measure the available bandwidth over the cellular networks on the mobile device.

Chapter 6 describes a new implementation of MobileASL on Android devices as follows:

- Redesign the MobileASL architecture to be compatible with existing systems
- Design the software architecture based on a baseline platform
- Provide the contact list and logging features for a field study.

Chapter 7 describes two field studies with the Windows Mobile 6 and Android versions of MobileASL and provides results and feedback from participants as follows:

- Explain the results for two field studies in 2010 and 2011
- Analyze what we get from the field studies and suggest directions for the next field study

Finally in Chapter 8, I conclude and provide future directions for research.

Chapter 2

BACKGROUND

In this chapter, I describe two key components for sign language video communications on the mobile phones: video compression and wireless communications.

2.1 Video Compression

Video compression is essential to reduce the amount of data for DVD-video, digital television, video conferencing and video streaming. H.264 is currently the best video compression yet standard [43], because it provides better video quality for fewer bits through features such as adaptive block size, intra coding with spatial prediction, and in-loop deblocking.

An H.264 video encoder consists of motion estimation/compensation, transform, and entropy encoding processes to produce a compressed bitstream (see Figure 2.1). Each of these parts are explained in detail below.

2.1.1 Motion Estimation and Compensation

One of the key parts of H.264 is motion estimation, which determines motion vectors that predict 16×16 macroblocks either with intra prediction or inter prediction. Intra prediction utilizes spatial correlation from the current frame, while inter prediction utilizes temporal correlation from adjacent frames in a video sequence.

Two types of intra-frame prediction (Intra-4x4 and Intra-16x16) are supported for either regions with significant detail or very smooth regions. In Intra-4x4 mode, each 4x4 block is predicted using prior decoded samples from adjacent neighboring samples with one of nine prediction modes (see Figure 2.2). For Intra-16x16, only four of the prediction modes from Intra-4x4 are supported (vertical, two horizontal, and DC).

Various coding types are specified for inter-frame prediction. For motion compensated prediction, a macroblock can be divided into partitions of 16x16, 16x8, 8x16, and 8x8

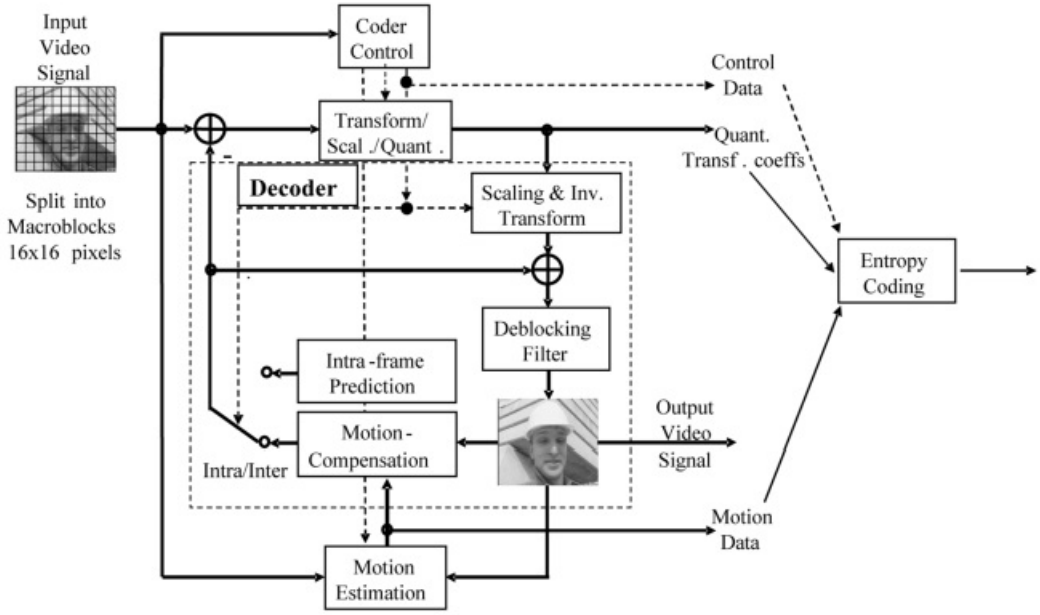


Figure 2.1: Basic coding structure for H.264 for a macroblock [63]. It is a hybrid of inter-frame prediction to exploit temporal dependencies using motion estimation and intra-frame prediction to exploit spatial dependencies. The region inside dotted line includes components that are common for both the H.264 encoder and decoder.

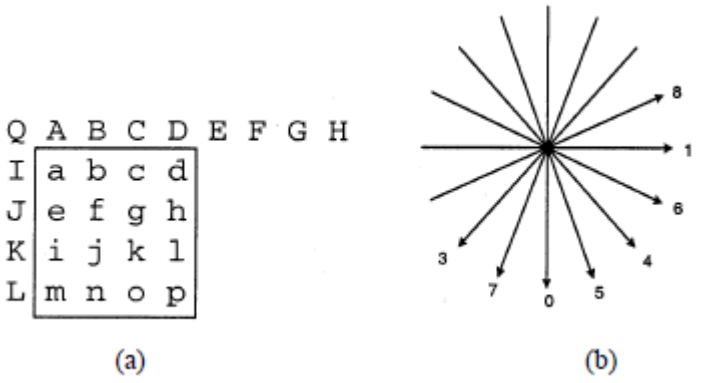


Figure 2.2: (a) Intra luma 4x4 prediction on samples a-p using samples A-Q that are already encoded and (b) the eight prediction direction of Intra luma 4x4 [63]

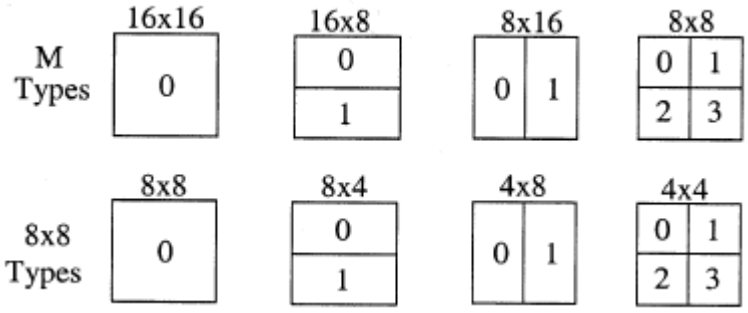


Figure 2.3: Different macroblock sizes for inter-frame motion estimation [63]. Top: segmentation of 16x16 macroblock; bottom: segmentation of 8x8 partitions.

samples. In the case of 8x8 samples, samples are further partitioned into partitions of 8x4, 4x8, or 4x4 samples if needed (see Figure 2.3).

There are also multiple choices for the method of full-pixel motion estimation for inter-frame prediction, which include diamond search, hexagon search, UMH (Uneven Multi-Hex), and exhaustive search. Diamond search simply checks the motion vectors one pixel up, left, down, and right to pick up the least difference. Hexagon search is similar to diamond search, except that it searches the 6 points of a surrounding hexagon. Hexagonal search searches more points than diamond search but it is almost as fast. UMH is considerably slower than hexagonal search because it searches a complex multi-hexagon pattern. Exhaustive search is mathematically equivalent to using a bruteforce method of searching every single motion vector. Therefore, choosing a method for full-pixel motion estimation is a matter of choosing either better estimation or faster estimation. This problem is especially important for mobile phones because of low computation power.

Motion compensation is used to restore the original macroblock. This process is used in both the video encoder and decoder so that they both use the same reference frame.

2.1.2 Transform

The resulting prediction residual from either inter prediction or intra prediction is transformed by a Discrete Cosine Transform (DCT), which compacts the residual into as few

coefficients as possible. H.264 is based on a 4x4 transform. In H.264, each 4x4 block is transformed using an integer transform. Finally, the transformed coefficients are quantized and encoded using entropy coding.

2.1.3 Other Key Features

H.264 also contains many features that can increase the compressed video's robustness to data loss. These features include slice-structured coding, Flexible Macroblock Ordering (FMO), Arbitrary Slice Ordering (ASO), and data partitioning [63]. In H.264, a group of macroblocks may be grouped into a slice, which become independently decodable. Slice sizes are highly flexible. ASO allows the encoder to enable sending and receiving the slices in any order. FMO partitions the picture into regions called slice groups, where the first slice group and second slice group are transmitted in separate packets. If one of them is lost, the remaining slice group can be used to restore the lost slice group. Therefore, these features can improve robustness to data losses.

Unfortunately, these features are not well-suited to my goal of real-time, low-complexity encoding and decoding because in my system, an encoded frame is carried on a single packet over the network. In my research, I have developed fast algorithms that ensure good quality throughout the cell phone video call even when packet loss occurs.

2.2 Wireless Communications

U.S. mobile phone service providers such as Verizon, AT&T, Sprint, and T-Mobile operate wireless networks using many different wireless communication standards. Depending on their origin, third generation (3G) mobile systems have evolved into two major standards: AT&T and T-Mobile using 3GPP based on UMTS, and Verizon and Sprint using 3GPP2 based on CDMA.

In order to increase the capacity and speed of wireless data networks, a lot of effort has been put towards developing fourth generation (4G) mobile systems, which has resulted in a number of new technologies including 3GPP Long Term Evolution (LTE), Mobile WiMAX (IEEE 802.16e), Ultra Mobile Broadband (UMB), Flash-OFDM, and Mobile Broadband Wireless Access (MBWA) (IEEE 802.20) [13].

Table 2.1: Current 4G technologies in the United States [13].

Service Provider	Technology	Deployment	Peak Data Rate (Downlink and Uplink)
Verizon	LTE	December 2010	12 Mbps and 5 Mbps
AT&T	HSPA+	June 2010	21 Mbps and 5.8 Mbps
	LTE	mid-2011 to 2013	
T-Mobile	HSPA+	July 2010	21 Mbps and 5.8 Mbps
Sprint	WiMAX	2008 - 2010	11.3 Mbps and 3.0 Mbps
	LTE	near future	

2.2.1 3G Mobile Systems

The most widely used 3G technology is the Global System for Mobile Communications, or GSM. On top of GSM is a data network called General Packet Radio Service (GPRS) as well as an upgrade for faster speeds called Enhanced Data rates for GSM Evolution (EDGE), which can carry data speeds from 35.2 kbps up to 236.8 kbps. Furthermore, there is a high-speed network, based on UMTS and High-Speed Downlink Packet Access (HSDPA), more commonly known as 3G.

This 3G system is still widely used in the United States. Since a video call is bi-directional, the effective bandwidth is limited by the uplink. Hence, this research, with its target bit rate of 30 kbps, is well suited to mobile sign language communication over current U.S. cellular data networks.

2.2.2 4G Mobile Systems

U.S. mobile service providers have developed many competing 4G technologies, so it is still too early to fully take advantage of 4G mobile systems. Table 2.1 summarizes the deployment dates and peak data rates for each of these technologies in the United States.

Figure 2.4 shows the downlink and uplink speeds over Sprint WiMAX 4G and T-Mobile HSPA+ as measured by INTOMOBILE [19]. This measurement indicates that the operating data rates in 4G networks are far from the peak data rates so 4G might not well-suited for

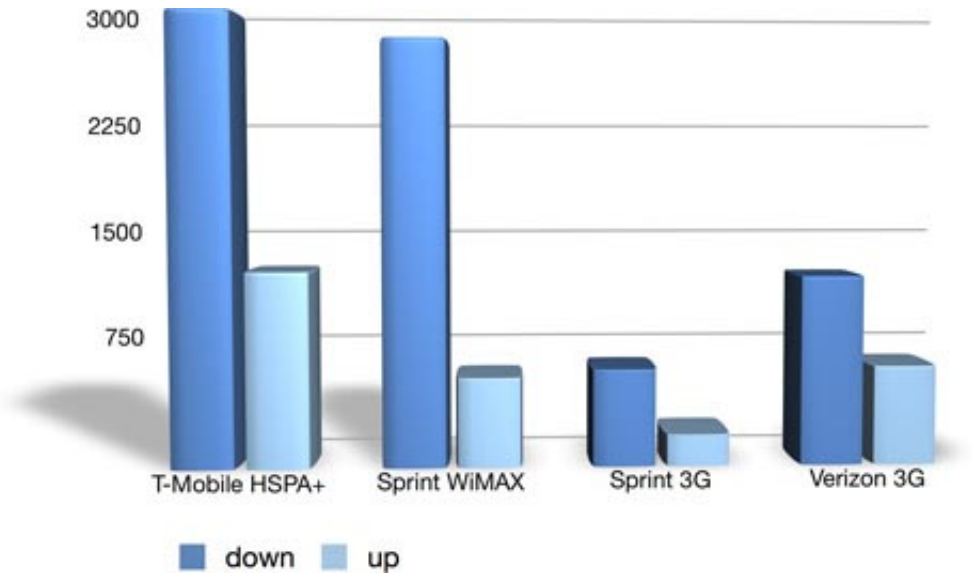


Figure 2.4: Data Speed Showdown: Sprint WiMAX 4G vs T-Mobile HSPA+ [19].

real-time video communication yet.

4G network coverage must also be considered. Currently, 4G networks are not nearly as widely available as 3G networks. Therefore, I mainly focus on current 3G cellular networks in this research.

2.2.3 Data Network over Mobile Systems

Mobile service providers AT&T and T-Mobile upgraded their core wireless network in 2004 to an all-IP platform. This core network turned GSM and UMTS voice networks into a flat IP platform using standard IP interfaces [44]. Therefore, the Internet protocol suite (commonly TCP/IP), the set of communications protocols used for the Internet, can be used over mobile networks as well. I took advantage of this and investigated how MobileASL could use a general TCP/IP networking system over both the Wi-Fi and a current cellular data network in the U.S. (AT&T).

There are two core protocols in the transport layer of the Internet protocol suite: TCP and UDP. UDP is a message-based connectionless protocol that provides a non-guaranteed datagram delivery. Since video communication is real-time and thus does not need guar-

anteed delivery of data that previously failed to arrive, MobileASL uses UDP to transmit encoded video frames. More detailed information about MobileASL's protocol design is described in the next chapter.

Chapter 3

BUILDING THE MOBILEASL SYSTEM

In this chapter, I describe original implementation of our MobileASL software, which runs on the Windows Mobile 6 operating system using HTC TyTN-II hand set. MobileASL is compatible with the H.264/AVC compression standard and can be decoded by any H.264 decoder. The x264 open source H.264 encoder was selected because of its fast speed [15, 47, 16], making it a good choice for compression on low-power devices such as mobile phones. In [47], x264 showed better quality than several commercial H.264/AVC encoders. When compared with the JM reference encoder (version 10.2) [3], x264 (version 0.47.534) was shown to be 50 times faster, while providing bit rates within 5%, for the same PSNR.

We selected parameters for mobile sign language communication over current U.S. cellular data networks to be 15 fps, 30 kbps, and 250ms delay. It is really difficult to achieve these targets together on the mobile phone. It requires IP enabled phones; a name server to keep track of the mapping of phone to IP addresses; a front facing camera; adequate video compression (10 fps or more, decent fidelity); adequate network (low delay, adequate bandwidth), and a usable user interface (UI). I created a working mobile video phone system from scratch under these constraints.

3.1 Prior Work

There are many solutions to provide video calls on mobile devices. iVisit is an application both for Windows Mobile and desktops or laptops [6]. vzomobile is a simple and high quality video calling application for Windows Mobile based smartphones [14].

Since I'm interested in building the MobileASL system for Windows Mobile, I tested iVisit and vzomobile on the HTC TyTN-II. The test placed an iPod's stopwatch in front of camera of one handset (the transmitter). By looking for skipped milliseconds on the other handset's screen (the receiver), it should be possible to estimate the transmission frame rate.

Application	Network	Frames Per Second	One-way Delay
iVisit	Wi-Fi	7-9	about 1 second
	Cellular	4-5	
vzomobile	Wi-Fi	3-4	about 5 seconds
	Cellular	3-4	

Table 3.1: Frames Per Second for iVisit and vzomobile on both Wi-Fi and 3G.

The receiver's screen was observed for at least 20 seconds. At a frame rate at or in excess of 10 fps the counter displayed on the receiver's screen should be indistinguishable from looking directly at the iPod's screen. At fewer than 10 frames per second numbers should become visible, for example at 4 fps the observer may see 2,6,8,0 clearly. The experience was conducted on Wi-Fi and 3G cellular connections where the location has a maximum signal strength for both Wi-Fi and 3G. The result is as follows (see Table 3.1).

There was a lot of frame skipping noticeable on both vzomobile and iVisit displays. There was also one-way trip time (the time it takes to send the frame and receive it on the other device) of around 1 second for iVisit and much longer, about 5 seconds, for vzomobile.

Because of the similarities between the frame rates over the Wi-Fi and Cellular networks, it is obvious that the limiting factor for both iVisit and vzomobile will be the processing power of the handset, not the data transfer rate.

3.2 MobileASL System

MobileASL is a peer-to-peer networking application that works both on a Wi-Fi network and the current cellular data network in the U.S. Since both networks are working on standard IP interfaces, I had to build a method to find and connect with other phones under IPv4 address exhaustion due to the dramatic growth of the Internet. This address depletion requires a NAT (Network Address Translator) [58], which translates one IP address space into another at a router. It enables multiple hosts on a private network to access the Internet using a single public IP address. I used a NAT traversal [26] technique to establish a direct

connection between clients when both are behind NATs.

As a hardware platform, I used the HTC TyTN-II mobile phone which has a Qualcomm MSM7200 (400MHz ARM1136EJ-S processor). We chose this phone because it has a front camera on the same side as the screen and runs Windows Mobile 6.1. Its processor adopts the ARMv6 core having a new SIMD instruction set [1]. This phone has a 320×240 pixel screen, a VGA video front camera, a QWERTY keyboard, a H.264 hardware decoder, etc. Also, the HTC TyTN-II provides wireless capabilities such as Wi-Fi 802.11b/g, 2G (GPRS and EDGE) and 3G (HSDPA). Figure 3.1 shows a MobileASL screenshot.

Figure 3.1: A screenshot of our MobileASL codec on the HTC TyTN-II. Two former UW graduate students are talking each other.

I next describe the protocol used for establishing a video conference session on mobile phones.

3.3 NAT-enabled MobileASL Protocol

For peer-to-peer communications behind NATs, the server may be used to deliver the packets to each other. Each client sends the packet to the server and the server will retransmit to the other. But it causes additional delay and increases the load to the server. Therefore, it will not be appropriate.

In order to make a direct connection between clients, the NAT traversal techniques are widely used to alleviate the exhaustion of the IPv4 address space at home and internal networks behind routers with a single public IP address. There are many techniques, but no single method works in every situation since NAT behavior is not standardized [26].

