Activity Analysis of Sign Language Video for Mobile Telecommunication

Neva Cherniavsky

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

Doctor of Philosophy

University of Washington

2009

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

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

Neva Cherniavsky

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:

_____

Richard E. Ladner

_____

Eve A. Riskin

Reading Committee:

_____

Richard E. Ladner

_____

Eve A. Riskin

_____

Jacob O. Wobbrock

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, or to the author.

Signature_____

Date_____

University of Washington

Abstract

Activity Analysis of Sign Language Video for Mobile Telecommunication

Neva Cherniavsky

Co-Chairs of the Supervisory Committee:
Professor Richard E. Ladner
Computer Science and Engineering

Professor Eve A. Riskin
Electrical Engineering

The goal of enabling access for the Deaf to the current U.S. mobile phone network by compressing and transmitting sign language video gives rise to challenging research questions. Encoding and transmission of real-time video over mobile phones is a power-intensive task that can quickly drain the battery, rendering the phone useless. Properties of conversational sign language can help save power and bits: namely, lower frame rates are possible when one person is not signing due to turn-taking, and the grammar of sign language is found primarily in the face. Thus the focus can be on the important parts of the video, saving resources without degrading intelligibility.

My thesis is that it is possible to compress and transmit intelligible video in real-time on an off-the-shelf mobile phone by adjusting the frame rate based on the activity and by coding the skin at a higher bit rate than the rest of the video. In this dissertation, I describe my algorithms for determining in real-time the activity in the video and encoding a dynamic skin-based region-of-interest. I use features available "for free" from the encoder, and implement my techniques on an off-the-shelf mobile phone. I evaluate my sign language sensitive methods in a user study, with positive results. The algorithms can save considerable resources without sacrificing intelligibility, helping make real-time video communication on mobile phones both feasible and practical.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ACTIVITY ANALYSIS OF VIDEO: classification of video into different categories based on the activity recognized in the video

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

BANDWIDTH: the data capacity of a communication channel, measured in bits per second (bps) or kilobits per second (kbps)

CENTER OF GRAVITY (COG): the average location of the weighted center of an object

CHROMINANCE: the color component of an image

DEZ: the part of sign corresponding to hand shape in ASL

FINGER SPELLING: sign language in which each individual letter is spelled

FOVEAL VISION: vision within two degrees of the center of the visual field

FRAME: a single video image

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

HA: the part of sign corresponding to the position of the hands relative to each other in British Sign Language

HAND SHAPE: the position the hand is held while making a sign

HIDDEN MARKOV MODEL (HMM): a statistical model of a temporal system often used in pattern recognition

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

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

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

LUMINANCE: the brightest component of an image

MACROBLOCK: a 16×16 square area of pixels

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

ORI: the part of sign corresponding to palm orientation in ASL

PERIPHERAL VISION: vision outside the center of the visual field

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

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

REPAIR REQUEST: a request for repetition

SIG: the part of sign corresponding to movement in ASL

SUPPORT VECTOR MACHINE (SVM): a machine learning classification algorithm

TAB: the part of sign corresponding to location in ASL

TELETYPEWRITER (TTY): a device that allows users to type messages in real-time over the phone lines

VARIABLE FRAME RATE (VFR): a frame rate that varies based on the activity in the video

X264: an open source implementation of H.264

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisors, Richard and Eve. Both were enormously helpful during my graduate studies. Richard is an excellent mentor who constantly pushed me to be productive and work well, while also bolstering my confidence as an independent researcher. Eve is an enormously energetic and enthusiastic scientist; we had a great many productive conversations, and her advice in finding a job, managing family, and dealing with personal crisis made my graduation possible. I would also like to thank Jake Wobbrock, who I only started working with a year ago, but who has taught me a great deal about human-centered research.

My colleagues Jaehong Chon and Anna Cavender helped with some of the research in this dissertation, and I throughly enjoyed working with them both. I am also grateful to the members of the MobileASL project team, including Rahul Varnum, Frank Ciaramello, Dane Barney, and Loren Merritt; discussions with them informed my approach to problems and kept me on the right track.

Finally, I would like to thank my family and friends. My parents have always been very supportive of my graduate education; my mother is my first and best editor, and my father always let me know that he believed in me and was proud of me. Visiting my brother, his wife, and my niece in San Jose was my favorite escape from the rigors of study. My friends kept me sane during good times and bad. I will miss them all terribly when I leave Seattle, but most especially Liz Korb, Dan Halperin, Schuyler Charf, Jess Williams, and Arnie Larson.