I used one of the simplest but most robust and practical NAT traversal techniques, commonly known as "hole punching" [26]. TCP and UDP hole punching are explained more thoroughly in [26]. The authors evaluated the robustness of these techniques on a variety of existing NATs, and showed that about 82% of the NATs tested support hole punching for UDP, and about 64% support hole punching for TCP streams. I chose UDP hole punching for use in the MobileASL protocol because of its robustness.

UDP (User Datagram Protocol) hole punching is efficient for MobileASL networking, and enables two clients to set up a direct peer-to-peer UDP session with the help of a MobileASL server, even if the clients are both behind NATs. Figure 3.2 shows the MobileASL architecture, which is assisted by the server. This server also acts a name server that keeps a client's name and phone number, its IP address, and online status. This information is used to make a direct connection.

Suppose clients A and B have private IP addresses behind different NATs, as shown in Figure 3.3. Both A and B keep individual UDP sessions on the server by sending periodic keep-alive packets because most NATs simply close the hole if the network is idle for some time period. There is unfortunately no standard timeout value: some NATs timeout as short as 20 seconds [26]. I explain the hole punching process with an example illustrated in Figure 3.3. Let me consider client A and B to have NAT sessions NAT A (Comcast) and NAT B (AT&T) with the server.

As an example, NAT A (Comcast) assigns port 62325 at its own public IP address, 96.225.223.175, for the use of A's session with the server. NAT B (AT&T) assigns port 8200 at its IP address, 32.157.152.185, for B's session with the server. Hole punching proceeds as follows:

1. Each client sends keep-alive messages periodically to maintain their respective UDP sessions with the server.

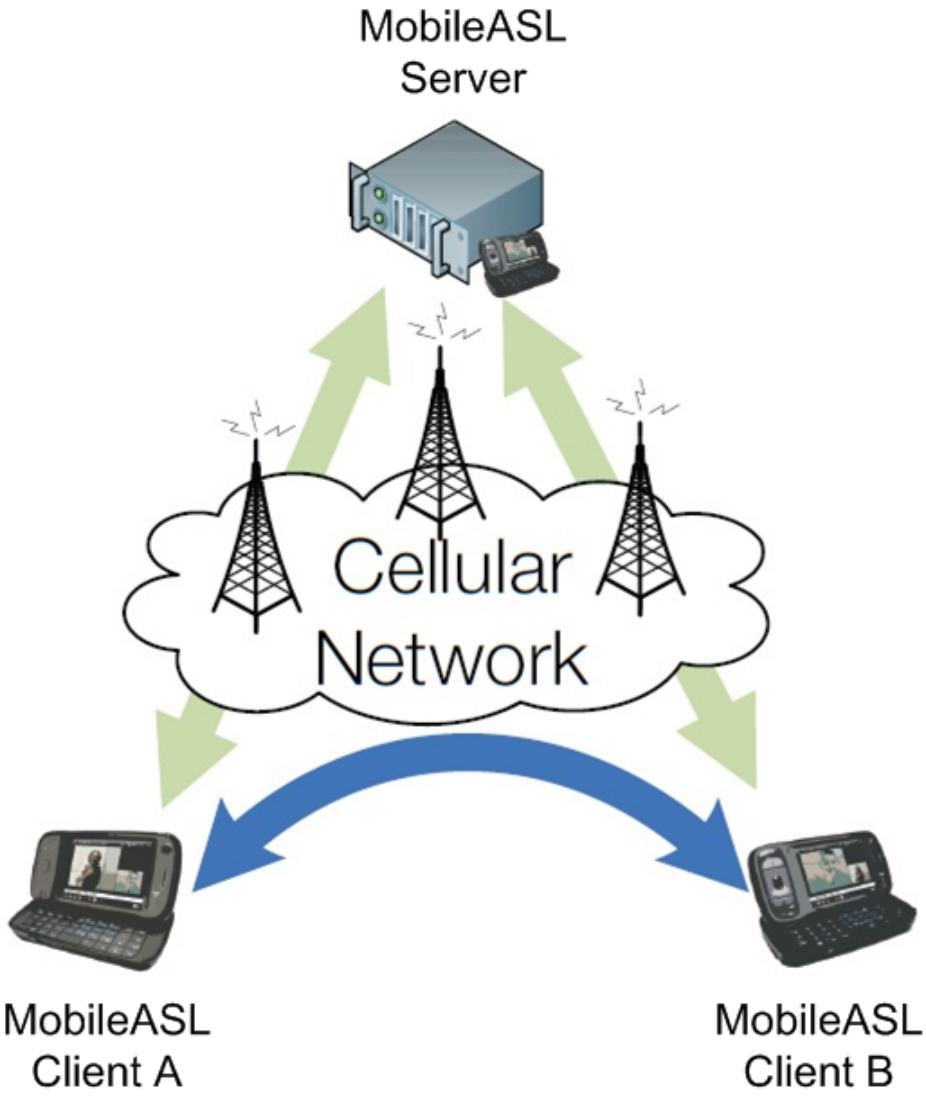


Figure 3.2: Network Architecture for MobileASL.

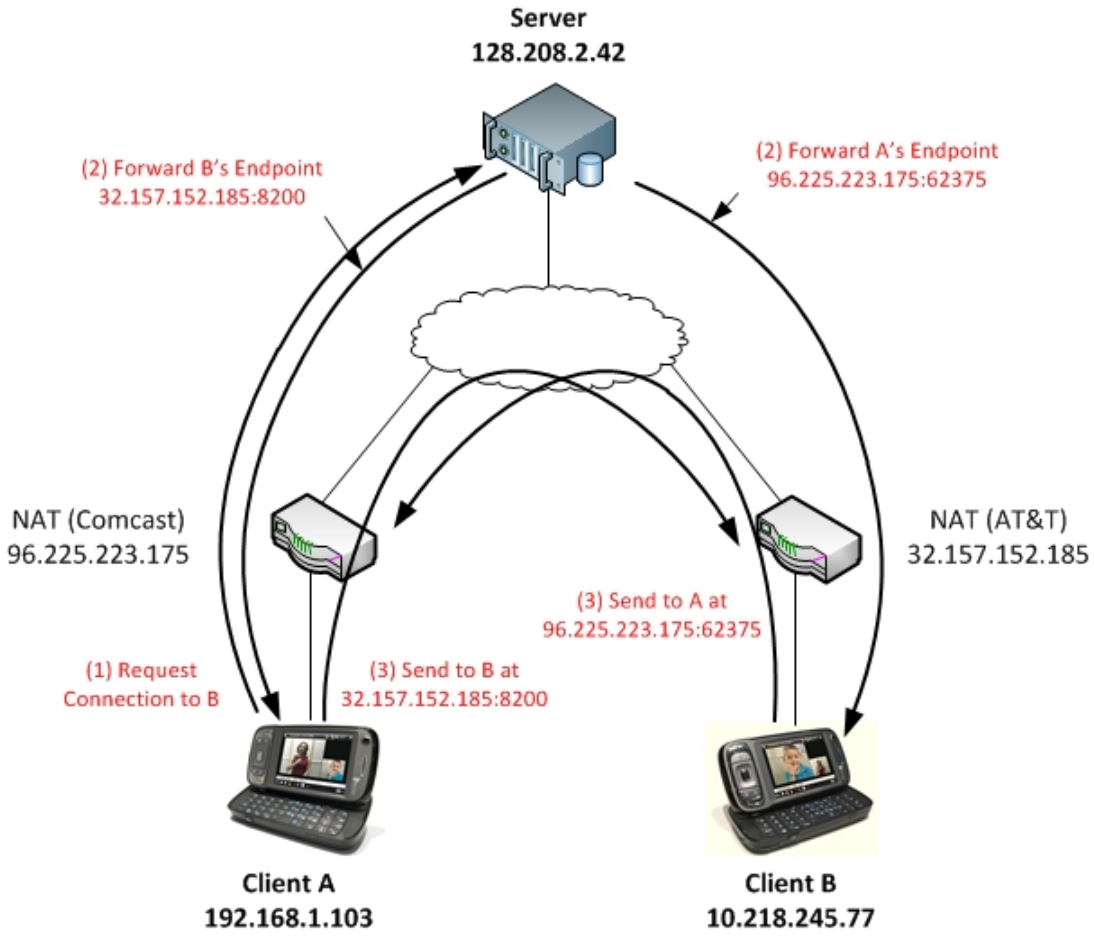


Figure 3.3: NAT-enabled MobileASL Protocol with UDP hole punching.

2. A requests B's information from the server in order to establish a UDP session with B.

3. The server replies to A with B's public endpoint. At the same time, the server forwards A's public endpoint through B's existing UDP session to allow B to establish a connection to A.

4. Both A and B start periodically sending UDP packets containing the calling information (such as name and phone number) to each others' NAT devices directly with public endpoint to punch the hole until A receives a response from B. The NAT devices use the exactly same translation which was previously created for the server so the packet is sent to the client.

Figure 3.4 describes the call flow for making a connection.

3.4 Speeding Up the Encoder

As explained earlier, the typical frame rate on mobile device is around 3-4 fps without speed optimization (see Table 3.1). Frame rates as low as 6 frames per second can be intelligible for signing, though they require the user to sign very slowly. A more comfortable frame rate would be 12 frames/second [37], and higher frame rates are needed for fingerspelling [34, 35, 25]. To reach the desired frame rate of 12-15 frames/second, I optimized the motion estimation, mode decision, transforms, quantization and motion compensation in the H.264 encoder using the Single Instruction Multiple Data (SIMD) instruction set. Then, I determined the H.264 parameter settings and video resolution needed to achieve my target frame rate.

3.4.1 Assembly Optimization

The ARM1136J-S processor used in the HTC TyTN-II is built around the ARM11 core in an integer unit that implements the ARM architecture v6. It supports a range of SIMD DSP instructions that operate on pairs of 16-bit values held in a single register, or on quadruplets of 8-bit values held in a single register [1]. The main operations include addition,

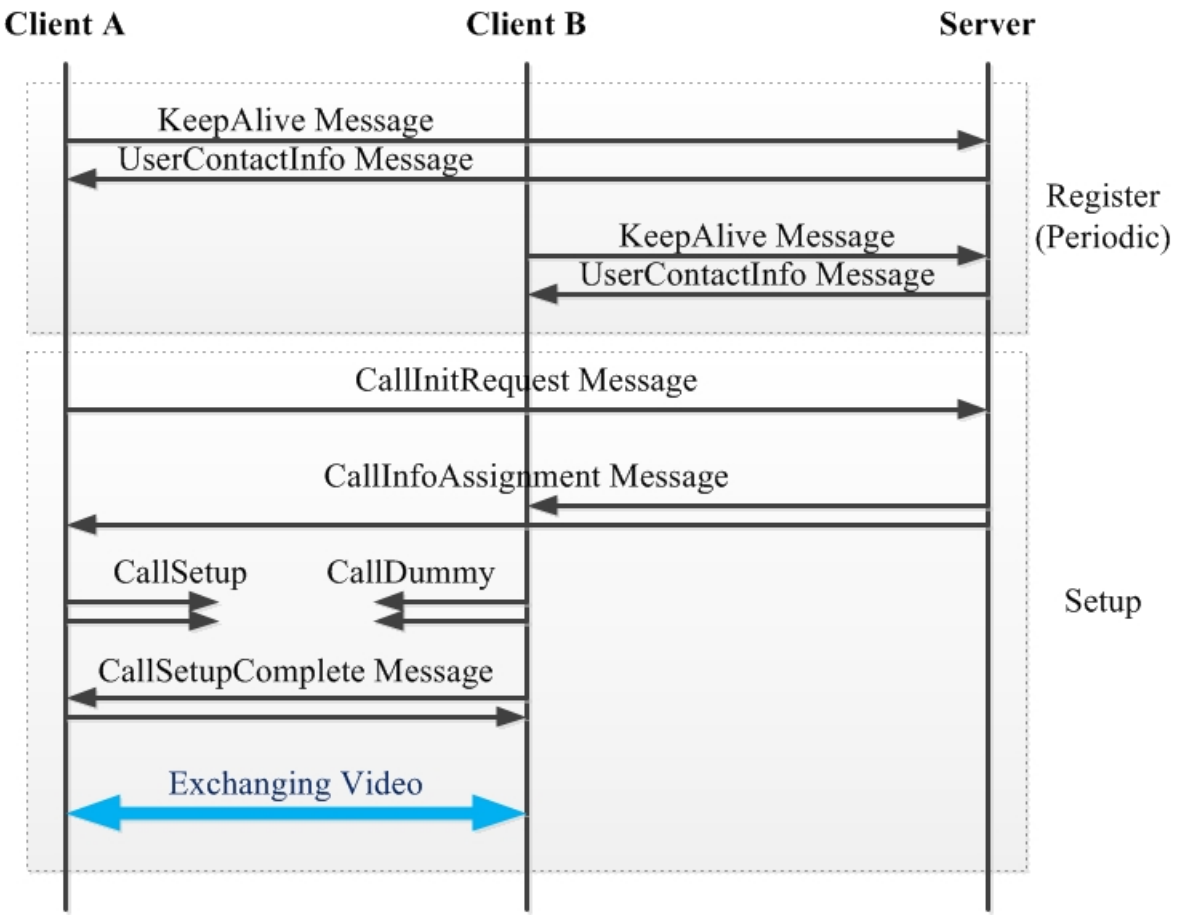


Figure 3.4: NAT-enabled MobileASL Call Flow (A is calling B).

subtraction, multiplication, selection, pack and saturation. Operations are performed in parallel.

Motion Estimation

Motion estimation is the most time-consuming module in video coding. The sum of absolute difference (SAD) is used as the distortion measure. In the ARMv6, the instruction USADA8 performs the sum of absolute differences of four 8-bit data and accumulates the results.

In H.264, there are 7 different block sizes (from 16×16 to 4×4). In the $16 \times 16, 16 \times 8, 8 \times 16, 8 \times 8$ and 4×8 modes, there are 8 pixels in one line which are stored consecutively in memory. Using doubleword loading and USADA8, only 4 cycles are needed (2 loads and 2 SAD operations) versus using 40 cycles for unoptimized case. Figure 3.5 shows this process.

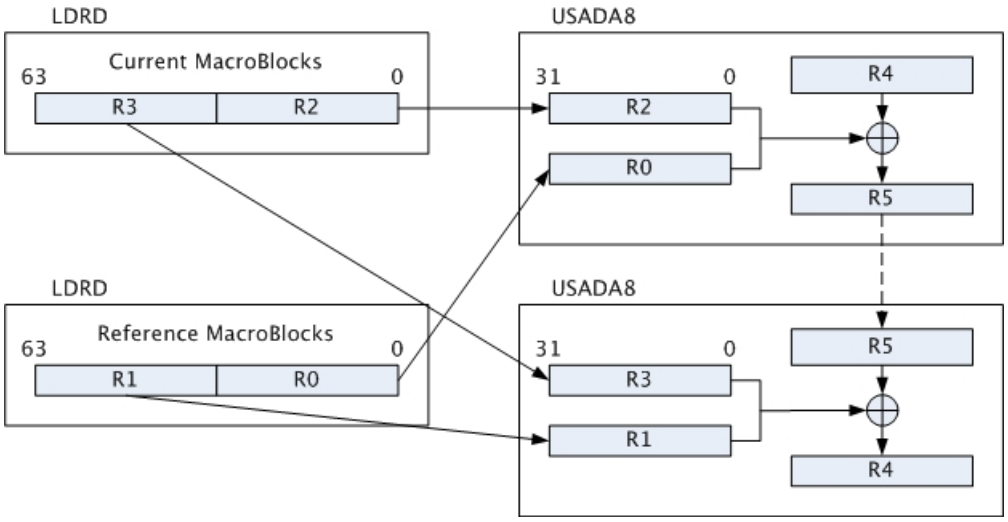


Figure 3.5: SAD calculation using the USADA8 instruction.

In the 8×4 and 4×4 modes, one line has 4 consecutive pixels (32 bits), so I can load two lines of data to the register with load instructions and then calculate the sum of absolute difference using USADA8. This costs 3 cycles.

Mode Decision

The H.264 encoder uses rate-distortion optimization to select the mode. Because of its complexity, mode selection is clearly an important candidate for speed up.

The distortion is computed as the sum of squared differences (SSD) between the original block and the reconstructed block. The SSD for 4 pixels between two macroblocks can be optimized using three SIMD instructions. There are two consecutive steps: 8-bit absolute difference (UQSUB8 and ORR) and multiplication (SMLAD). The unsigned saturating subtraction between two pixels equals zero when normal subtraction is negative. The same operation with the opposite order is performed, and the logical OR operation for those results becomes the absolute difference. Using the SMLAD instruction and a 32-bit accumulator to accumulate the sums of products of 16-bit data doubles the speed of mode selection.

The detailed operation is shown in Figure 3.6.

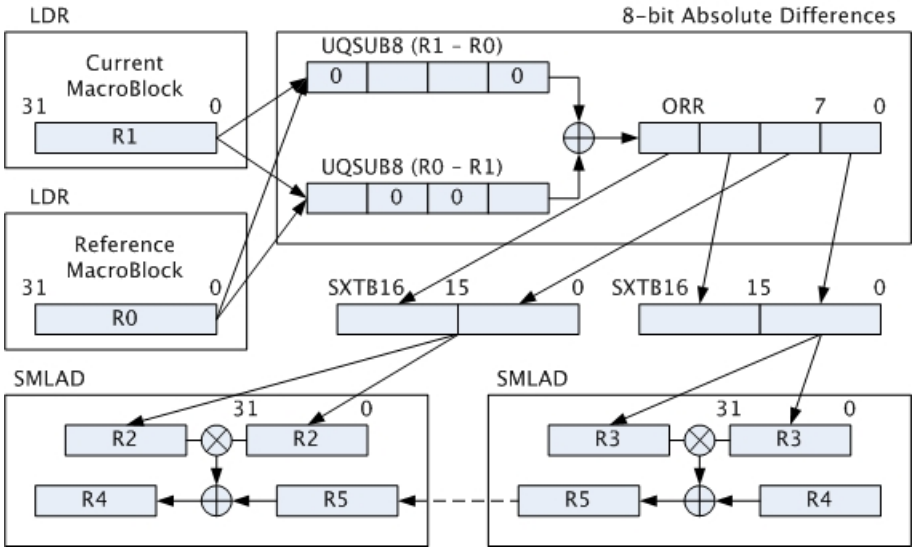


Figure 3.6: Distortion computation by SSD using SIMD.

Transforms

H.264 uses three transforms depending on the type of residual data that are to be coded: a transform for the 4×4 array of luma DC coefficients in intra macroblocks (predicted in 16×16 mode); a transform for the 2×2 array of chroma DC coefficients (in any macroblock); and a transform for all other 4×4 blocks of residual data.

The DC coefficient of each 4×4 block in the 16×16 intra prediction mode is transformed using a twice 1-D 4×4 Hadamard transform, which uses additions and subtractions (QADDSUBX, QADD16, QSUB16) and shifts (SHADD16) in 16-bit arithmetic. The other two transforms are also implemented in a similar way.

The detailed procedure is described in Figure 3.7.

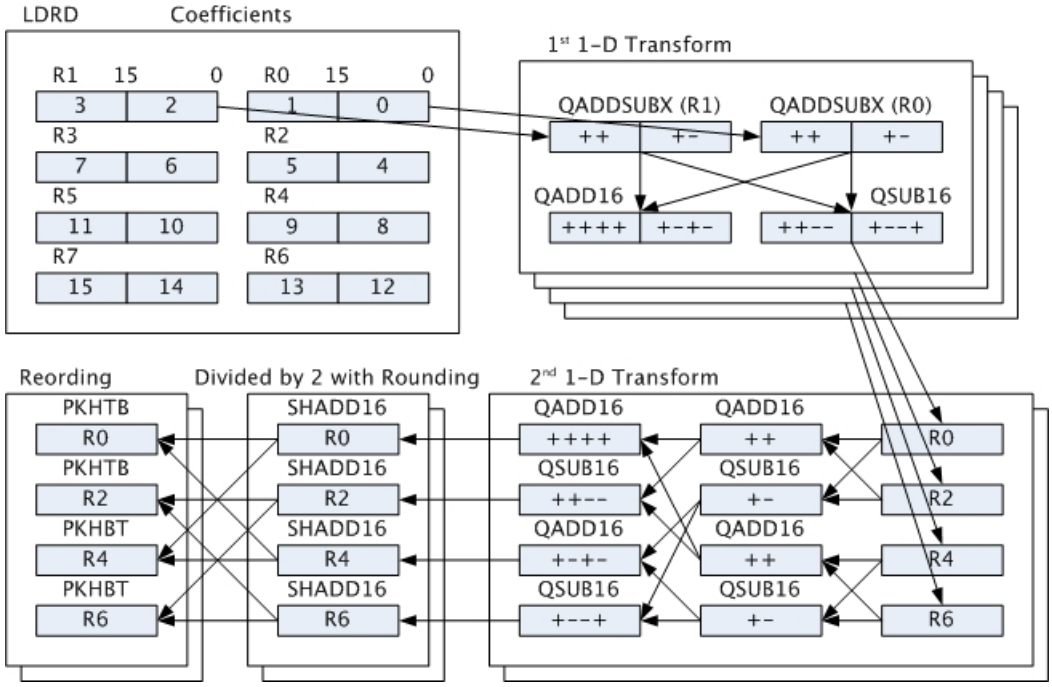


Figure 3.7: 4×4 transform for luma DC coefficients using SIMD.

3.4.2 H.264 Parameters Optimization

After I optimized the encoder in assembly language, I determined H.264 parameter settings that have low complexity but still offer high video quality at 30 kbps. Choosing x264 encoding parameters is a distortion-complexity (D-C) optimization problem [55] because it has a lot of parameters: number of reference frames (`ref`); sub-pixel motion estimation method (`subme`); partition size for intra and inter motion compensation (`part`); quantization approach (`trellis`); DCT size (`dct`); motion estimation method (`me`); motion search method (`ms`); and the use of mixed reference (`mixed-refs`).

I picked the lowest complexity settings from the result of above D-C optimization and then tested each parameter by training on 6 videos recorded with a cell phone at QCIF resolution. I found that setting some x264 encoding parameters, such as the number of reference frames and the motion search method, to their lowest complexity settings produced almost the same quality video when they were set to their highest complexity settings. Since the sub-pixel motion estimation parameter is highly complex, I chose its lowest complexity setting. I also chose I4×4,P8×8 for the partition size because it provides the best tradeoff of speed and quality.

3.4.3 Video Resolution

Even though I sped up the processing time through assembly and parameter optimization, I achieved 7-8 fps which was not high enough for my target frame rate (15 fps). Since H.264/AVC uses block-based motion compensation, the encoding time is highly related to the spatial resolution of the video. I investigated tradeoffs in spatial resolution versus speed for QCIF (176×144), 160×128, 144×112, 128×96, 112×80, 96×80, 80×64, 64×48, 48×32, and 32×16. The speed improvement is presented in later section.

The overall procedure for different spatial resolution is to encode after downsampling and then upsamples after decoding. This approach is quite good for the speed as well as the quality especially at very low bit rates. This observation is explained later in Chapter 5.

Figure 3.8: A snapshot of ROI-based Encoder (a) original frame with skin blocks (red lines) (b) 0 ROI (c) 12 ROI. The quality in skin blocks is enhanced while the quality in the rest of blocks is degraded.

3.5 Enhancing the Video Intelligibility

At 30 kbps, video quality is low. In [22], they implemented a fixed ROI by varying the level of distortion in a fixed region surrounding the face of the signer. They found that a ROI of 6 decreased quantization steps near the face of the signer (doubling the quality in that region) was preferred over a typical (no region-of-interest) encoding. I then extended this enhancement by finding face and hands regions using skin detection.

I first divided each frame into 16×16 blocks and then detected skin in real-time via a simple and well-known RGB-based algorithm [59] that works for many different skin tones. I designated macroblocks as skin blocks if they have more skin pixels than a threshold. These blocks were encoded with lower quantization parameters to increase the quality. Figure 3.8 shows which areas are considered to be skin blocks and how the intelligibility of those blocks is enhanced.

Figure 3.9 shows the MobileASL framework for enhancing the intelligibility of the video call.

3.6 Experimental Results

In this section, I present the results for encoder speed improvement using SIMD optimization, and intelligibility improvement using ROI-based encoding.

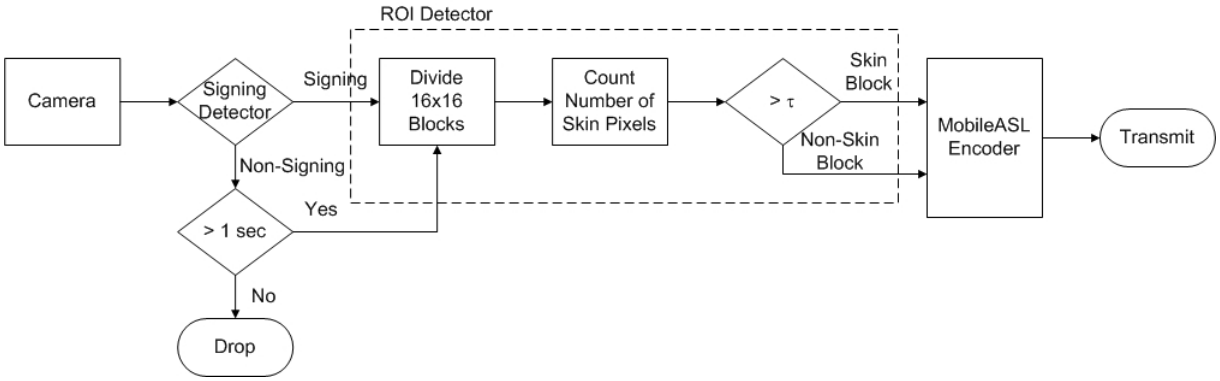


Figure 3.9: MobileASL Framework. The variable frame rate and ROI detector based on skin blocks are concatenated before encoding.

3.6.1 SIMD Performance Comparison

Experiments were conducted with fourteen QCIF (176×144) test sequences from the standard MPEG data set [9] and an ASL data set developed at the University of Washington [8], each representing a different class of spatial detail and motion. The MPEG set videos are foreman, carphone, container, grandma, missam, salesman and Akiyo and the ASL videos are accident, day in the life, education, favorite restaurant, food at home, graduation and segment8.

For the encoder only, Table 3.2 shows a frames/second comparison of my data sets for two scenarios. My instruction optimization for H.264 increased the encoding frame rate up to 15.3% (1.6 Δfps) for the MPEG data set and 13.4% (1.7 Δfps) for the ASL data set.

3.6.2 ROI Encoding

I compared the quality of the video recorded by the phone for three different ROIs (0, 6 and 12). Table 3.3 shows the PSNR for three different ROIs. The ROI PSNR for an ROI of 12 at 30 kbps corresponds to ROI PSNR for an ROI of 0 at 44 kbps. It means I can reduce the required bandwidth for same quality by 32%. As expected, an ROI of 12 increased quality in the ROI at the expense of quality in the background.

Another advantage of ROI-based encoding is its speed. Table 3.4 explains how mac-

Table 3.2: Frames Per Second Comparison with MPEG and ASL Set (encoder only).

Type	QCIF test sequence	without SIMD (fps)	with SIMD (fps)	Δfps
ASL	accident	13.6	15.2	1.6
	day in the life	13.1	14.7	1.6
	education	14.6	16.1	1.5
	favorite restaurant	12.2	13.8	1.6
	food at home	12.6	14.3	1.7
	graduation	14.6	16.2	1.6
	segment8	14.3	15.9	1.6
MPEG	foreman	9.7	11.2	1.5
	carphone	9.8	11.3	1.5
	container	11.8	13.2	1.4
	grandma	12.9	14.4	1.5
	missam	10.2	11.5	1.3
	salesman	14.7	16.3	1.6
	akiyo	14.6	16.1	1.5

Table 3.3: PSNR Comparison of Different ROI.

	30kbps 0 ROI (dB)	30kbps 12 ROI (dB)	44kbps 0 ROI (dB)
ROI PSNR	27.5	29.8	29.8
Non-ROI PSNR	29.3	24.3	31.4

Table 3.4: Macroblock Distribution in P frames for different ROI.

Macroblock Size	0 ROI (%)	12 ROI (%)
SKIP	45.6	57.7
Others	54.4	42.3

Table 3.5: Encoding Time Breakdown and Comparison for different ROI.

Encoder Function	0 ROI (ms)	12 ROI (ms)
Mode Decision	37.3	32.5
Transform and Encoding	12.2	10.0
Entropy Coding (CAVLC)	2.0	1.7
NAL and Slice Coding	2.7	2.6
Others	27.6	27.1
Total	81.9	74.0

roblocks are selected for different ROIs during the mode decision stage of encoding. Because more bits are allocated to skin blocks and fewer bits are used for other blocks, more blocks in the background are chosen as skip blocks. The increased number of skip blocks also sped up macroblock encoding, entropy coding and slice coding (see Table 3.5). In total, I sped up encoding by an additional 9.7% over the SIMD optimization.

3.6.3 Picking The Resolution

The encoder is not the only module that consumes resources in real-time video communication. The decoder, the camera interface capturing the video, the screen interface displaying the video, and the transmission module all operate simultaneously with the encoder. My encoder occupies about 50% of the CPU time for video calls. Hence, I only achieved 7-8 fps with SIMD assembly and H.264 parameter optimization.

Finally, I performed the encoding of different video resolutions for the test sequences “foreman” and “salesman” from the MPEG data set and “accident” and “graduation” from

our ASL data set. Based on the experiment with these data set, I chose the 96×80 resolution and it can achieve up to 15 fps. I interpolate the frame at the receiver to fit the screen.

3.7 Summary

In this chapter, I presented the assembly optimization and parameter selection for an H.264 encoder in order to enable real time video communication over a mobile phone. Experimental results demonstrate that I can provide 12-15 frames/second at 30kbps with a 96×80 resolution using my optimized ASL encoder in real-time on the HTC TyTN-II. My MobileASL framework is suitable for video communication in very low bit rate wireless network environments. I demonstrated the success of my system over Wi-Fi and the AT&T cellular network. A sample encoded ASL video can be viewed on YouTube at `http://www.youtube.com/watch?v=FaE1PvJwI8E`.

Chapter 4

MAINTAINING INTELLIGIBILITY FOR PACKET LOSS

Compressed video is highly vulnerable to packet loss because video coding uses spatial and temporal correlation between video frames. The most important problem that afflicts compressed video is error propagation because of inter-frame prediction. Periodically encoding some frames as intra-frames can prevent this problem, but because it requires many more bits, it is not suitable for low bit rate video communication over cellular networks.

Developing an effective algorithm to overcome the effect of packet loss is critical to maintaining intelligibility for sign language communication on mobile devices.

4.1 Related Works

A variety of error-resilient techniques have been proposed to mitigate the effects of packet loss and inter-frame error propagation. The error control schemes with source coding layer such as interleaving and forward error control (FEC) are not suitable for video telephony and video conferencing services because these services require very short end-to-end transmission delay. Moreover, schemes that request the retransmission of corrupted frames cannot be used because the current arriving frames cannot be decoded until receiving the retransmitted frame. This will delay the call.

4.1.1 Error Resiliency Features in H.264

The H.264 standard employs various error resiliency features such as Flexible Macroblock Ordering (FMO), Arbitrary Slice Ordering (ASO) and Data Partitioning (DP). Partitioning the picture into independently-decodable regions called slice groups enhances robustness to data losses by managing the spatial relationship between the regions, and preventing errors to be scattered across the whole frame. Since some encoded bitstreams in a frame are more important than others, these features also enable unequal error protection according

to their importance. For example, low and high frequency transform coefficients can be assigned to different slices and can be processed using different error protection algorithms. In addition, prediction beyond the slice boundaries is forbidden to prevent error propagation from intra-slice predictions.

However, for MobileASL, since the frames are small and encoded at a low bit rate, each predicted frame (P-frame) typically fits within a single packet. So, each occurrence of packet loss corresponds to the loss of an entire frame. Thus, the data partitioning and slice structuring done as a result of FMO and ASO in the H.264 standard are not useful for MobileASL.

4.1.2 Intra Refresh

Intra refresh is the most effective error resilient scheme in practical applications and has been extensively explored [29]. In intra refresh, intra pictures/blocks are inserted to prevent or minimize the effect of error propagation.

H.264 uses intelligent intra-block refresh by rate-distortion optimization to select between inter/intra mode at the encoder. In rate-distortion optimization, packet-loss probability models have been proposed to predict the error of the decoder's reconstructed video [62, 60]. Rate-distortion optimization can reduce the required bit budgets but doesn't guarantee prevention of temporal error propagation due to frame dropping.

I use intra refresh to maintain intelligibility by utilizing feedback. Feedback is a useful strategy that is further developed in the next section.

4.1.3 Reference Picture Selection

Another efficient way to prevent temporal error propagation is feedback-based reference picture selection, which tries to avoid using a corrupted frame as a reference frame [70]. Because the decoder sends feedback acknowledgement, the encoder can know which parts of pictures were erroneously decoded and then use multiple reference frames to stop temporal error propagation. However, in such cases, the computational complexity for motion estimation and motion compensation increases in proportion to the number of reference frames

used. Since motion estimation and compensation are the most time-consuming modules of video encoding, it causes the overall encoding time to increase. This is especially true for mobile phones. In addition, correlation between frames decreases between reference frames that are farther apart in time [45]. This, in turn, decreases coding efficiency. Moreover, when the encoder decides not to use a corrupted frame as reference, it must use a reference frame from before the error occurred. Therefore the encoder must store a number of previous reference frames. This may not be possible because of the limited memory resources of a mobile phone.

4.1.4 Error Concealment Schemes

Typically, error concealment by the decoder utilizes spatial, spectral, and/or temporal correlation of the received video. Spatial and spectral error concealment uses the information of the neighboring blocks in the spatial or frequency domains. In contrast, temporal error concealment uses the temporal correlation between adjacent frames.

The simplest method to recover the missing frames is to repeat the last received frame with all zero motion vectors, which is called temporal replacement (TR) or frame copy (FC) [66, 28]. Most of the temporal error concealment techniques assume that only a few macroblocks are lost and these techniques utilize previous frames to estimate the motion vectors associated with the lost macroblocks [49, 71]. However, in MobileASL, one data packet carries a whole frame so FC is not very useful here.

There have been methods that have addressed this as follows. Zhu et al. proposed motion vector extrapolation (MVE) in which the motion vectors of macroblocks are extrapolated from the last received frame and then the overlapped areas between the damaged block and the motion extrapolation macroblocks are estimated [52]. Li et al. proposed a pixel-based MVE (PMVE) method to conceal the missing frame which extends MVE to the pixel level [66]. Yan and Gharavi proposed hybrid motion vector extrapolation (HMVE) method based on PMVE, which uses not only the extrapolated motion vectors of the pixels, but also the extrapolated motion vectors of the macroblock to discard the wrongly extrapolated motion vectors [28].

For estimating motion vectors for whole-frame loss, the upper bound of MVE is that the original motion vectors are correctly received, but the residual information is lost. However, this upper bound can neither prevent the temporal error propagation nor remove the corrupt regions until receiving a new independent frame [28]. Therefore, any method that deals with whole-frame loss will not be appropriate for two-way sign language communication.

4.2 Low-Complexity Feedback-Based Encoding

Feedback about dropped frames can be useful to recover data loss in two-way communication. This requires minimal additional complexity because lost frames can be detected by numbering the packets or tracking the frame number contained in a packet.

Some visual distortion will exist between the time of packet loss and the time that the feedback is received. Because MobileASL conversations are in real-time and users can ask for clarification if necessary, this short distortion may not be too disruptive.

Whittle investigated a feedback method to maintain intelligibility of ASL Video in the presence of data loss [61]. I further developed this method to satisfy both required bits and quality for low bit rate communication using the simulator of 3G cellular system. I focus on several strategies that utilize feedback with the scheme mentioned above and compare the results.

4.2.1 Bursty I-frame Refresh

The simplest way to stop temporal error propagation is to use intra-frame encoding whenever feedback about a lost packet is received. Unfortunately the losses over a wireless link are sometimes bursty, which can produce a larger distortion than an equal number of isolated losses [67].

A single intra-frame based on feedback can be used efficiently for both isolated loss and bursty losses. When the loss is detected at the decoder, feedback indicating the necessity of an intra-frame is transmitted to the encoder. However, additional frames may be lost before the corrected intra-frame is received by the decoder and the decoder will generate feedback again. When the encoder receives the additional feedback requesting another intra-frame within the round trip time, this request can be safely ignored because the initial intra-frame

already terminated the error propagation at the decoder. Moreover, the encoder can recover the corrupted video without knowing about future losses.

4.2.2 Bursty P-frame Refresh

Intra-frame refreshment based on feedback can be further enhanced by exploiting temporal correlation through inter-frame prediction.

In inter-frame prediction, some macroblocks are coded in the SKIP mode if they are in regions that do not change over time, such as the background. If there are macroblocks that are unchanged during the time from before the loss occurred to until the recovery frame is received, these macroblocks can be reused in a P-frame. Therefore, I can reduce the number of intra-blocks needed to refresh the video by keeping track of which blocks are coded as SKIP at the encoder. In ASL communication, the portion of SKIP blocks are generally high since there is no motion in the background.

Now, instead of an intra-frame, a P-frame that has only intra-blocks and skip-blocks can be used to stop error propagation in response to feedback about the loss. The resulting encoded frames will have fewer intra-blocks and thus fewer bits. The unnecessary use of intra-blocks in the background when using intra-frame can cause a flickering effect because of color mismatch between frames, especially when the frames are encoded at a lower bit rate. Therefore, the use of SKIP blocks in the background can further improve the quality when using P-frame.