DEDICATION

To my parents, John and Ellen

Chapter 1

INTRODUCTION

Mobile phone use has skyrocketed in recent years, with more than 2.68 billion subscribers worldwide as of September 2007 [53]. Mobile technology has affected nearly every sector of society [64]. On the most basic level, staying in touch is easier than ever before. People as diverse as plumbers, CEOs, real estate agents, and teenagers all take advantage of mobile phones, to talk to more people, consult from any location, and make last minute arrangements. In the United States, nearly one-fifth of homes have no land line [40]. Bans on phone use while driving or in the classroom are common. Even elementary school children can take advantage of the new technology; 31% of parents of 10-11 year-olds report buying phones for their children [57].

Deaf[1] people have embraced mobile technologies as an invaluable way to enable communication. The preferred language of Deaf people in the United States is American Sign Language (ASL). Sign languages are recognized linguistically as natural languages, with the accompanying complexity in grammar, syntax, and vocabulary [103]. Instead of conversing orally, signers use facial expressions and gestures to communicate. Sign language is not pantomime and it is not necessarily based on the oral language of its community. For example, ASL is much closer to French Sign Language than to British Sign Language, because Laurent Clerc, a deaf French educator, co-founded the first educational institute for the Deaf in the United States [33]. While accurate numbers are hard to come by [69], as of 1972 there were at least 500,000 people that signed at home regardless of hearing status [97]. Since then, the numbers have probably increased; ASL is now the fourth most taught "foreign" language in higher education, accounting for 5% of language enrollment [32].

Previously, the telephone substitute for Deaf users was the teletypewriter (TTY), invented in 1964. The original device consisted of a standard teletype machine (in use since

[1]Capitalized Deaf refers to members of the signing Deaf community, whereas deaf is a medical term.

the 1800s for telegrams), coupled with an acoustic modem that allowed users to type messages back and forth in real-time over the phone lines. In the United States, federal law mandates accessibility to the telephone network through free TTY devices and TTY numbers for government offices. The devices became smaller and more portable over the years, and by the 1990s a Deaf user could communicate with a hearing person through a TTY relay service.

However, the development of video phones and Internet-based video communication essentially made the TTY obsolete. Video phones are dedicated devices that work over the broadband Internet. It is also possible to forgo the specialized device and instead use a web camera attached to a computer connected to the Internet. Skype, a program that enables voice phone calls over the Internet, has a video chat component. Free software is widely available, and video service is built into services such as Google chat and Windows Live messenger. Video phones also enable Deaf-hearing communication, through video relay service, in which the Deaf user signs over the video phone to an interpreter, who in turn voices the communication over a regular phone to a hearing user. Since 2002, the federal government in the United States has subsidized video relay services. With video phones, Deaf people finally have the equivalent communication device to a land line.

The explosion of mobile technologies has not left Deaf people behind; on the contrary, many regularly use mobile text devices such as Blackberries and Sidekicks. Numerous studies detail how text messaging has changed Deaf culture [87, 42]. In a prominent recent example at Gallaudet University, Deaf students used mobile devices to organize sit-ins and rallies, and ultimately to shut down the campus, in order to protest the appointment of the president [44]. However, text messaging is much slower than signing. Signing has the same communication rate as spoken language of 120-200 words per minute (wpm) versus 5-25 wpm for text messaging [54]. Furthermore, text messaging forces Deaf users to communicate in English as opposed to ASL. Text messaging is thus the mobile equivalent of the TTY for land lines; it allows access to the mobile network, but it is a lesser form of the technology available to hearing people. Currently, there are no video mobile phones on the market in the U.S. that allow for real-time two-way video conversation.

Figure 1.1: MobileASL: sign language video over mobile phones.

1.1 MobileASL

Our MobileASL project aims to expand accessibility for Deaf people by efficiently compressing sign language video to enable mobile phone communication (see Figure 1.1). The project envisions users capturing and receiving video on a typical mobile phone. The users wear no special clothing or equipment, since this would make the technology less accessible.

Work on the project began by conducting a focus group study on mobile video phone technology and a user study on the intelligibility effects of video compression techniques on sign language video [12]. The focus group discussed how, when, where, and for what purposes Deaf users would employ mobile video phones. Features from these conversations were incorporated into the design of MobileASL.