Furthermore, for bursty losses, this algorithm works similarly to intra-frame refreshment. After the first P-frame is received, the video quality after the P-frame will be smooth even if further loss occurs within the round trip time. This algorithm can be easily implemented using only one variable per macroblock to keep track of how many times skip-blocks have been continuously used.

4.3 Results

This section contains simulated results of my techniques. I present comparisons between my approaches and the no-feedback case to demonstrate the effectiveness of utilizing feedback.

4.3.1 Simulation Tools

To evaluate the efficiency of the proposed approach, experiments were conducted using a network simulator provided by the 3GPP video ad hoc group [21]. This network simulator has been used in various ways. Tizon et al. used this simulator to evaluate the efficiency of their approach on cellular networks [51]. Devadoss et al. implemented their own module with the same behavior that takes care of fragmenting packets into RLC (Radio Link Control) frames, reassembling the received ones into IP level packets, and also introduces link losses and delays as specified [39]. Singh et al. conformed the behavior described in this simulator to investigate rate adaptation mechanisms for conversational 3G video [64].

This software provided by the 3GPP video ad hoc group is implemented to simulate an RTP streaming session over 3GPP networks (GPRS, EDGE, and UMTS) offline. The network parameters throughout the session in the simulator are nearly constant. Error masks, which are used to inject errors at the physical layer, are generated from link-level simulations at various bearer rates and block error rates to simulate packet errors. If the RLC-PDU (Radio Link Control - Protocol Data Unit) is corrupted or lost due to errors at the lower layer, it is discarded and then not given to the upper layer. Moreover, this simulator deals with actual transmission time over the networks using frame sizes of RLC layer and scheduling. If the maximum transfer delay caused by retransmission or buffering at lower layer is reached, the corresponding RTP packet is also discarded.

I assumed a fixed round trip time, measured in frames, denoted by $rttf$. For my results, the round trip time was set equal to seven frames in the simulator. This is reasonable because the video runs at 15 frames per second and the latency of the cell-phone network is estimated to be 500ms. In practice, the round trip time in frames could be calculated based upon actual network conditions, thus calibrating the recovery to be most effective and efficient.

4.3.2 Simulation Results

The simulation was carried out a video taken by the HTC TyTN-II. This video is centered on an ASL signer who occupies approximately the center third of the frame (see Figure 4.1).

(a) Frame Copy (b) Bursty I-Frame Refresh (c) Bursty P-Frame Refresh

Figure 4.1: Visual comparison of recovery techniques. Figure shows frame number 325 of the test sequence.

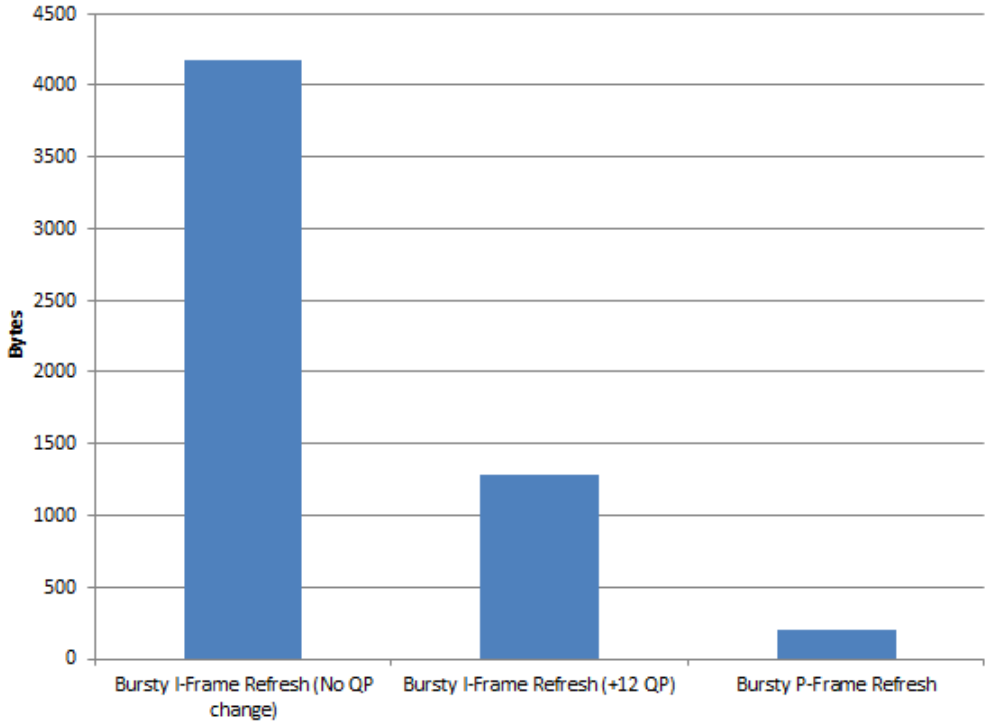
Frame Copy	Bursty I-Frame Refresh	Bursty P-Frame Refresh
11.01 dB	18.50 dB	18.49 dB

Table 4.1: Average PSNR from simulation for test sequence.

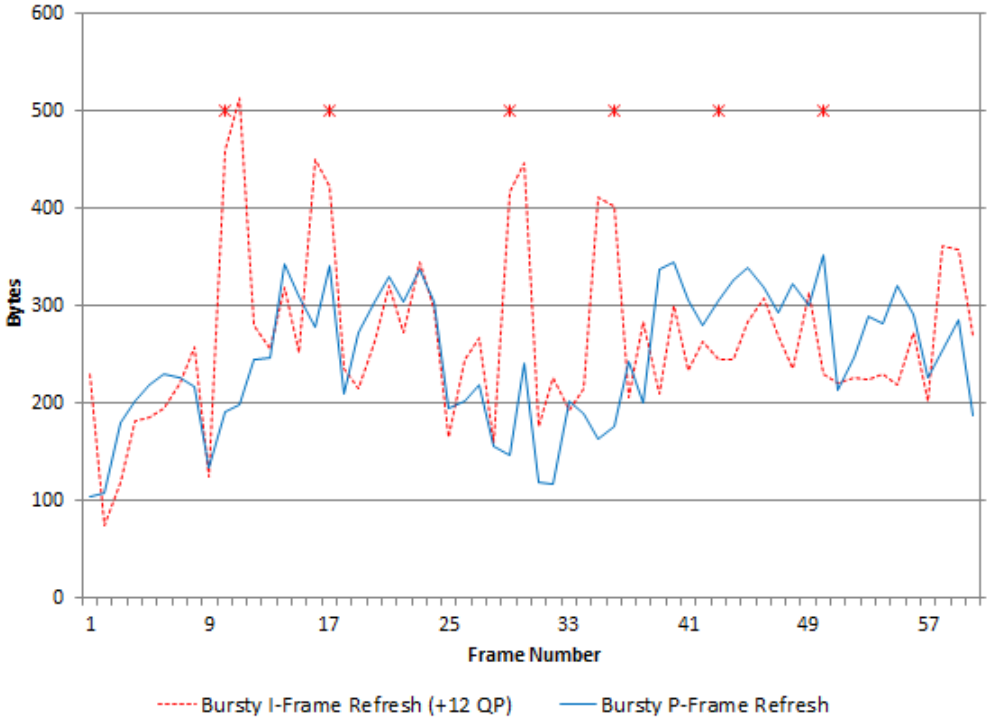
The phone was set down on the table and the user sometimes moved forward and backward. The resulting video has a quasi-static background.

4.3.3 Measuring Intelligibility

ASL intelligibility is difficult to measure. To substitute for user-studies at this stage of research, we instead measure the peak signal to noise ratio (PSNR), which is a common measure of distortion that results from image compression. The PSNR is calculated for corresponding frames between the simulated video and the source video. Since PSNR is inversely related to the mean squared error of the simulated video, a higher PSNR usually indicates a better reconstruction. Although this metric does not directly correspond to the intelligibility of ASL, the average PSNR over all frames of a particular video can indicate the persistence of distortion introduced by data loss. In simulation, the average PSNR for the decoded video was 7.48dB higher when using my techniques than when frame copy method was used (Table 4.1).



(a) for second loss



(b) for 4 seconds

Figure 4.2: Resulting Packet Size for different recovery techniques on the test sequence. Above figure shows packet size for second loss and below figure shows a snapshot for 4 seconds. Red stars indicate lost frames. Note that the y-axis scales are different.

4.3.4 Analysis

First of all, bursty I-frame refresh was investigated for two different cases by adjusting the quantization parameter (QP): no QP change and +12 QP only for intra-frame because encoding a frame as an intra-frame takes many more bits than as an inter-frame (Figure 4.2). Increasing the QP by 12 lowers the PSNR but the corresponding packet size is smaller, comparable to bursty P-frame refresh.

Now I compare adjusted bursty I-frame refresh with bursty P-frame refresh. From Figure 4.3 it appears that both adjusted bursty I-frame refresh and bursty P-frame refresh methods have the same PSNRs on average. The bursty P-frame refresh recovered more quickly from a loss than adjusted bursty I-frame refresh. Each method recovered once feedback was received. This is only part of our goal, since I am also concerned with each method's consumption of bits and added computational complexity.

As indicated earlier, the number of intra-blocks used by a recovery method greatly affects the size in bits of a refresh frame. The intra frame refresh method produces refresh frames that use the most bits, which is observed in Figure 4.2. The resulting frame size is 4171 bytes for bursty I-frame refresh (no QP change), 1282 bytes for bursty I-frame refresh (+12 QP), and 204 bytes for bursty P-frame refresh.

4.4 Summary

Since video compression utilizes spatial and temporal correction, the quality can't be recovered until receiving a new independent frame even if the decoder estimates the motion vectors perfectly. Feedback-based encoding is the best approach for MobileASL.

Adjusted bursty I-frame method is quite good but it has a severe undesirable flickering effect because of background change and recovery time. Therefore, the bursty P-frame method is the best algorithm in terms of required bits and quality because there is no flickering effect induced by a new intra-frame.

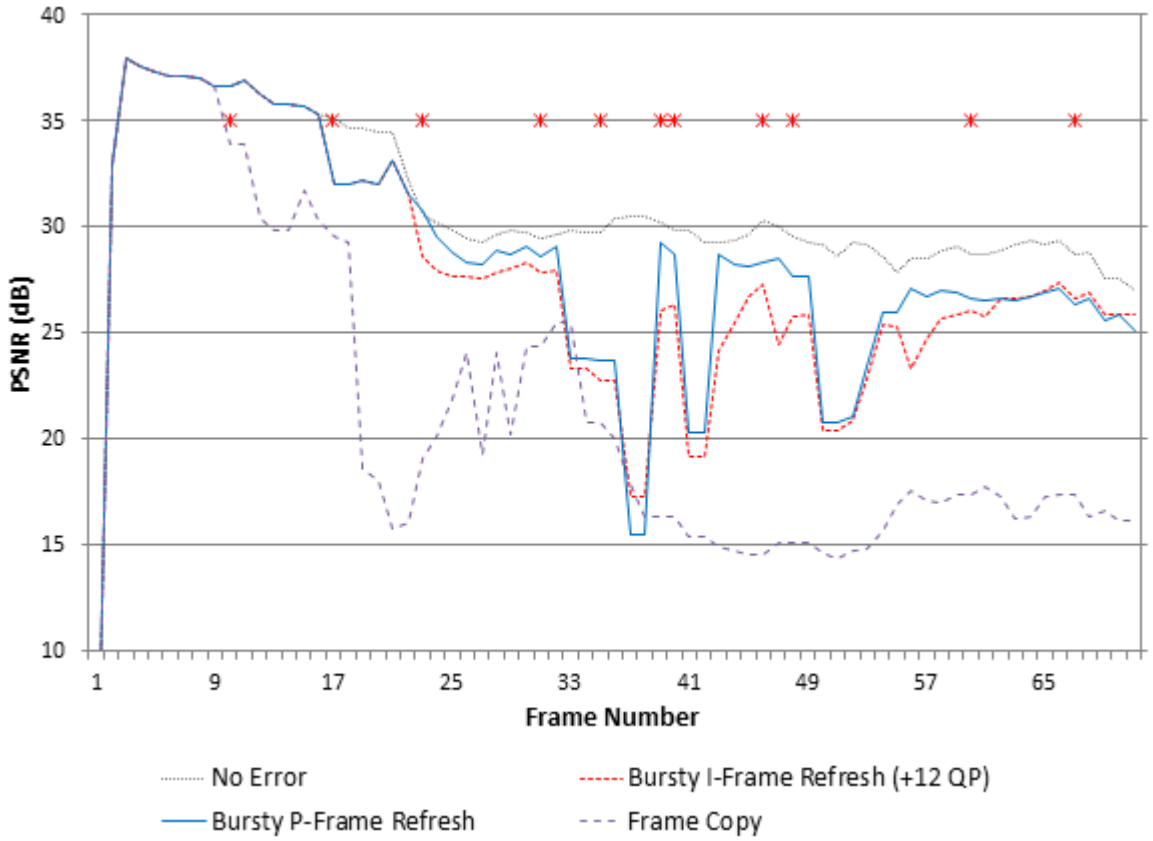


Figure 4.3: Resulting PSNR for different recovery techniques on the test sequence. Red stars indicate lost frames. Note that the bursty P-frame refresh recovers more quickly from a loss than adjusted bursty I-frame refresh.

Chapter 5

INTELLIGIBILITY FOR SIGN LANGUAGE VIDEO COMMUNICATION

Mobile video calls have been of continual interest to the Deaf community. Since sign language is a form of visual communication, it is important that video calls maintain intelligibility of signed conversations. There are many things that affect the intelligibility of mobile video calls, including image quality, frame rate, video resolution, packet loss ratio, and delay (or jitter). Intuitively, all of these items will interact with each other to affect intelligibility. For example, at a specific bit rate, a video running at 1 frame per second will have a higher image quality than a video running at 15 frames per second because all the bits would be used to encode just 1 frame. However, a video running at 1 frame per second will not have higher intelligibility, because not as much information is being shown over time. As another example, image quality goes up with the bit rate, so bit rate can be considered a part of image quality. Furthermore, the image quality is also affected by video resolution and packet loss. The interactions between all the items are summarized in Table 5.1. It means that image quality is affected by frame rate, video resolution, packet loss, but not by delay or jitter. Therefore, coming up with an equation to describe all the interactions between these items would be extremely difficult.

I first explain the relationship between the video resolution and image quality (i.e., bit rates).

5.1 Video resolution and bit rates

Since MobileASL aims to enable sign language video communication over the current U.S. cellular network on mobile devices, there are some limitations that affect video calls such as lower bandwidth and smaller screen size than for personal computers connected to a broadband network at home.

As explained in Chapter 3, one of the obvious facts for lower spatial resolution in video

x-axis / y-axis	Image Quality	Frame Rate	Video Resolution	Packet Loss	Delay & Jitter
Image Quality	-	Yes	Yes	Yes	No
Frame Rate	Yes	-	No	Yes	Yes
Video Resolution	Yes	No	-	No	No
Packet Loss	Yes	Yes	No	-	Yes
Delay & Jitter	Yes	Yes	No	Yes	-

Table 5.1: Interactions between all items. For example, image quality in y-axis is affected by frame rate, video resolution, and packet loss in x-axis but not by delay and jitter. Likewise, frame rate is affected by image quality but frame rate is independent of video resolution.

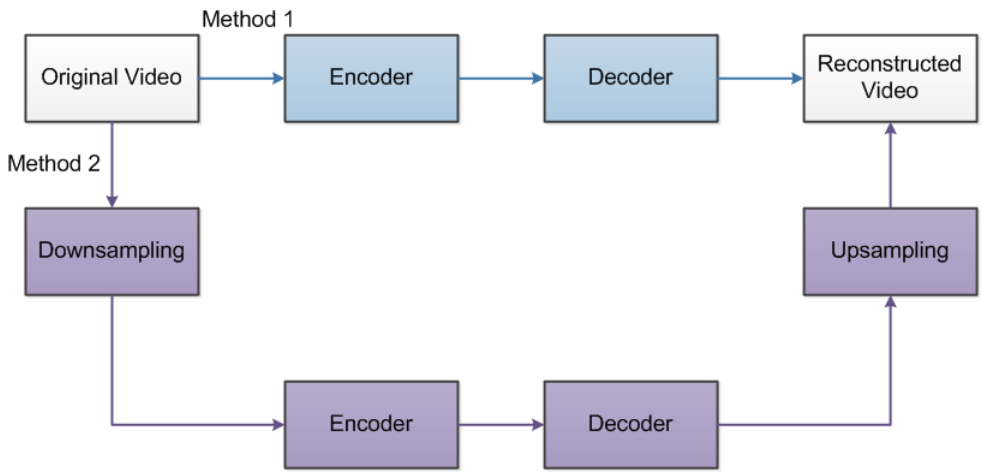


Figure 5.1: The steps for two different encoding approaches. Method 1 (blue) is just a regular encoding and decoding and method 2 (purple) encodes after downsampling and then upsamples after decoding.