The user study examined two approaches for better video compression. In previous eyetracking studies, researchers had found that over 95% of the gaze points fell within 2 degrees visual angle of the signer's face. Inspired by this work, members of the project team conducted a study into the intelligibility effects of encoding the area around the face at a higher bit rate than the rest of the video. They also measured intelligibility effects at different frame rates and different bit rates. Users found higher bit rates more understandable, as expected, but preferred a moderate adjustment of the area around the signer's face. Members of the team then focused on the appropriate adjustment of encoding parameters [112, 13]; creating an objective measure for intelligibility [18]; and balancing

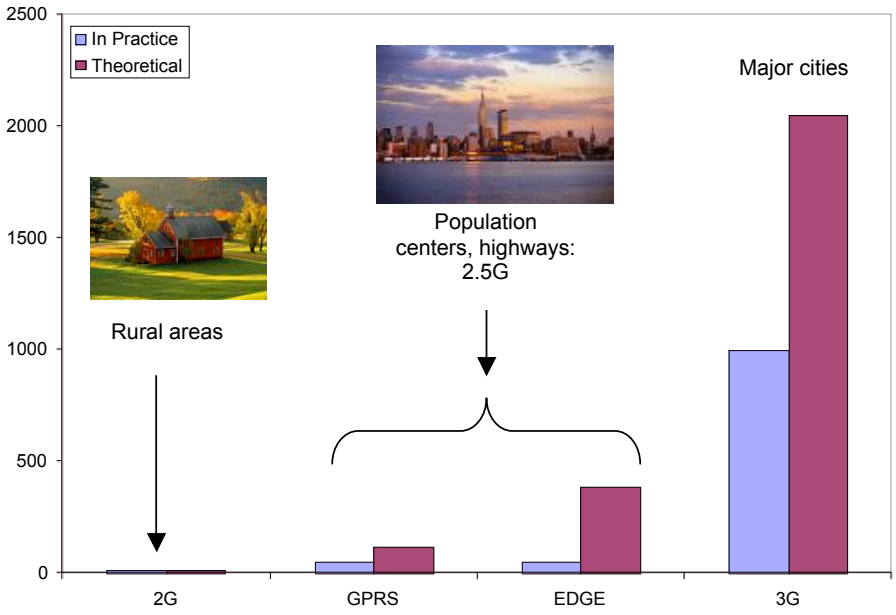


Figure 1.2: Mobile telephony maximum data rates for different standards in kilobits per second [77].

intelligibility and complexity [19].

The central goal of the project is real-time sign language video communication on off-the-shelf mobile phones between users that wear no special clothing or equipment. The challenges are three-fold:

- **Low bandwidth:** In the United States, the majority of the mobile phone network uses GPRS [38], which can support bandwidth up to around 30-50 kbps [36] (see Figure 1.2). Japan and Europe use the higher bandwidth 3G [52] network. While mobile sign language communication is already available there, the quality is poor, the videos are jerky, and there is significant delay. Figure 1.3 shows AT&T's coverage of the United States with the different mobile telephony standards. AT&T is the largest provider of 3G technology and yet its coverage is limited to only a few major

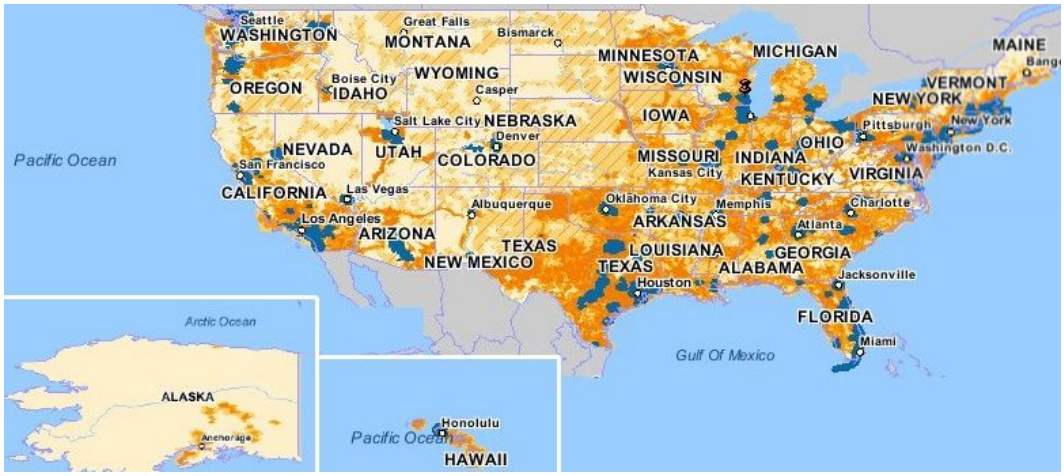


Figure 1.3: AT&T's coverage of the United States, July 2008. Blue is 3G; dark and light orange are EDGE and GPRS; and banded orange is partner GPRS. The rest is 2G or no coverage.

cities. Since even GPRS is not available nationwide, it will be a long time until there is 3G service coast to coast. Moreover, from the perspective of the network, many users transmitting video places a high burden overall on the system. Often phone companies pass this expense on to users by billing them for the amount of data they transmit and receive.

- **Low processing speed:** Even the best mobile phones available on the market, running an operating system like Windows Mobile and able to execute many different software programs, have very limited processing power. Our current MobileASL phones (HTC TyTN II) have a 400 MHz processor, versus 2.5 GHz or higher for a typical desktop computer. The processor must be able to encode and transmit the video in close to real-time; otherwise, a delay is introduced that negatively affects intelligibility.

- **Limited battery life:** A major side effect of the intensive processing involved in video compression on mobile phones is battery drain. Insufficient battery life of a mobile device limits its usefulness if a conversation cannot last for more than a few minutes. In an evaluation of the power consumption of a handheld computer, Viredaz and Wallach

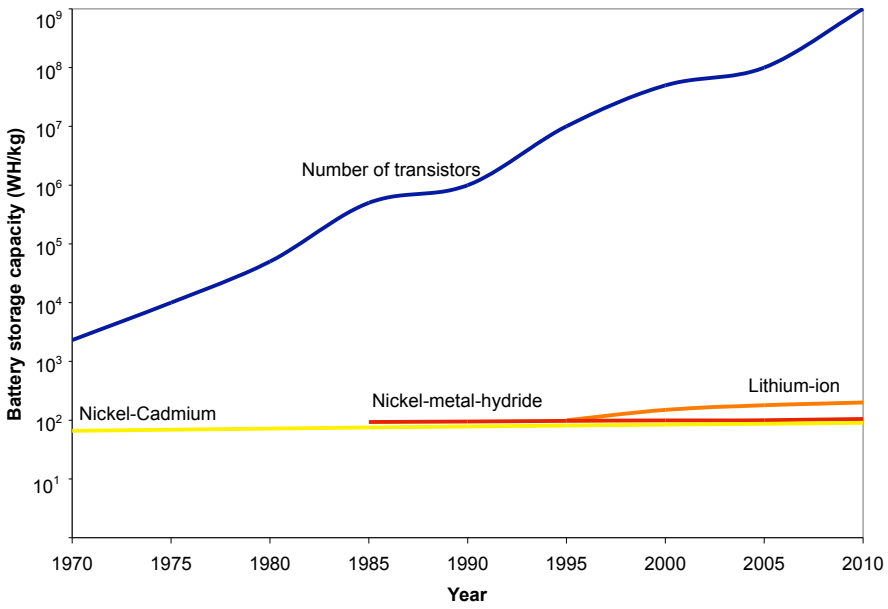


Figure 1.4: Growth in rechargeable-battery storage capacity (measured in watt hours per kilogram) versus number of transistors, on a log scale [26].

found that decoding and playing a video was so computationally expensive that it reduced the battery lifetime from 40 hours to 2.5 hours [113]. For a sign language conversation, not only do we want to play video, but we also want to capture, encode, transmit, receive and decode video, all in real-time. Power is in some ways the most intractable problem; while bandwidth and processing speed can be expected to grow over the next few years, battery storage capacity has not kept up with Moore's law (see Figure 1.4).

In the same way that unique characteristics of speech enable better compression than standard audio [11], sign language has distinct features that should enable better compression than is typical for video. One aspect of sign language video is that it is conversational;

times when a user is signing are more important than times when they are not. Another aspect is touched upon by the eye-tracking studies: much of the grammar of sign language is found in the face [110].

1.2 Contributions

My thesis is that it is possible to compress and transmit intelligible video in real-time on an off-the-shelf mobile phone by adjusting the frame rate based on the activity and by coding the skin at a higher bit rate than the rest of the video. My goal is to save system resources while maintaining or increasing intelligibility. I focus on recognizing activity in sign language video to make cost-savings adjustments, a technique I call *variable frame rate*. I also create a *dynamic skin-based region-of-interest* that detects and encodes the skin at a higher bit rate than the rest of the frame.

Frame rates as low as 6 frames per second can be intelligible for signing, but higher frame rates are needed for finger spelling [30, 101, 55]. Because conversation involves turn-taking (times when one person is signing while the other is not), I save power as well as bit rate by lowering the frame rate during times of not signing, or "just listening" (see Figure 1.5). I also investigate changing the frame rate during finger spelling.