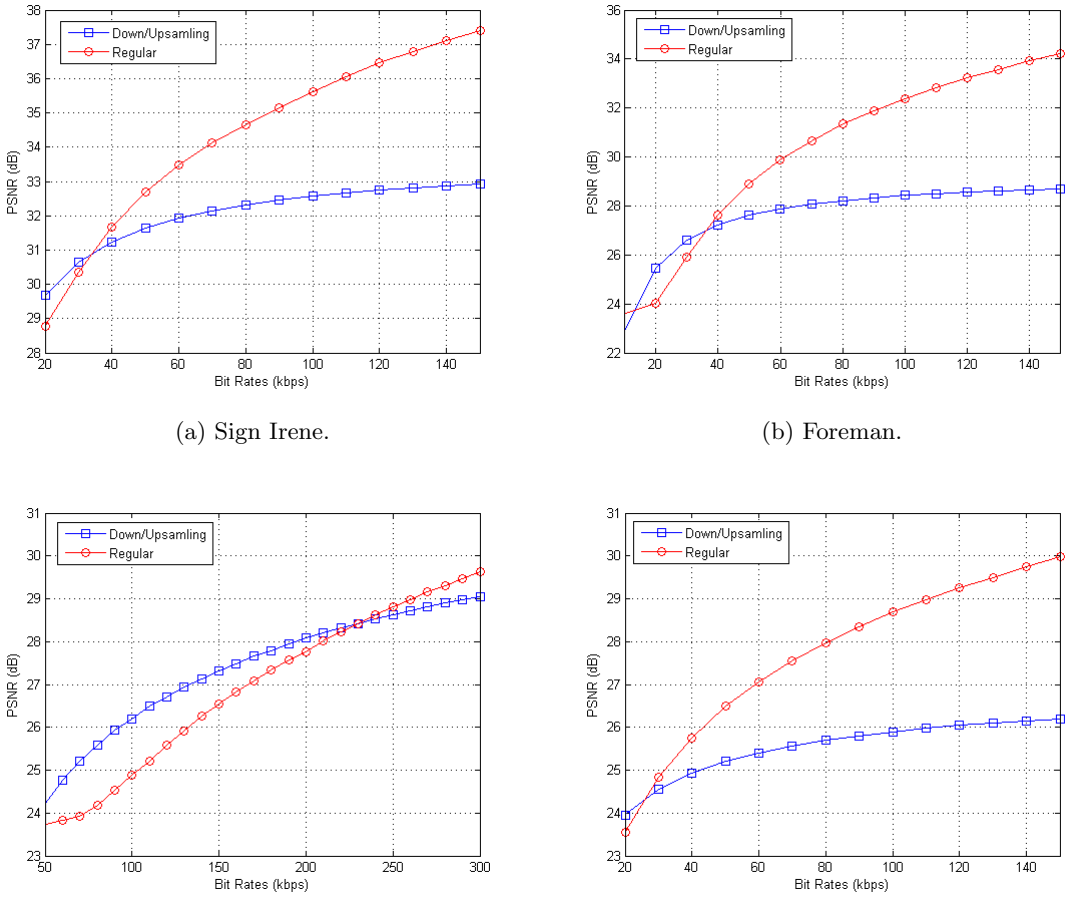
compression is the speed improvement because of fewer macroblocks. Here is another interesting point for lower spatial resolution. I argue that a video with a lower spatial resolution at very low bit rate will have higher quality than a video at a higher spatial resolution running at the same bit rate. Figure 5.1 shows the steps for two different encoding approaches at low bit rates. Method 1 is just a regular encoding and decoding. Method 2 encodes after downsampling and then upsamples after decoding. By downsampling and upsampling, more bits can be allocated per macroblock in a frame since there are fewer macroblocks. However, the corresponding decoded video will be blurry because of upsampling. Furthermore, regular video encoding at very low bit rates leads to blockiness because there are not enough bits per macroblock. Therefore, it is a tradeoff between blurry or blocky images.

First of all, I calculate the PSNR of two different spatial resolutions, CIF (352x288) and QCIF (176x144), at every bit rate 10 kbps to 300 kbps, increasing 10 kbps at a time. Figure 5.2 shows the PSNR curves for different video sequences. At very low bit rates, the PSNRs of videos at lower spatial resolutions are higher than the PSNRs of the same videos at higher spatial resolution.

The reason that the PSNR curves cross is because at very low bit rates, the encoder must code the higher resolution video with more coarse quantization. This causes severe blockiness and the loss of high frequency details. A lower resolution version of the video does not need to be quantized as coarsely for the same number of bits. When it is then interpolated to the higher resolution, it may wind up with higher quality. As the bit rates are increased, this effect decreases. The video will have higher quality at higher resolutions since it will not be blurry due to the interpolation step.

An interesting observation from Figure 5.2 is that the bit rate at which the PSNR curves cross differs depending on the content of the video. Table 5.2 shows the crossover bit rates and average absolute pixel difference between consecutive frames. As expected, videos with a higher pixel difference value between frames also have a higher crossover bit rate. For the Football sequence, which is a high motion sequence, the crossover bit rate is 230 kbps, which is a big difference from that of the Foreman sequence, which has a crossover point at 37 kbps.

Figure 5.3 shows snapshots for two different video encodings at 70kbps. The left figure



(a) Sign Irene.

(b) Foreman.

(c) Football.

(d) Coast Guard.

Figure 5.2: PSNR curves for different bit rates for various video contents at CIF (352x288) resolution.

Video Sequences	Crossing Bit Rate	Average Absolute Pixel Difference Between Frame
Foreman	37 kbps	8.2841
Football	230 kbps	20.5581
Sign Irene	32 kbps	3.4759
Coast Guard	26 kbps	9.8463

Table 5.2: The crossing bit rates and average absolute pixel difference between frames for the different video sequences. Sequences with higher absolute pixel difference between frames have higher crossover bit rates.

(a) Down and Upsampling (b) Regular

Figure 5.3: The visual comparison of video encoded with two different approaches at 70kbps. These are screen captures of the 121st frame of the Football sequence. Note that the numbers are more clear for the video downsampled before coding.

shows the encoding with downsampling and upsampling, and the right figure shows regular encoding. Noticeable differences between the two screenshots can be found at the jersey numbers on the back of the players, the outlines of the bodies, and the background. The average PSNR is 25.21dB for the downsampling and upsampling encoding and 23.92dB for regular encoding. The video quality that results encoding with downsampling and upsampling is better than the video quality from regular encoding for both subjective and objective measures for a certain range of bit rates.

5.2 Frame Rates and Bit Rates

The frame rate is the next item that affects intelligibility in sign language video communication. I investigated how an objective measure for intelligibility might be modeled based on frame rate and the bit rate. There is existing work on this, which will be addressed in the chapter 5.2.1, but it is not perfect. I present a different method for developing a better intelligibility metric.

5.2.1 Prior Work

Modeling the intelligibility of video communication is difficult because intelligibility is affected by many different factors such as frame rate and image quality.

Ciaramello and Hemami developed an objective metric for predicting the intelligibility of coded ASL video [31]. They found that distortion in the face and hand has a larger impact on intelligibility than distortion in other areas of the frame. Therefore, they considered three different coding parameters: bit rate, frame rate, and region-of-interest rate allocation, as follows.

$$Dist_I = \frac{1}{N}\sum_{n=1}^{N} \beta_{FPS}(W_F MSE_F(n) + W_H MSE_H(n)) \tag{5.1}$$

$$I = 10 \log \frac{255^2}{Dist_I} \tag{5.2}$$

where, W_F and W_H are the weights for face and hands. β_{FPS} is a weight for frame rate.

Ciaramello's metric focused on more distortions in each frame rather than on the frame rate of sign language. The frame rate can be variable depending on network conditions and contents. In addition, this metric is difficult to implement in real-time on the phone because it requires face and hand detection, which is a computationally expensive, and thus slow, process; at best, this metric ran at approximately 2.8 frames per second [32].

5.2.2 Intelligibility Model

As explained earlier, sign language video running at 6 frames per second can be intelligible when the speaker is signing very slowly. From this observation, I estimated that video at 8 frames per second would result in medium intelligibility, and that video at more than 12 frames per second would be most intelligible. Based on this assumption, intelligibility for frame rate can be formulated with a sigmoid function (see equation 5.3) because the sigmoid accelerates the value near 8 frames per second and approaches a maximum at more than 12 frames per second. Furthermore, it explains the intelligibility well for very high frame rates such as 20 or 30 because I model intelligibility as intelligibility flatters out over 13 frames per seconds.

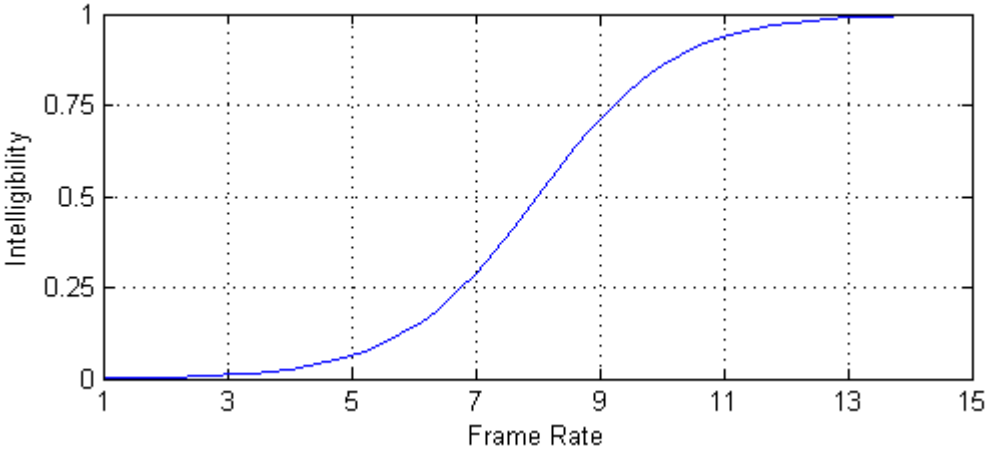


Figure 5.4: Individual intelligibility for frame rate using a sigmoid function. Medium intelligibility is achieved for 8 frames per second.

Table 5.3: ITU-R quality and impairment scale.

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

$$I_{frame\ rate}(x) = \frac{1}{1 + e^{-\alpha(x-8)}} \tag{5.3}$$

where x is frame rate and α is 0.9.

Figure 5.4 plots intelligibility vs. frame rate.

PSNR is a basic metric for image quality, and is calculated by computing the MSE of each pixel between the original and received frames. The Mean Opinion Score (MOS), specified by ITU-T recommendation P.800 [7], is a measure of quality as perceived by a human user, and is given on a scale from 1 (worst) to 5 (best) (see Table 5.3). Klaue

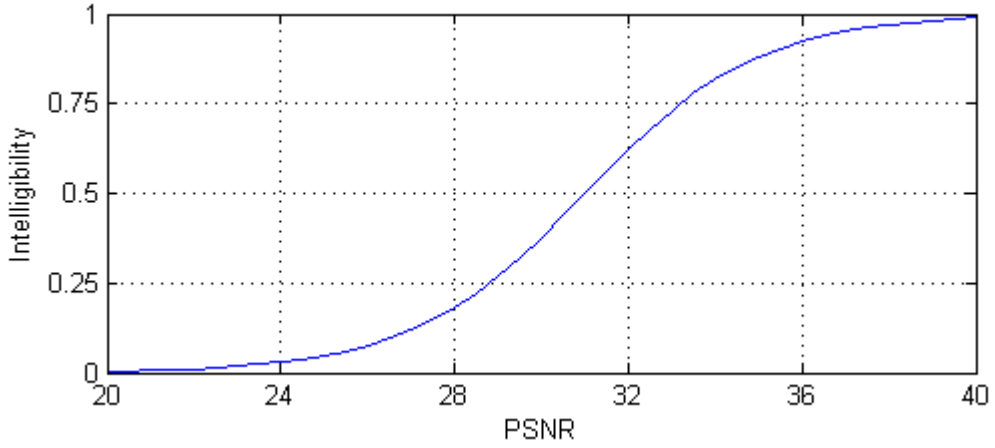


Figure 5.5: Individual intelligibility for PSNR using a sigmoid function. Medium intelligibility is achieved for 31 dB in PSNR.

Table 5.4: Possible PSNR to MOS conversion [42].

PSNR	dB	MOS
> 37	5	Excellent
31 - 37	4	Good
25 - 31	3	Fair
20 - 25	2	Poor
< 20	1	Bad

et al. proposed a mapping between PSNR and MOS for video quality evaluation (see Table 5.4) [42]. Therefore, intelligibility can be mapped to PSNR similarly using a sigmoid function (see equation 5.4). Medium PSNR for intelligibility is chosen to be 31 dB from Table 5.4.

$$I_{PSNR}(y) = \frac{1}{1 + e^{-\beta(y-31)}} \tag{5.4}$$

where y is PSNR and β is 0.5.

Figure 5.5 shows the curve of the intelligibility for PSNR.

In general, higher PSNR indicates better video quality, but this doesn't mean that higher PSNR indicates higher intelligibility. This fact can also be applied to frame rate. Ou et al. proposed video quality as the product of a temporal quality factor (a function of frame rate) and a spatial quality factor (a function of PSNR) [69]. Therefore, we model total intelligibility as:

$$I_{total} = I_{frame\ rate}(x) \times I_{PSNR}(y) \tag{5.5}$$

From Figure 5.6, maximum intelligibility can be achieved when both the PSNR and the frame rate are at their highest values. However, intelligibility is at its lowest both when the frame rate is at its lowest (even if the PSNR is high) or when the PSNR is at its lowest (even if the frame rate is high). It is clear that the intelligibility is not a function strictly of frame rate or PSNR.

In the next section, this model will be utilized to investigate how frame rate interacts with spatial quality, which is affected by encoding parameters and video resolution for sign language communication.

5.2.3 Tradeoff between Spatial and Temporal Quality

In order to investigate how to provide high video quality during calls, I investigate the tradeoffs between spatial and temporal quality based on the above model. There are two additional parameters that affect intelligibility of a video: the video encoding parameters and the video resolution.

Since x264, an open source H.264 encoder, is already being used in MobileASL software, I investigate how x264 encoding parameter settings will affect the overall intelligibility; because x264 has many encoding parameters, encoding time may vary widely, which may affect the intelligibility of a video. Another reason to consider the parameter settings is that mobile phone performance has tremendously improved in recent years. Therefore, there is room to use more complicated encoding parameters for better video quality.

Vanam, et al. proposed fast offline algorithms for obtaining H.264 encoder parameter settings that result in excellent distortion-complexity performance [54, 53]. These distortion-

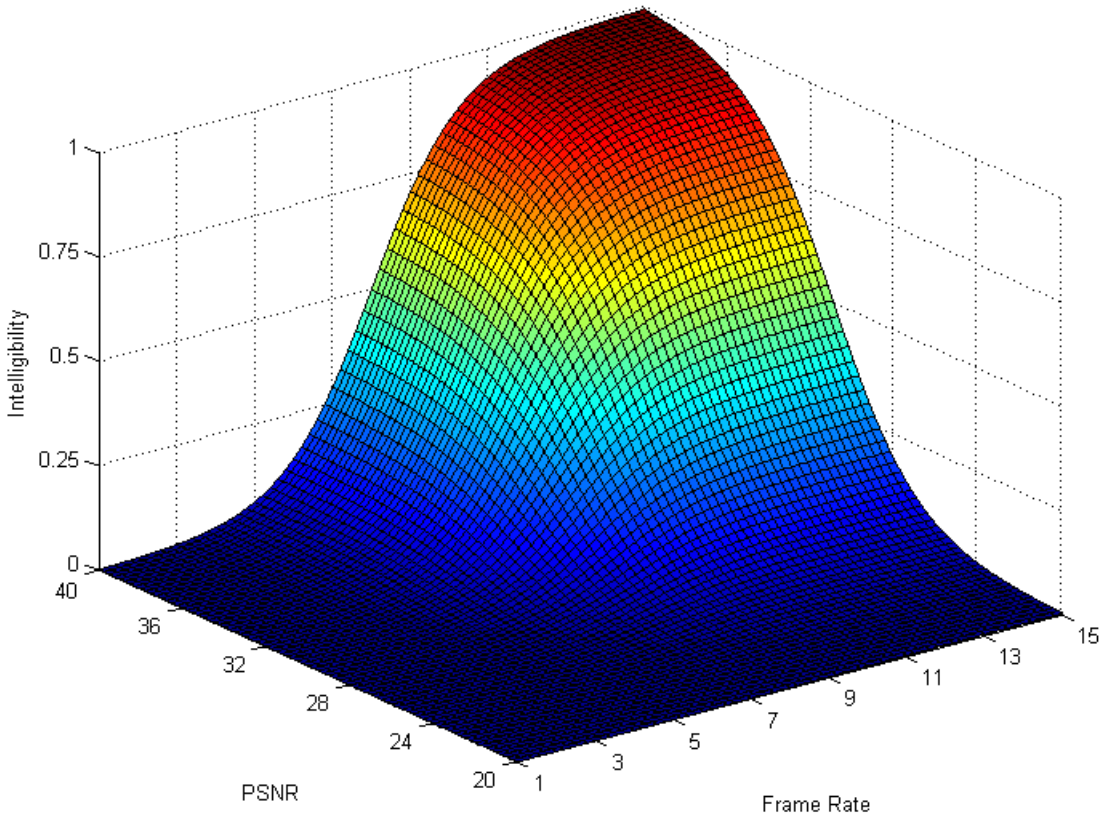


Figure 5.6: The effect of PSNR and Frame Rate on Intelligibility, which is the product of a function of frame rate and a function of PSNR. Maximum intelligibility can be achieved when both the PSNR and the frame rate are at their highest values.

Table 5.5: Presets in the latest x264 video encoder: partition size for inter motion compensation (part), motion estimation method (me), sub-pixel motion estimation method (subme), number of reference frames (ref), and quantization approach (trellis).

preset	part	me	subme	ref	others
ultrafast	none	Diamond	0	1	CAVLC, no trellis
superfast	i8x8,i4x4	HEX	1	1	
veryfast	i8x8,i4x4,psub16x16	HEX	2	1	
faster	i8x8,i4x4,psub16x16	HEX	4	2	
fast	i8x8,i4x4,psub16x16	HEX	6	2	
medium (default)	i8x8,i4x4,psub16x16	HEX	7	3	
slow	i8x8,i4x4,psub16x16	UMH	8	5	
slower	all	UMH	9	8	
veryslow	all	UMH	10	16	

complexity optimization algorithms result in parameter selections for x264, which can be stored in a look-up table for use by an online algorithm which selects parameters based on available computational resources [56]. One must need to run these algorithms for each platform and video sequence type because optimal parameter settings change for different platforms and video sequences.

The universal approach for parameter settings utilizes presets defined in x264. The latest x264 encoder provides a variety of preset encoding settings that can be applied according to target platform or application. These presets are "ultrafast," "superfast," "veryfast," "faster," "fast," "medium," "slower," "slower," and "veryslow" (see Table 5.5). In general, the settings in "ultrafast" provide the fastest encoding speed but the worst quality.

There are many new mobile phones with faster processors than the one in the HTC TyTN-II, which was used in chapter 3. I chose two different phones: the HTC EVO 4G and the Samsung Nexus S 4G. Both feature 1GHz ARM processors, WVGA (480x800) pixel displays, 3G/4G/Wi-Fi connectivity, and front-facing VGA cameras (see Figure 5.7). Due to these new features (faster processor, bigger screen size, and larger bandwidth), it is necessary

Figure 5.7: HTC EVO 4G (left) and Samsung Nexus S 4G (right).

to re-examine how determining encoding parameter settings will affect intelligibility.

The idea of mobile video calls has been of continual interest to the Deaf community. However, since sign language is a form of visual communication, it is important that video calls maintain intelligibility of signed conversations. The overall intelligibility of a video will also be affected by its content and the type of motion it contains. Figure 5.8 shows intelligibility plot for Equation 5.5 for the Foreman sequence running on the HTC EVO 4G at bit rate of 150 kbps at CIF resolution for all the preset encoding parameter settings. Until using "veryfast" setting, the intelligibility is going up. However, it is down to near the zero when higher settings are used because of lower frame rate. Figure 5.9 shows all intelligibility curves for the Foreman sequence at bit rates from 30 kbps to 150 kbps at CIF resolution for all the preset encoding parameter settings.