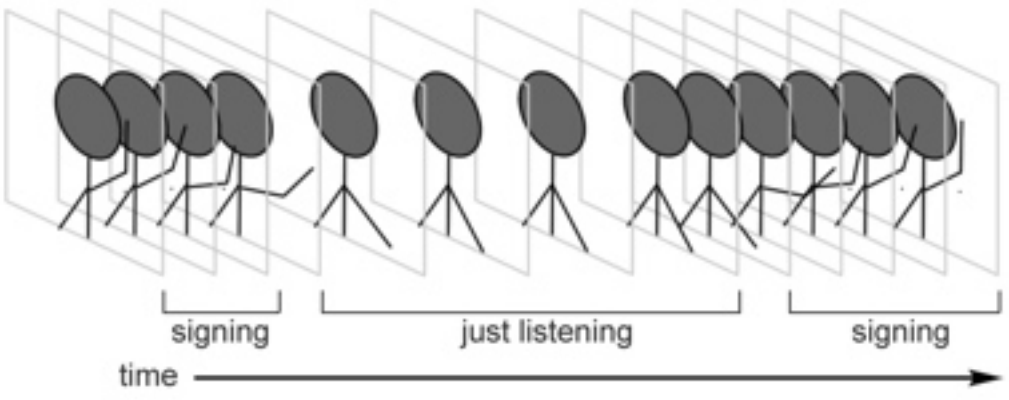


Figure 1.5: Variable frame rate. When the user is signing, we send the frames at the maximum possible rate. When the user is not signing, we lower the frame rate.

To prove this, I must show that a variable frame rate saves system resources and is intelligible. I must also show that real-time automatic recognition of the activity is possible on the phone and that making the skin clearer increases intelligibility. I must implement my techniques on the phone, verify the resource savings, and evaluate intelligibility through a user study.

1.2.1 Initial evaluation

I show in Chapter 3 that lowering the frame rate on the basis of the activity in the video can lead to savings in data transmitted and processor cycles, and thus power. I conduct a user study with members of the Deaf community in which they evaluate artificially created variable frame rate videos. The results of the study indicate that I can adjust the frame rate without too negatively affecting intelligibility.

1.2.2 Techniques for automatic recognition

My goal is to recognize the signing activity from a video stream in real-time on a standard mobile telephone. Since I want to increase accessibility, I do not restrict our users to special equipment or clothing. I only have access to the current frame of the conversational video of the signers, plus a limited history of what came before.

To accomplish my task, I harness two important pieces: the information available "for free" from the video encoder, and the fact that we have access to both sides of the conversation. The encoder I use is H.264, the state-of-the-art in video compression technology. H.264 works by finding motion vectors that describe how the current frame differs from previous ones. I use these, plus features based on the skin, as input to several different machine learning techniques that classify the frame as signing or not signing. I improve my results by taking advantage of the two-way nature of the video. Using the features from both conversation streams does not add complexity and allows me to better recognize the activity taking place. Chapter 4 contains my methods and results for real-time activity analysis.

I also try to increase intelligibility by focusing on the important parts of the video. Given

that much of the grammar of sign language is found in the face [110], I encode the skin at higher quality at the expense of the rest of the frame.

After verifying my techniques offline, I implement them on the phone. This presents several technical challenges, as the processing power on the phone is quite low. Chapter 5 describes the phone implementation.

1.2.3 Evaluation

I evaluate the sign language sensitive algorithms for variable frame rate and dynamic skin-based region-of-interest in a user study, contained in Chapter 6. I implement both methods within the video encoder on the phone to enable real-time compression and transmission. I assess my techniques in a user study in which the participants carry on unconstrained conversation on the phones in a laboratory setting. I gather both subjective and objective measures from the users.

The results of my study show that my skin-based ROI technique reduces guessing and increases comprehension. The variable frame rate technique results in more repeats and clarifications and in more conversational breakdowns, but this did not affect participants' likelihood of using the phone. Thus with my techniques, I can significantly decrease resource use without detracting from users' willingness to adopt the technology.

Chapter 2
BACKGROUND AND RELATED WORK

Compression of sign language video so that Deaf users can communicate over the telephone lines has been studied since at least the early 1980s. The first works attempted to enable communication by drastically modifying the video signal. Later, with the advent of higher bandwidth lines and the Internet, researchers focused on adjusting existing video compression algorithms to create more intelligible sign language videos. They also explored the limits of temporal compression in terms of the minimum frame rate required for intelligibility. Below, I detail early work on remote sign language communication; give some background on video compression; describe similar research in the area of sign language-specific video compression; and briefly overview the related area of sign language recognition, particularly how it applies to my activity analysis techniques.

2.1 Early work

The bandwidth of the copper lines that carry the voice signal is 9.6 kbps or 3 kHz, too low for even the best video compression methods 40 years later. The earliest works tested the bandwidth limitations for real-time sign language video communication over the phone lines and found that 100 kbps [83] or 21 kHz [100] was required for reasonable intelligibility. However, researchers also found that sign language motion is specific enough to be recognizable from a very small amount of information. Poizner et al. discovered that discrete signs are recognizable from the motion patterns of points of light attached to the hands [86]. Tartter and Knowlton conducted experiments with a small number of Deaf users and found they could understand each other from only seeing the motion of 27 points of light attached to the hands, wrists, and nose [107].

Building on this work, multiple researchers compressed sign language video by reducing multi-tone video to a series of binary images and transmitting them. Hsing and Sosnowski

took videos of a signer with dark gloves and thresholded the image so that it could be represented with 1 bit per pixel [46]. They then reduced the spatial resolution by a factor of 16 and tested with Deaf users, who rated the videos understandable. Pearson and Robinson used a more sophisticated method to render the video as binary cartoon line drawings [84]. Two Deaf people then carried on a conversation on their system. In the Telesign project, Letelier et al. built and tested a 64 kbps system that also rendered the video as cartoon line drawings [61]. Deaf users could understand signing at rates above 90%, but finger spelling was not intelligible. Harkins et al. created an algorithm that extracted features from video images and animated them on the receiving end [41]. Recognition rates were above 90% on isolated signs but low at the sentence level and for finger spelling.

More recently, Manoranjan and Robinson processed video into binary sketches and experimented with various picture sizes over a low bandwidth (33.5 kbps) and high bandwidth network [67]. In contrast to the preceding works, their system was actually implemented and worked in real-time. Two signers tested the system by asking questions and recording responses, and appeared to understand each other. Foulds used 51 optical markers on a signer's hands and arms, the center of the eyes, nose, and the vertical and horizontal limits of the mouth [31]. He converted this into a stick figure and temporally subsampled video down to 6 frames per second. He then interpolated the images on the other end using Bezier splines. Subjects recognized finger spelled words and isolated signs at rates of over 90%.

All of the above works achieve very low bit rate but suffer from several drawbacks. First, the binary images have to be transmitted separately and compressed using runtime coding or other algorithms associated with fax machines. The temporal advantage of video, namely that an image is not likely to differ very much from its predecessor, is lost. Moreover, complex backgrounds will make the images very noisy, since the edge detectors will capture color intensity differences in the background; the problem only worsens when the background is dynamic. Finally, much of the grammar of sign language is in the face. In these works, the facial expression of the signer is lost. The majority of the papers have very little in the way of evaluation, testing the systems in an ad-hoc manner and often only testing the accuracy of recognizing individual signs. Distinguishing between a small number of signs from a given pattern of lights or lines is an easy task for a human [86], but it is not the

same as conversing intelligibly at the sentence level.

2.2 Video compression

With the advent of the Internet and higher bandwidth connections, researchers began focusing on compressing video of sign language instead of an altered signal. A video is just a sequence of images, or *frames*. One obvious way to compress video is to separately compress each frame, using information found only within that frame. This method is called *intra-frame coding*. However, as noted above, this negates the temporal advantage of video. Modern video compression algorithms use information from other frames to code the current one; this is called inter-frame coding.

The latest standard in video compression is H.264. It performs significantly better than its predecessors, achieving the same quality at up to half the bit rate [118]. H.264 works by dividing a frame into 16×16 pixel *macroblocks*. These are compared to previously sent reference frames. The algorithm looks for exact or close matches for each macroblock from the reference frames. Depending on how close the match is, the macroblock is coded with the location of the match, the displacement, and whatever residual information is necessary. Macroblocks can be subdivided to the 4×4 pixel level. When a match cannot be found, the macroblock is coded as an intra block, from information within the current frame.

2.2.1 Region-of-interest and foveal compression

The availability of higher quality video at a lower bit rate led researchers to explore modifying standard video compression to work well on sign language video. Many were motivated by work investigating the focal region of ASL signers. Separate research groups used an eyetracker to follow the visual patterns of signers watching sign language video and determined that users focused almost entirely on the face [2, 71]. In some sense, this is intuitive, because humans perceive motion using their peripheral vision [9]. Signers can recognize the overall motion of the hands and process its contribution to the sign without shifting their gaze, allowing them to focus on the finer points of grammar found in the face.

One natural inclination is to increase the quality of the face in the video. Agrafiotis et al. implemented *foveal* compression, in which the macroblocks at the center of the user's focus

are coded at the highest quality and with the most bits; the quality falls off in concentric circles [2]. Their videos were not evaluated by Deaf users. Similarly, Woelders et al. took video with a specialized foveal camera and tested various spatial and temporal resolutions [120]. Signed sentences were understood at rates greater than 90%, though they did not test the foveal camera against a standard camera.

Other researchers have implemented region-of-interest encoding for reducing the bit rate of sign language video. A *region-of-interest*, or ROI, is simply an area of the frame that is coded at a higher quality at the expense of the rest of the frame. Schumeyer et al. suggest coding the skin as a region-of-interest for sign language videoconferencing [98]. Similarly, Saxe and Foulds used a sophisticated skin histogram technique to segment the skin in the video and compress it at higher quality [96]. Habili et al. also used advanced techniques to segment the skin [39]. None of these works evaluated their videos with Deaf users for intelligibility, and none of the methods are real-time.