There are some interesting points on this graph. Lower bit rates with faster encoding settings result in higher intelligibility than higher bit rates with slower encoding settings (see Table 5.6). Furthermore, maximum intelligibility for the same bit rate is achieved by using

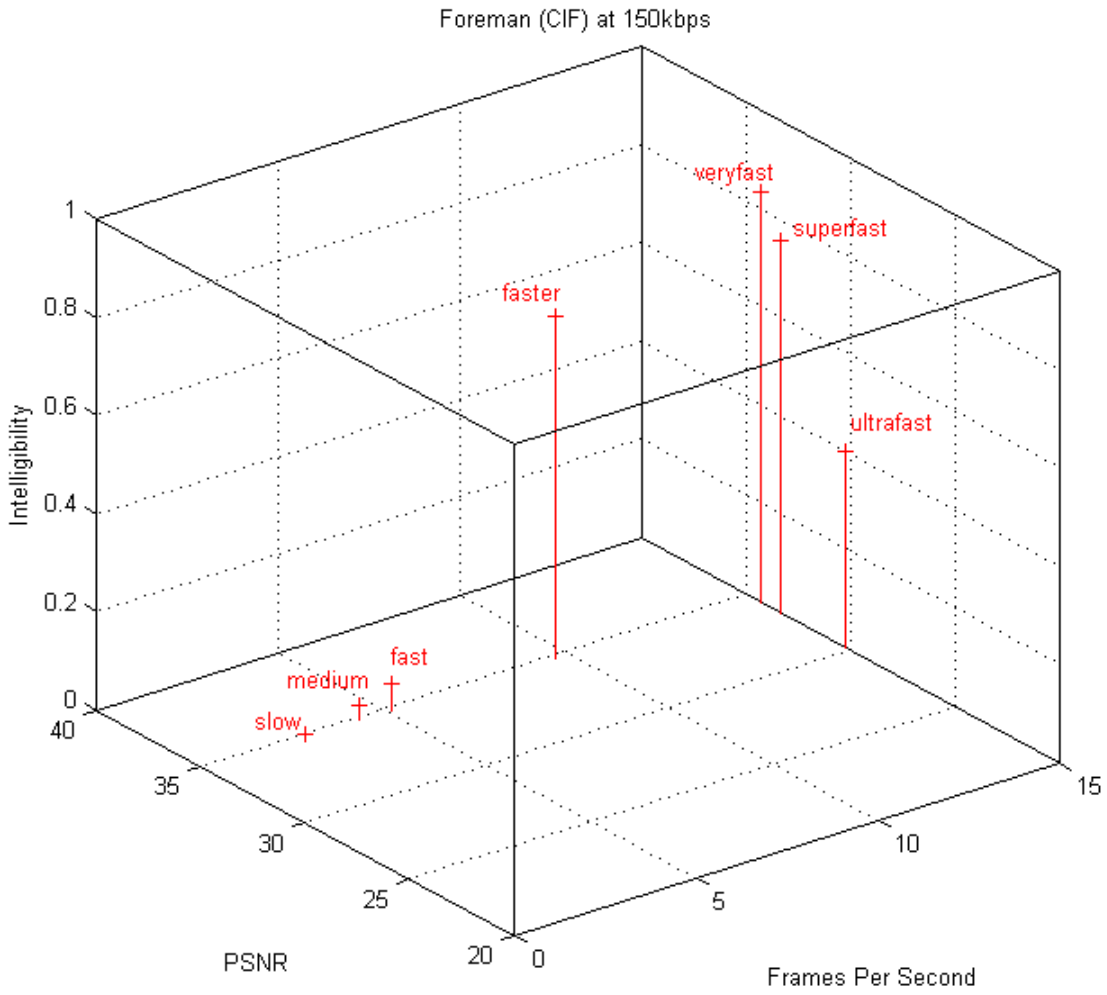


Figure 5.8: Intelligibility plots for the Foreman sequence at bit rate of 150kbps at CIF resolution, for different encoding parameters ranging from "ultrafast" to "slow" settings on the HTC EVO 4G. The maximum intelligibility is achieved with "veryfast" setting because it still provides the encoding with 15 fps.

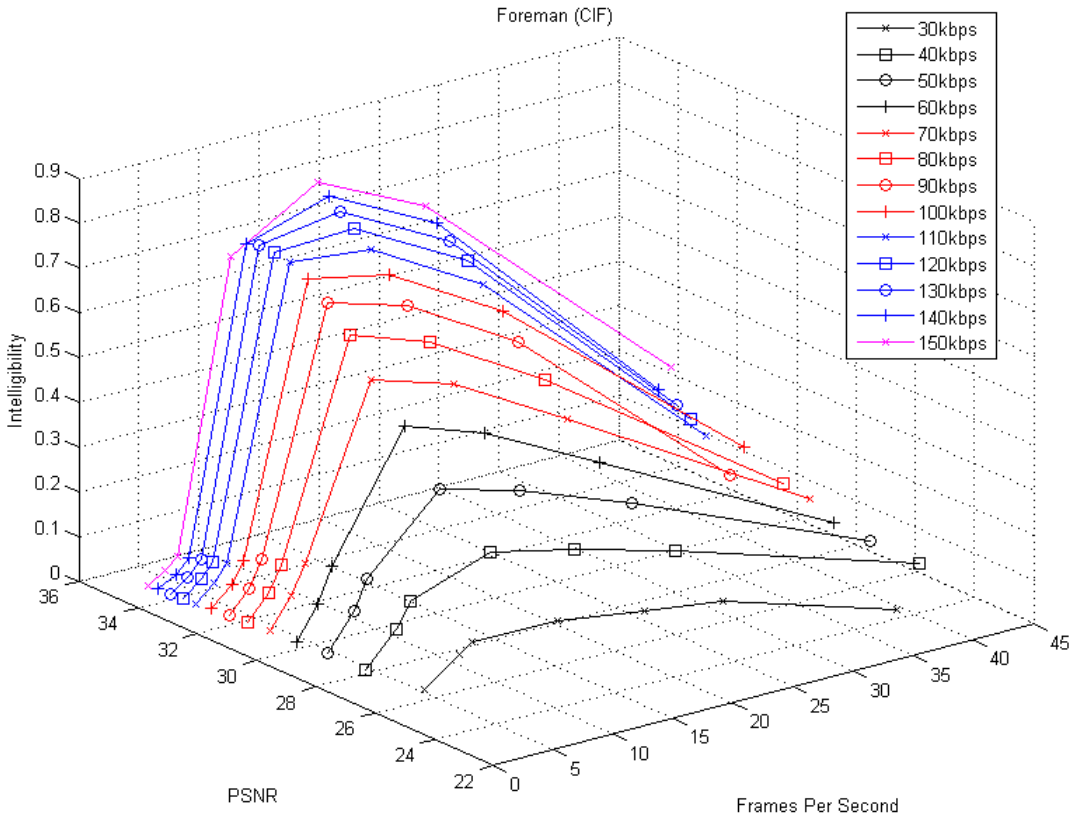


Figure 5.9: Intelligibility curves for the Foreman sequence from 30kbps to 150kbps, at CIF resolution, for different encoding parameters ranging from "ultrafast" to "slow" settings on the HTC EVO 4G. The leftmost point on the curve indicates use of the "slow" setting while the rightmost point indicates use of the "ultrafast" setting.

Table 5.6: Comparison of intelligibility for varying the bit rate and encoding parameters used for two different settings. Faster encoding settings at lower bit rates are better than slower encoding settings at higher bit rates.

Setting 1	Setting 2	Δintelligibility for Setting 1 from Setting 2
140kbps veryfast	150kbps faster	0.1132
130kbps veryfast	140kbps faster	0.0556
30kbps ultrafast	150kbps slow	0.0194
40kbps ultrafast	150kbps medium	0.0239

the “faster” setting until 100 kbps; at bit rates higher than that, maximum intelligibility is achieved using the “veryfast” setting. It is observed that when the bits per macroblock are high enough, it is better to choose a faster encoding setting because the degradation in quality caused by the encoder with a faster setting is negligible since there are enough bits per macroblock.

Figure 5.10 shows another intelligibility graph, this time of the Football sequence at CIF resolution running on the HTC EVO 4G from 30 kbps to 300 kbps. The overall shape of the curves is similar to that of the Foreman sequence, but the actual intelligibility achieved is quite a bit lower because the Football sequence contains a high amount of motion. Maximum intelligibility is achieved using different encoding parameter settings. From 50 to 150 kbps, the maximum intelligibility is achieved using the “faster” setting. For 200 kbps and higher, the maximum intelligibility is achieved using the “veryfast” setting. The crossover point of switching encoding parameter settings from “faster” to “veryfast” is at a much higher bit rate for the Football sequence than for the Foreman sequence because of the very high motion in the Football sequence.

Figure 5.11 shows intelligibility graph on sign language video, the Richard sequence at QCIF resolution running on the HTC EVO 4G from 30 kbps to 150 kbps from “ultrafast” to ”veryslow” because of lower resolution than other sequences. Likewise, this overall shape of the curve is similar. Maximum intelligibility is achieved using the “medium” setting (up to 50 kbps), the “fast” setting (up to 90 kbps), and “faster” setting (100 kbps and higher).

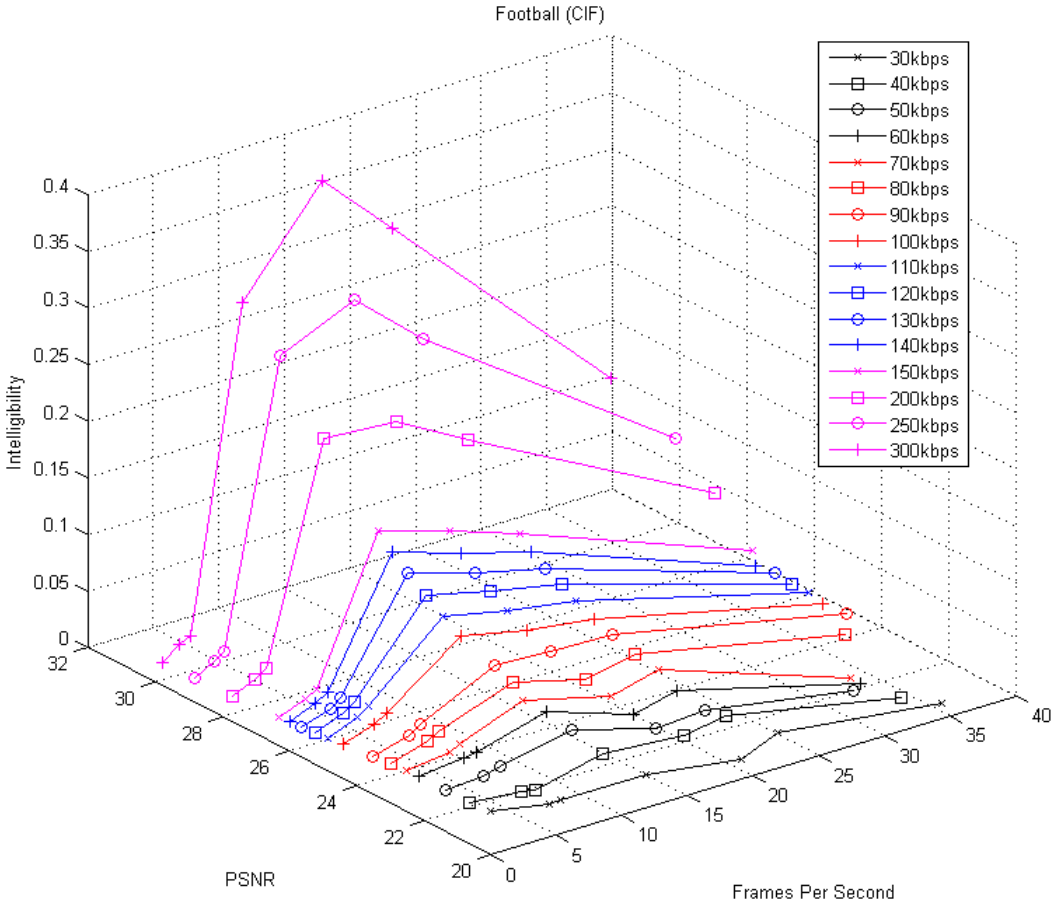


Figure 5.10: Intelligibility graph for the Football sequence from 30 kbps to 300 kbps, at CIF resolution, for different encoding parameters ranging from "ultrafast" to "slow" settings, on the HTC EVO 4G.

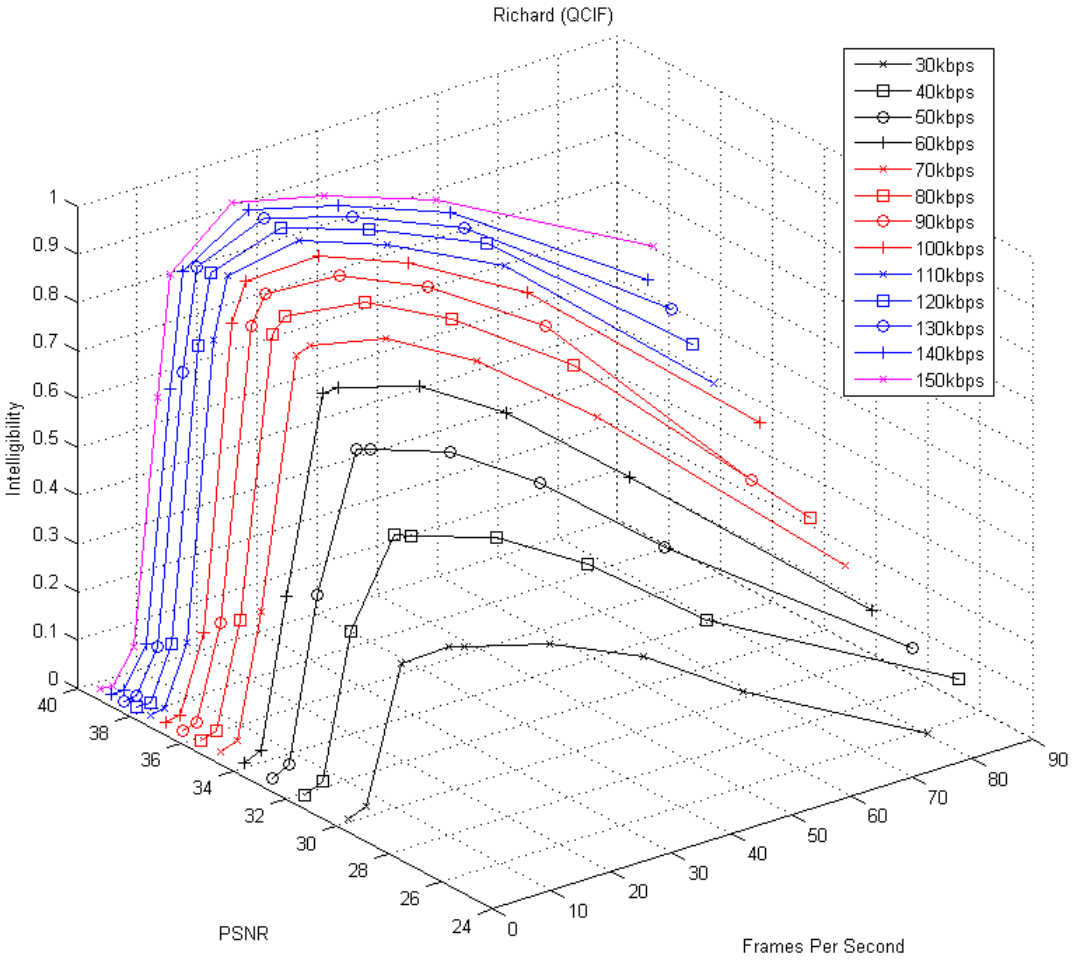


Figure 5.11: Intelligibility graph for the Richard sequence captured on a mobile phone from 30 kbps to 150 kbps, at QCIF resolution, for different encoding parameters ranging from "ultrafast" to "veryslow" settings, on the HTC EVO 4G.

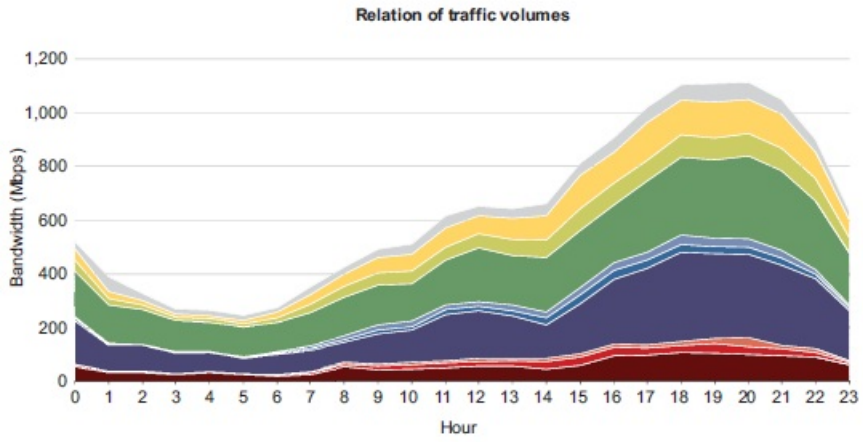


Figure 5.12: Relation of traffic volumes over time [36].

5.3 Measuring Available Bandwidth

According to the Swedish mobile operator TeliaSonera, traffic volumes vary noticeably between 2pm and 8pm, before declining rapidly until reaching their lowest usage level at 5am (see Figure 5.12) [36]. The traffic volume over wireless networks affects the quality of video calls because of variation in the available bandwidth and in the transmission time which, in turn, affects packet loss, delay, and jitter.

In [68], an experiment with four representative video conferencing systems was conducted. They concluded that the quality of a user's experience is very sensitive to bandwidth fluctuations, especially in the uplink and then suggested that an adaptive resolution/frame rate policy can be deployed. Therefore, it is necessary to have an algorithm that will encode the frame adaptively based on the available resources.

Since mobile phones are designed to use cellular networks, it is useful to measure available bandwidth to see how it fluctuates during the day. I chose to measure the Sprint 3G network since our research lab already had a Sprint phone (the HTC EVO).

For more accurate results, I made my measurements using one of the popular applications for mobile platforms, Speedtest.net (see Figure 5.13). Ookla is the global leader in broadband testing and web-based network diagnostic applications. It operates Speedtest.net

Figure 5.13: Speedtest.net Mobile for the Android platform.

using a massive global infrastructure to minimize the impact of Internet congestion and latency [10].

Figure 5.14 shows the measurements of available bandwidth on the Sprint 3G network at different times from the University of Washington in Seattle. These are two important points to consider. One is the very low available bandwidth in the uplink ($\leq 50kbps$), while the other is the very large round time delay ($\geq 500ms$). Those two numbers exceed the desirable targets for two-way video calls (see chapter 3). Unlike video streaming, two-way video calls are very sensitive to bandwidth and delay fluctuation.

5.4 Summary

In this chapter, I describe the intelligibility of mobile video calls, including image quality, frame rate, video resolution. Even though the faster 3G cellular network is available in Seattle, it can not currently be used for real-time video communication.