2.2.2 Temporal compression

The above research focused on changing the spatial resolution to better compress the video. Another possibility is to reduce the temporal resolution. The temporal resolution, or *frame rate*, is the rate at which frames are displayed to the user. Early work found a sharp drop off in intelligibility of sign language video at 5 fps [83, 46]. Parish and Sperling created artificially subsampled videos with very low frame rates and found that when the frames are chosen intelligently (i.e. to correspond to the beginning and ending of signs), the low frame rate was far more understandable [82]. Johnson and Caird trained sign language novices to recognize 10 isolated signs, either as points of light or conventional video [55]. They found that users could learn signs at frame rates as low as 1 frame per second (fps), though they needed more attempts than at a higher frame rate. Sperling et al. explored the intelligibility of isolated signs at varying frame rates [101]. They found insignificant differences from 30 to 15 fps, a slight decrease in intelligibility from 15 to 10 fps, and a large decrease in intelligibility from 10 fps to 5 fps.

More recently, Hooper et al. looked at the effect of frame rates on the ability of sign

language students to understand ASL conversation [45]. They found that comprehension increased from 6 fps to 12 fps and again from 12 fps to 18 fps. The frame rate was particularly important when the grammar of the conversation was more complex, as when it included classifiers and transitions as opposed to just isolated signs. Woelders et al. looked at both spatial resolution and temporal resolution and found a significant drop off in understanding at 10 fps [120]. At rates of 15 fps, video comprehension was almost as good as the original 25 fps video. Finger spelling was not affected by the frame rates between 10 and 25 fps, possibly because the average speed of finger spelling is five to seven letters per second and thus 10 fps is sufficient [90].

Researchers also investigated the effect of delay on sign video communication and found that delay affects users less in visual communication than in oral communication [73]. The authors suggest three possible explanations: physiological and cognitive differences between auditory and visual perception; sign communication is tolerant of simultaneous signing; and the end of a turn is easily predicted.

2.3 Sign language recognition

Closely related to sign language video compression is sign language recognition. One possible way to achieve sign language compression is to recognize signs on one end, transmit them as text, and animate an avatar on the other end. There are several drawbacks to this approach. First of all, the problem of recognizing structured, three-dimensional gestures is quite difficult and progress has been slow; the state-of-the-art in sign language recognition is far behind that of speech recognition, with limited vocabularies, signer dependence, and constraints on the signers [66, 76]. Avatar animation is similarly limited. Secondly, there is no adequate written form of ASL. English and ASL are not equivalent. The system proposed above would require translation from ASL to English to transmit, and from English to ASL to animate, a difficult natural language processing problem. Most importantly, this approach takes the human element entirely out of the communication. Absent the face of the signer, emotion and nuance, and sometimes meaning, is lost. It is akin to putting a speech recognizer on a voice phone call, transmitting the text, and generating speech on the other end from the text. The computer can't capture pitch and tone, and nuance such as

sarcasm is lost. People prefer to hear a human voice rather than a computer, and prefer to see a face rather than an avatar.

Though my goal is not to recognize sign language, I use techniques from the literature in my activity analysis work. Signs in ASL are made up of five parameters: hand shape, movement, location, orientation, and nonmanual signals [109]. Recognizing sign language is mostly constrained to recognizing the first four. Nonmanual signals, such as the raising of eyebrows (which can change a statement into a question) or the puffing out of cheeks (which would add the adjective "big" or "fat" to the sign) are usually ignored in the literature. Without nonmanual signals, any kind of semantic understanding of sign language is far off. Nonetheless, progress has been made in recognition of manual signs.

2.3.1 Feature extraction for sign recognition

The most effective techniques for sign language recognition use direct-measure devices such as data gloves to input precise measurements on the hands. These measurements (finger flexion, hand location, roll, etc.) are then used as the features for training and testing purposes. While data gloves make sign recognition an easier problem to solve, they are expensive and cumbersome, and thus only suitable for constrained tasks such as data input at a terminal kiosk [4]. I focus instead on vision-based feature extraction.