I proposed an intelligibility measure and then demonstrated how it can be changed by varying the bit rate and encoding parameters. The next step is to verify this intelligibility measure with users and implement it on the phone.

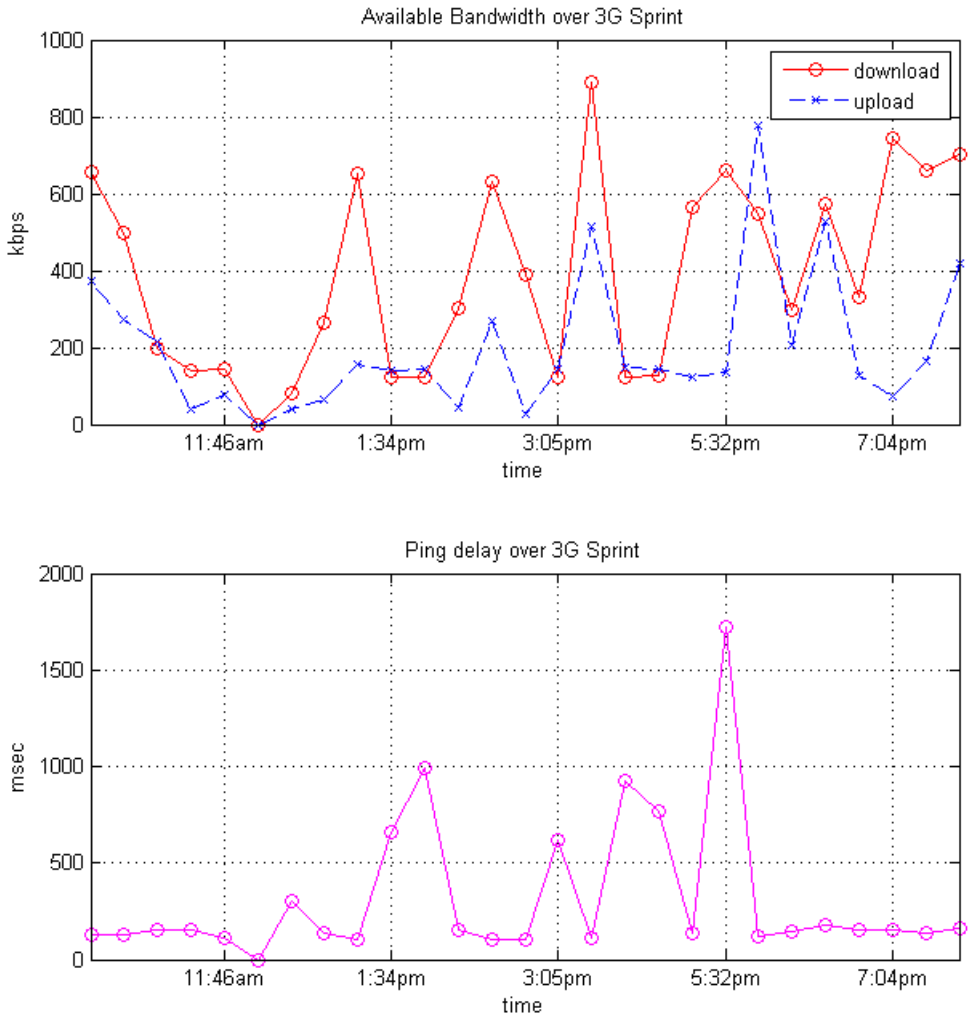


Figure 5.14: Measurements of available bandwidth and Ping delay over the Sprint 3G network at different times throughout the day at the University of Washington.

Chapter 6

PORTING MOBILEASL TO ANDROID

Mobile phones have radically changed since the first Apple iPhone came out in early 2007. Later that same year, Google unveiled Android, a smartphone operating system built on the Linux kernel. As I described in chapter 1, the market share for Android became the highest of all platforms in Q4 2010 (see Figure 1.5). Therefore, we decided to port MobileASL to Android.

Video relay service providers, such as Sorenson and ZVRS, also need to be considered in the design of the new version of MobileASL because it will expand the use of MobileASL. This means I need to use SIP or H.323 protocols, which are compatible with their systems. I cannot use the NAT-eabled MobileASL protocol designed in the previous MobileASL system.

6.1 Framework

Figure 6.1 shows the software framework for MobileASL Android. IMSDroid is the first fully featured open source 3GPP IMS client for Android devices (1.5 and later) [4]. Rather than developing software from scratch, I chose IMSDroid, a mobile video call client running on doubango [2], as a baseline platform because it uses "x264" and "ffmpeg" as its H.264 encoder and decoder. This is convenient since the MobileASL codec had been developed on "x264." IMSDroid provides an user interface for doubango, which implements network protocols for many different audio and video codecs.

In order to utilize the MobileASL codec in IMSDroid, I exchanged it with the original H.264 encoder when building the doubango library. In addition, I added two features to IMSDroid: the contact list and the field study observer, which creates logs of usage data (see Figure 6.1) directly and indirectly. While the contact list was added to the IMSDroid application itself, the field study observer is a separate application that communicates with

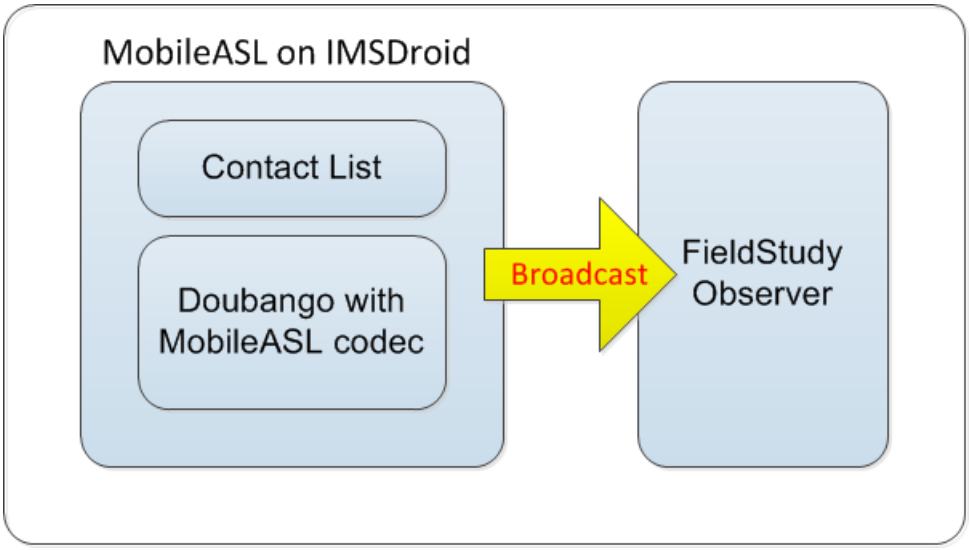


Figure 6.1: Software Framework for MobileASL Android.

IMSDroid. The communication between IMSDroid and the field study observer uses the internal broadcast messaging feature available for Android.

The contact list is used to keep track of who is online/offline in a common group, which includes all users in a field study. The contact list server is periodically informed of the offline/online status of users.

Figure 6.2 shows the system framework for MobileASL Android. Besides the contact list server, there is another server to maintain the SIP protocol for the video call, which is located in the public domain. I chose "sip2sip.info" or "iptel.org" for this SIP server.

6.2 MobileASL on IMSDroid

IMSDroid includes key components needed to interface with the camera and to project the video to the mobile device screen. It makes use of doubango's features for audio/video codec and communication. Figure 6.3 shows the internal architecture we built on top of IMSDroid as part of MobileASL, which allows a user to adjust encoding parameters, bit rates, ROI encoding, and variable frame rate. In addition, IMSDroid is customized for sign language video communication, which are video layout and size on the screen, menu, and texts.

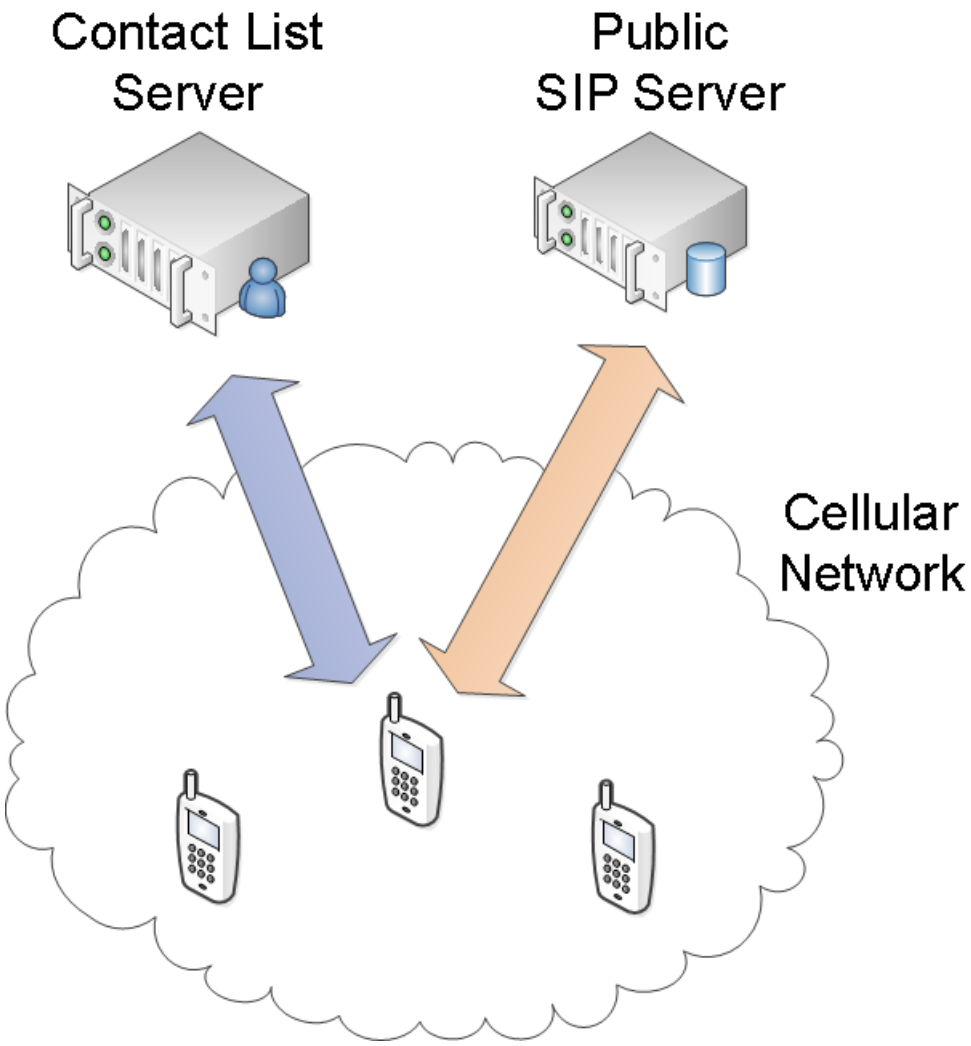


Figure 6.2: System Framework for MobileASL Android.

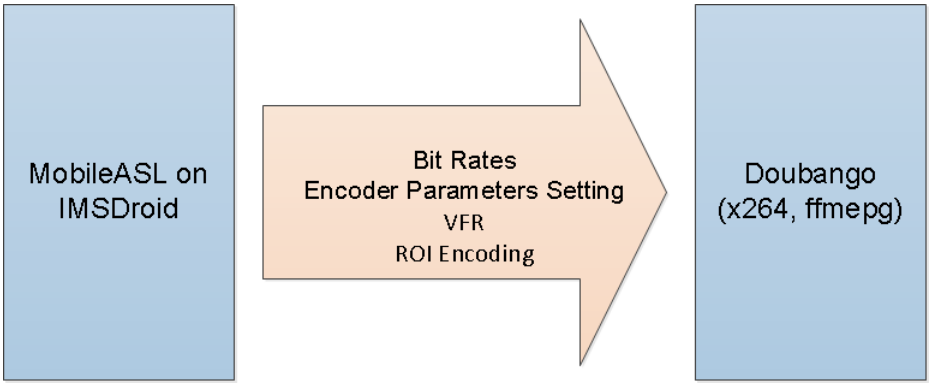


Figure 6.3: MobileASL on IMSDroid to control the settings.

6.3 Contact List

The main purpose of the contact list is to allow users to see who else is currently signed into MobileASL, so they can see who is available for calling. The contact list server periodically receives status information from all online MobileASL phones and sends it back to all the active phones in the entire contact list. Figure 6.4 is a screen shot of the contact list.

6.4 Experience Sampling

Experience sampling is used to collect usage data directly from participants. Different questions can be asked depending on the call usage, time, location, and so on. A multiple-choice question pops up to query participants. Figure 6.5 is a screen shot of an experience sampling question on the Android platform.

6.5 Discussion

Even if we use an existing software platform for MobileASL, there are critical uncontrollable issues that affect performance. One is the SIP protocol which did not work well. Sometimes it took a lot of time to be logged into the SIP server and other times, it did not log into the SIP server at all.

Furthermore, the user interface needs to be enhanced. The first example is an automatic logging into the SIP server when MobileASL is started. The second is to develop the private

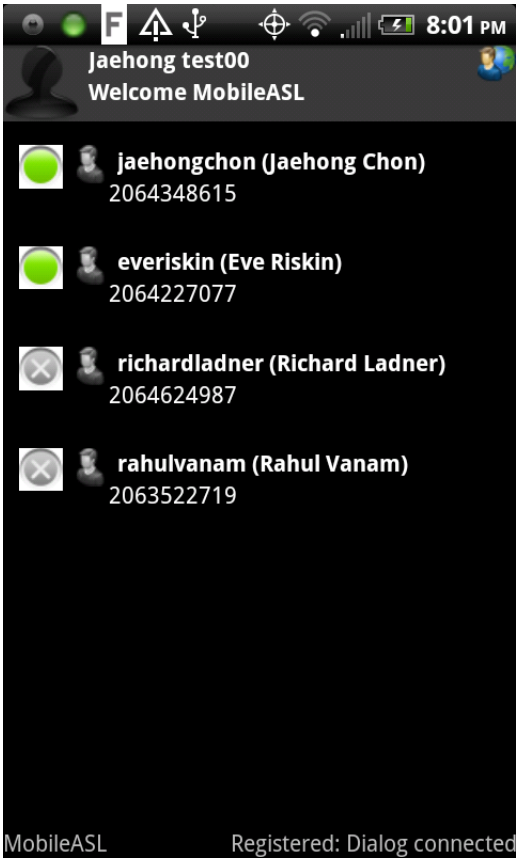


Figure 6.4: Screen shot of the contact list in MobileASL Android.

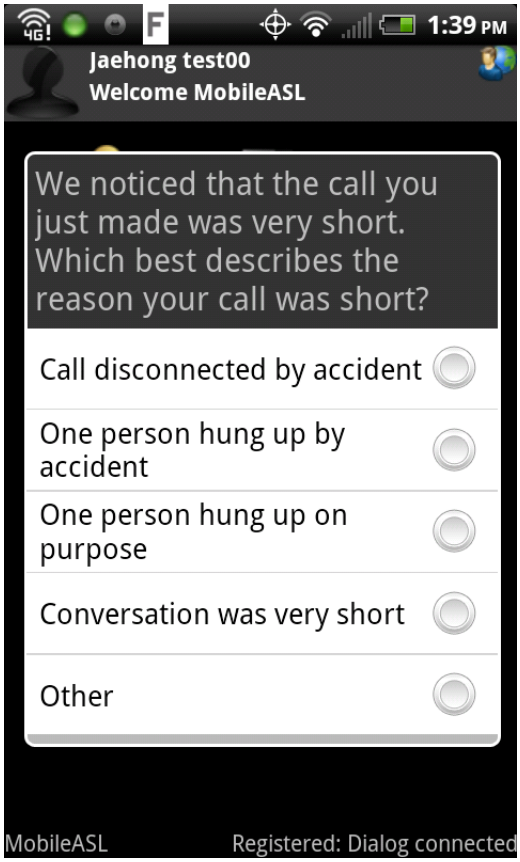


Figure 6.5: A screen shot of an experience sampling question for MobileASL Android.

contact list per user so that they can easily access to their closest contacts.

Finally, the automatic control is needed for the settings such as bit rates and encoding parameter settings according to the available bandwidth over the network.

Chapter 7

FIELD STUDIES

In this chapter, I describe the field studies run in 2010 and 2011 that evaluated the usability of the MobileASL system. The first field study evaluated MobileASL for the HTC TyTN II on Windows Mobile, and the second field study evaluated MobileASL for the HTC EVO 4G on Android. In both of these field studies, phones equipped with MobileASL were distributed to Deaf high school and college students recruited from the Summer Academy for Advancing Deaf & Hard of Hearing (DHH) in Computing at University of Washington.

Since 57.1 % of people ages 18-24 years old have cell phones and no landline [17] and major mobile video applications, such as Skype and FaceTime, support only Wi-Fi, I expect that users will actively take advantage of MobileASL (since it can communicate over cell phone networks) for a variety of cases. For example, a user might call their friend when they are downtown and show where they are so that they can find each other. Or, they might be able to call an interpreter in case of an emergency while they are away from their home video phone. The field studies showed that users used MobileASL for many different situations.

7.1 MobileASL on Windows Mobile

MobileASL was evaluated in a three-week field study during the summer of 2010 on HTC TyTN-II phones running over the AT&T 3G network. We asked participants about their usage behavior through questions shown on-device and collected data such as the number of total bytes sent and received to observe how MobileASL was used.

Figure 7.1 shows two field study participants talking to each other using MobileASL. Although we found that use of MobileASL was limited by its short battery life and that participants disliked the form factor of the HTC TyTN-II, participants made full use of the mobility provided by MobileASL by using it in places like buses, restaurants, and shopping

Figure 7.1: Two students talking to each other using MobileASL during Summer 2010 [41].

areas.

7.1.1 Procedure for the 2010 Field Study

Since the current cellular network in the U.S. is not optimized for two-way real-time video communication, I set MobileASL to use 30kbps for video encoding, which is high enough to encode the video for 96x80 pixel resolution. In order to transmit intelligible ASL, I enabled region-of-interest encoding around the face and hands (see chapter 3.5 for details), which are the most important areas for intelligible sign language communication.

We developed two different methods for collecting usage data directly from the MobileASL phones. The first was to use on-device, multiple choice questions to collect subjective data from study participants, and the second was to log objective usage data in the background as the participant interacted with MobileASL (see Table 7.1).

In addition, we conducted interviews with and distributed pre- and post-questionnaires to the participants. The interviews were particularly useful because we got very rich feedback from the participants.

Table 7.1: Subjective and objective data collected from the field study.

Subjective Data	Purpose of the Call
	Quality of the Call
	Whom They Call
	Reason to Hang Up the Call
	Where They Are
Objective Data	Battery Usage
	Bytes Sent and Received
	MobileASL Phone Calls
	Changes in IP Address
	MobileASL Running Time
	Number of Packet Loss

7.1.2 Results

Calls were infrequent, with each participant making 0 to 2 calls a day. When participants did make calls, they tended to be short, but the duration varied widely. From this, it seems that participants were making calls for short information gathering when they needed it (Average = 106.16 secs, Standard Deviation = 158.66 secs) [41]. Participants also mentioned that they preferred to use MobileASL over existing wired video conferencing technologies or texting.

We found that when asked about their current location while using MobileASL, users frequently responded with "public place or business" and "other" [41]. From this, it seems that participants were taking advantage of the mobility enabled by MobileASL.

There was also negative feedback from participants. Participants stated that they found the battery life of MobileASL to be too short, and that they disliked the form factor of the HTC TyTN-II device; they found it too big and wanted a touch-screen interface rather than an interface designed for a stylus.

7.2 MobileASL on Android

From the field study in 2010, we learned that participants were unable to call their friends or family who cannot speak sign language because MobileASL lacked some features such as texting. Furthermore, in order to make a variety of calls for sign language video communication, we need a desktop client that can talk with MobileASL.

Another possibility for expanding the study is to allow MobileASL to connect to video relay services, which will enable Deaf users to call hearing people even while away from their home video phones. For example, a user could make an appointment with a doctor while on-the-go.

With these possibilities in mind, I ported MobileASL to Android; the details of this port is explained in Chapter 6.

7.2.1 Procedure for the 2011 Field Study

Like in the field study in 2010, we recruited students attending the Summer Academy for Advancing DHH in Computing at University of Washington as participants for our 2011 field study. We switched from using the HTC TyTN-II to the HTC EVO 4G, which runs on Android (see Figure 5.7).

MobileASL was improved in many ways for this field study. Because we were now using HTC EVO 4Gs, we were able to take advantage of the Android environment; the finger-friendly touch screen interface it provides; and the slimmer, lighter form factor of the actual device. There is also a desktop client, boghe [5] using doubango's features, which can talk with MobileASL. Overall, the phones themselves were lighter and faster, and had a bigger screen than the phone we used in 2010. We also utilized the SIP protocol so that we could use public SIP servers, like "sip2sip.info" and "iptel.org," for better interoperability with different platforms. The last improvement was in the choice to use a faster cellular network (Sprint 4G) than in the last study. Because we were using this faster network and a phone with a larger screen, I set the video encoder to 50kbps.

On top of the baseline software platform, IMSDroid, we again made use of on-device questions and logging in the background to observe how MobileASL was used.

Table 7.2: Major differences between mobile phones we used in the field studies.

item	HTC TyTN-II	HTC EVO 4G
Release	September 2007	June 2010
OS	Windows Mobile 6	Android 2.3
CPU	400MHz	1GHz
Screen	320x240	800x480
Network	AT&T 3G	Sprint 4G

7.2.2 Results

Although this field study was longer than last time and utilized "better" technologies, the actual results were not very informative. Participants made very few calls, except during the first couple of days. The main reason for this was because the new MobileASL and the Sprint 3G/4G network were not stable.

In general, the Sprint 4G network is faster than the AT&T 3G network with respect to downlink and uplink speed. But Sprint 4G is currently deployed in limited areas, so participants could barely access this network in Seattle and at the University of Washington. Furthermore, the available bandwidth fluctuated widely and variation in round trip delay was huge (see Figure 5.14) so the Sprint 3G network was not useable for two-way real-time video communication. A network must be fast, easily available to users, and stable enough for video communication.

In addition, the SIP protocol did not work well because of network problems and the instability of IMSDroid. Sometimes it took a lot of time to be logged into the SIP server. It also crashed frequently when the call was terminated.

7.3 Discussion

Mobile technologies have significantly improved over the recent years. We ourselves observed this fact through our two field studies. The difference between the two mobile platforms we used, the HTC TyTN II and the HTC EVO 4G, was huge (see Table 7.2).

The applications on mobile devices have also highly improved due to the new technologies. Therefore, I suggest taking advantage of these new applications for future field studies by using existing solutions (such as the ZVRS applications for mobile and desktop platforms) so that we can avoid engineering issues that were the main barriers to our field studies. Using a stable system is most important.

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 Future Work

There are several improvements that could be made to further improve video communication over mobile networks. The intelligibility model with frame rate and bit rates for sign language communication could transcend changes in the technology because I do not expect human behavior during video calls is to change. This model has many potentials to be improved. For example, another important factor in the mobile device is the battery life. The more bits used, the more power is consumed. Therefore, this model could be extended to account for battery consumption.

Since the essential concept to reduce the amount of data in video compression is motion estimation and compensation with previous frames, the video codec is highly vulnerable to packet loss. The intra block refresh to recover the quality could also transcend changes in the technology.

8.1.1 User study of intelligibility

The intelligibility metric was driven by mapping between objective numbers (frame rate and PSNR) and MOS. Each mapping for frame rate and the PSNR was verified in previous work. However, there was no effort to adjust the total intelligibility metric to fit human perception. Therefore, it would be useful to have a user study of intelligibility.

8.1.2 Dynamic Video Quality

Although the overall speed of 4G networks is faster than that of 3G networks, the actual speed of a 4G network at a given time and location may vary dramatically. Therefore, it may be useful to provide dynamic control over video resolution, bit rate, and frame rate so that video calls can adapt to the amount of bandwidth currently available.

8.1.3 Adjusting MobileASL

MobileASL sets the encoding parameters for H.264, bit rate, and video resolution at the moment a video call is established; these parameters cannot be controlled adaptively. It would be interesting to be able to choose the best setting depending on system resources such as the available bandwidth, encoding quality, and battery. My intelligibility metric can be further evolved to achieve this goal.

8.1.4 ROI Improvement

MobileASL system uses ROI-based video encoding to ensure better quality around the face and hands. However, the simple algorithm currently being used the detect the face and hands can vary in accuracy, because it is heavily affected by the lighting conditions of the camera. It can be enhanced by tracking the region after detecting the initial region with a more accurate algorithm.

8.1.5 Field Study

From the two field studies, we learned that participants prefer phones with slim form factors and long battery life. Furthermore, we realized the importance of having a stable system. However, developing a system like MobileASL that works over cellular networks requires many different technologies such as video compression, network protocols, and a decent user interface. As I explained in chapter 7, using an existing application, such as ZVRS, rather than starting from scratch can increase the stability of the MobileASL system. We could determine pre-defined settings we want to use with the existing application for future studies. As far as I know, there are few, if any, studies observing how Deaf users utilize video calls on cellular networks, for example, video call vs. texting.

8.2 Conclusion

In this research, I described a system that allows real-time sign language video communication on mobile phones over current cellular networks. I built the MobileASL system on Windows Mobile to give Deaf users equal access. I also developed a method for recovering

video quality by using feedback-based refreshment when packet loss occurs. Furthermore, MobileASL was ported to Android so that it could be used on increasingly popular Android devices. I then evaluated MobileASL system in two field studies in which participants used MobileASL in their lives. Results showed that participants took advantage of the mobility provided by MobileASL, but that use was limited by low battery life.

In order to improve overall user satisfaction with MobileASL's video communication, I propose a metric for intelligibility based on image quality, frame rate, and video resolution. In addition, I show the network status for the latest cellular systems. Therefore, my intelligibility metric can be further extended to adjust the video encoder, in real time, depending on current system and network conditions. My thesis brings us closer towards achieving MobileASL's goal of making full use of mobile networks for real-time ASL conversations.

BIBLIOGRAPHY

[1] ARMv6-M Architecture Reference Manual. http://infocenter.arm.com/help/index.jsp.

[2] Doubango. http://www.doubango.org/.

[3] H.264/AVC JM Reference Software. http://iphome.hhi.de/suehring/tml/.

[4] IMSDroid. http://code.google.com/p/imsdroid/.

[5] IMS/RCS client for Windows. http://code.google.com/p/boghe/.

[6] iVisit: Free Video Conferencing, Web Conferencing, and Collaboration. http://www.ivisit.com/.

[7] Methods for objective and subjective assessment of quality. ITU-T Recommendation P.800.

[8] MobileASL Study Videos. http://mobileasl.cs.washington.edu/downloads/studyVideos/.

[9] MPEG Test Media. http://media.xiph.org/video/derf/.

[10] Ookla Net Metrics. http://www.ookla.com.

[11] Siprelay. http://www.siprelay.com/.

[12] Summer Academy for Advancing Deaf & Hard of Hearing in Computing. http://www.washington.edu/accesscomputing/dhh/academy/.

[13] the fourth generation (4g). http://en.wikipedia.org/wiki/4G.

[14] VZO Mobile Video Phone. http://mobile.vzochat.com/en/.

[15] x264 - a free H.264/AVC encoder. http://www.videolan.org/developers/x264.html.

[16] 4th Annual MSU MPEG-4 AVC/H.264 Video Codec Comparison, 2007. http://www.compression.ru/video/codec_comparison/mpeg-4_avc_h264_2007_en.html.

[17] Telecom milestone, 2007. More Cell Phone-Only Than Landline-Only Households.

[18] Google's Android becomes the world's leading smart phone platform, January 2011. Canalys.

[19] Data Speed Showdown: Sprint 4G vs T-Mobile HSPA+, June 2010. `http://www.intomobile.com/2010/06/04/data-speed-showdown-sprint-4g-vs-t-mobile-hspa/`.

[20] Google and Apple Gain Ground in Smartphone Market, May 2011. comScore MobiLens.

[21] 3GPP, Siemens. Software simulator for MBMS streaming over UTRAN and GERAN. *document for proposal, TSG System Aspects Working Group4#36, Tdoc S4-050560*, Sep 2005.

[22] A. Cavender, R. Vanam, D. Barney, R. E. Ladner, E. A. Riskin. MobileASL: Intelligibility of sign language video over mobile phones. In *Disability and Rehabilitation: Assistive Technology*. London: Taylor and Francis, Jun 2007.

[23] A. Khan, L. Sun, E. Ifeachor, J. Fajardo, F. Liberal, H. Koumaras. Video Quality Prediction Models Based on Video Content Dynamics for H.264 Video over UMTS Networks. *International Journal of Digital Multimedia Broadcasting*, Feb 2010.

[24] B. Aswathappa, K. R. Rao. Rate-Distortion Optimization using Structural Information in H.264 Strictly Intra-frame Encoder. In *42nd South Eastern Symposium on System Theory*, Mar 2010.

[25] B. F. Johnson, J. K. Caird. The effect of frame rate and video information redundancy on the perceptual learning of American Sign Language gestures. In *CHI '96: Conference companion on Human factors in computing systems*, pages 121–122, New York, NY, USA, 1996. ACM Press.

[26] B. Ford, P. Srisuresh, D. Kegel. Peer-to-Peer Communication Across Network Address Translators. In *Proceedings of the 2005 USENIX Annual Technical Conference*, Apr 2005.

[27] B. Yan, H. Gharavi. Efficient Error Concealment for the Whole-Frame Loss based on H.264/AVC. In *Proceeding of IEEE International Conference on Image Processing*, 2008.

[28] B. Yan, H. Gharavi. A Hybrid Frame Concealment Algorithm for H.264/AVC. *IEEE Transactions on Image Processing*, 19, Jan 2010.

[29] C. Zhu, Y. Wang, M. M. Hannuksela, H. Li. Error Resilient Video Coding Using Redundant Pictures. *IEEE Transactions on Circuit and Systems for Video Technology*, 19, Jan 2009.

[30] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2014, Feb 2010.

[31] F. M. Ciaramello, S. S. Hemami. 'Can you see me now?' An Objective Metric for Predicting Intelligibility of Compressed American Sign Language Video. In *Proceeding of Human Vision and Electronic Imaging(HVEI)*, Jan 2007.

[32] F. M. Ciaramello, S. S. Hemami. Real-Time Face and Hand detection for Videoconferencing on a Mobile Device. In *4th International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM)*, Jan 2009.

[33] F. Yang, X. Wang, Y. Chang, S. Wan. A No-Reference Video Quality Assessment Method Based on Digital Watermark. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC)*, 2003.

[34] R. A. Foulds. Biomechanical and perceptual constraints on the bandwidth requirements of sign language. In *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, volume 12, pages Vol I: 65–72, Mar 2004.

[35] Y. Cohen M. Pavel G. Sperling, M. Landy. Intelligible encoding of ASL image sequences at extremely low information rates. *Computer vision, graphics, and image processing*, 31(3):335–391, Sep 1985.

[36] Informa UK Ltd. Mobile Internet Traffic: Analysing Global Usage Trends, 2010.

[37] I.T.S. Sector. Draft application profile: Sign language and lip reading real time conversation usage of low bit rate video communication, 1998.

[38] J. Chon, N. Cherniavsky, E. Riskin, R. Ladner. Enabling Access Through Real-Time Sign Language Communication Over Cell Phones. In *43rd Annual Asilomar Conference on Signals, Systems, and Computers*, Nov 2009.

[39] J. Devadoss, V. Singh, J. Ott, C. Liu, Y. Wang, I. Curcio. Evaluation of Error Resilience Mechanisms for 3G Conversational Video. In *10th IEEE International Symposium on Multimedia (ISM)*, Dec 2008.

[40] J. Han, Y. Kim, J. Jeong, J. Shin. Video Quality Estimation for Packet Loss Based on No-Reference Method. In *International Conference on Advanced Communication Technology (ICACT)*, Feb 2010.

[41] J. Kim, J. J. Tran, T. W. Johnson, R. Ladner, E. Riskin, and J. O. Wobbrock. Effect of MobileASL on Communication Among Deaf Users. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI '11)*, May 2011.

[42] J. Klaue, B. Rathke, A. Wolisz. EvalVid - A Framework for Video Transmission and Quality Evaluation. In *proceeding of the 13th international conference on Modeling Techniques and Tools for Computer Performance Evaulation*, Sep. 2004.

[43] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG JVTG050r1. ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC: Draft ITU-T recommendation and final draft international standard of joint video specification, 2003.

[44] K. Fitchard. AT&T Cellular Network Going All IP, Oct 2007.

[45] K. Lee, T. Kim, B. Seo, J. Suh. Fast Reference Frame Selection Algorithm for H.264/AVC Based on Reference Frame Map. 2010.

[46] K. Liao, J. Yang, M. Sun. Rate-Distortion Cost Estimation for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 20, Jan 2010.

[47] R. Vanam L. Merritt. Improved rate control and motion estimation for H.264 encoder. In *Proceedings of ICIP*, volume 5, pages 309–312, 2007.

[48] Visual Communications Lab. `http://foulard.ece.cornell.edu/index.php`.

[49] M. Chen, C. Chen, M. Chi. Temporal Error Concealment Algorithm by Recursive Block-Matching Principle. *IEEE Transactions on Circuits and Systems for Video Technology*, 15, Nov 2005.

[50] MobileASL. `http://mobileasl.cs.washington.edu`.

[51] N. Tizon, B. Pesquet-Popescu, M. Cagnazzo. Adaptive video streaming with long term feedbacks. In *16th IEEE International Conference on Image Processing (ICIP)*, 2009.

[52] Q. Peng, T. Yang, C. Zhu. Block-Based Temporal Error Concealment for Video Packet Using Motion Vector Extrapolation. In *IEEE Conference on Communications, Circuits and Systems and West Sino Expositions*, June 2002.

[53] R. Vanam, E. Riskin, and R. Ladner. H.264/MPEG-4 AVC encoder parameter selection algorithms for complexity distortion tradeoff. In *Proceedings of the IEEE Data Compression Conference*, March 2009.

[54] R. Vanam, E. Riskin, S. Hemami, and R. Ladner. Distortion-Complexity Optimization of the H.264/MPEG-4 AVC Encoder using the GBFOS Algorithm. In *Proceedings of the IEEE Data Compression Conference*, March 2007.

[55] R. Vanam, E. Riskin, S. Hemami, R. Ladner. Distortion-Complexity Optimization of the H.264/MPEG-4 AVC Encoder using the GBFOS Algorithm. In *Proceedings of the IEEE Data Compression Conference*, Mar 2007.

[56] R. Vanam, F. Ciaramello, E. Riskin, S. Hemami and R. Ladner. Joint Rate-Intelligibility-Complexity Optimization of an H.264 Video Encoder for American Sign Language. In *Western New York Image Processing Workshop (WNYIP)*, September 2009.

[57] RBC Capital Markets. Wireless Industry - Sizing the Golbal Smartphone Market.

[58] RFC3022. Traditional IP Network Address Translator (Traditional NAT), Jan 2001.

[59] S. L. Phung, A. Bouzerdoum, D. Chai. Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 27:148–154, Jan 2005.

[60] S. Wan, E. Izquierdo. Rate-Distortion Optimized Motion Compensated Prediction for Packet Loss Resilient Video Coding. *IEEE Transactions on Image Processing*, 16, May 2007.

[61] S. Whittle. *Maintaining Intelligibility of ASL Video in the Presence of Data Loss*. University of Washington, Computer Science and Engineeing, Senior Thesis, 2008.

[62] T. Stockhammer, D. Kontopodis, T. Wieg. Rate-Distortion Optimization for JVT/H.26L Video Coding in Packet Loss Environment. In *Proceedings of the International Packet Video Workshop*, Apr 2002.

[63] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13, Jul 2003.

[64] V. Singh, J. Ott, I.D.D Curcio. Rate Adaptation for Conversational 3G Video. In *IEEE INFOCOM Workshops*, 2009.

[65] X Liu, K Sohn, LT Yang, W Zhu. A Pervasive Temporal Error Concealment Algorithm for H. 264/AVC Decoder.

[66] Y. Chen, K. Yu, J. Li, S. Li. An Error Concealment Algorithm for Entire Frame Loss in Video Transmission. In *Proceeding of IEEE Picture Coding Symposium (PCS)*, 2004.

[67] Y. J.Liang, J. G. Apostolopoulos, B. Girod. Analysis of packet loss for compressed video: Does burst-length matter? In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Apr 2003.

[68] Y. Lu, Y. Zhao, F. Kuipers, P. V. Mieghem. Measurement Study of Multi-party Video Conferencing. In *IFIP International Conferences on Networking*, May 2010.

[69] Y. Ou, Z. Ma, Y. Wang. Modeling the Impact of Frame Rate and Quantization Step Sizes and their Temporal Variations on Perceptual Video Quality: a Review of Recent Works. In *IEEE 44th Annual Conference on Information Sciences and Systems (CISS)*, 2010.

[70] Y. Wang, S. Wenger, J. Wen, A. K. Katsaggelos. Error Resilient Video Coding Techniques. *IEEE Signal Processing Magazine*, 17, July 2000.

[71] Y. Xu, Y. Zhou. Adaptive temporal error concealment scheme for H.264/AVC video decoder. *IEEE Transactions on Consumer Electronics*, 54, Nov 2008.

[72] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13, Apr 2004.