The goal of feature extraction is to find a reduced representation of the data that models the most salient properties of the raw signal. Following Stokoe's notation [103], manual signals in ASL consist of hand shape, or *dez*; movement, or *sig*; location, or *tab*; and palm orientation, or *ori*. Most feature extraction techniques aim to recognize one or more of these parameters. By far the most common goal is to recognize hand shape. Some methods rotate and reorient the image of the hand, throwing away palm orientation information [65]. Others aim only to recognize the hand shape and don't bother with general sign recognition [50, 49, 65]. Location information, or where the sign occurs in reference to the rest of the body, is the second most commonly extracted feature. Most methods give only partial location information, such as relative distances between the hands or between the hands and the face. Movement is sometimes explicitly extracted as a feature, and other times

Features	Part of sign	Constraints	Time	1st Author
Real-time (measured in frames per second)				
COG; contour; movement; shape	dez, tab, sig	isolated	25 fps	Bowden [10]
COG	dez, ori	gloves; background; isolated	13 fps	Assan [5] Bauer [8]
COG, bounding ellipse	dez, tab, ori	gloves; background; no hand-face overlap; strong grammar	10 fps	Starner [102]
COG	dez, tab	isolated, one hand	n.r.	Kobayashi [60]
COG; Area; # protusions; motion direction	dez, tab, sig, ori	background; isolated	n.r.	Tanibata [106]
Not real-time (measured in seconds per frame)				
Fourier descriptors; optical flow	dez, sig	moving; isolated, one hand	1 s	Chen [15]
COG	dez, tab	background; isolated, one hand	3 s	Tamura [105]
Fourier descriptors	dez	moving; dark clothes; background; shape only	10 s	Huang [49]
Active shape models	dez	Background; shape only	25 s	Huang [50]
Intensity vector	dez	moving; isolated, one hand; away from face	58.3 s	Cui [21]
PCA	dez	isolated	n.r.	Imagawa [51]
Motion trajectory	sig	isolated	n.r.	Yang [122]

Table 2.1: Summary of feature extraction techniques and their constraints. The abbreviations are: *COG*, center of gravity of the hand; *dez*: hand shape; *tab*: location; *sig*: movement; *ori*: palm orientation; *background*: uniform background; *isolated*: only isolated signs were recognized, sometimes only one-handed; *gloves*: the signers wore colored gloves; *moving*: the hands were constantly moving; *n.r.*: not reported.

implicitly represented in the machine learning portion of the recognition. Palm orientation is not usually extracted as a separate feature, but comes along with hand shape recognition.

Table 2.1 summarizes the feature extraction methods of the main works on sign language recognition. I do not include accuracy because the testing procedures are so disparate. There is no standard corpus for sign language recognition, and some of the methods can only recognize one-handed isolated signs while others aim for continuous recognition. Ong and Ranganath have an excellent detailed survey on the wide range of techniques, their limitations, and how they compare to each other [76]. Here I focus on methods that inform my activity analysis.

The last column of the table lists the time complexity of the technique. If feature extraction is too slow to support a frame rate of 5 frames per second (fps), it is not real-time and thus not suitable to my purposes. This includes Huang et al. and Chen et al.'s Fourier descriptors to model hand shape [15, 49]; Cui and Weng's pixel intensity vector [21]; Huang and Jeng's active shape models [50]; and Tamura and Kawasaki's localization of the hands with respect to the body [105]. Though the time complexity was unreported, it is likely that Imagawa et al.'s principal component analysis of segmented hand images is not real-time [51]. Yang et al. also did not report on their time complexity, but their extraction of motion trajectories from successive frames uses multiple passes over the images to segment regions and thus is probably not real-time [122]. Nonetheless, it is interesting that they obtain good results on isolated sign recognition using only motion information.

Bowden et al. began by considering the linguistic aspects of British sign language, and made this explicitly their feature vector [10]. Instead of orientation, British sign language is characterized by the position of hands relative to each other (*ha*). They recognize *ha* via COG, *tab* by having a two dimensional contour track the body, *sig* by using the approximate size of the hand as a threshold, and *dez* by classifying the hand shape into one of six shapes. They use a rules-based classifier to group each sign along the four dimensions. Since they only have six categories for hand shape, the results aren't impressive, but the method deserves further exploration.

Most promising for my purposes are the techniques that use the center of gravity (COG) of the hand and/or face. When combined with relative distance to the fingers or face, COG

gives a rough estimate about the hand shape, and can give detailed location information. One way to easily pick out the hands from the video is to require the subjects to wear colored gloves. Assan and Grobel [5] and Bauer and Kraiss [8] use gloves with different colors for each finger, to make features easy to distinguish. They calculate the location of the hands and the COG for each finger, and use the distances between the COGs plus the angles of the fingers as their features. Tanibata et al. use skin detection to find the hands, then calculate the COG of the hand region relative to face, the area of hand region, the number of protrusions (i.e. fingers), and the direction of hand motion [106]. Signers were required to start in an initial pose. Kobayashi and Haruyama extract the head and the right hand using skin detection and use the relative distance between the two as their feature [60]. They recognized only one-handed isolated signs. Starner et al. use solid colored gloves to track the hands and require a strong grammar and no hand-face overlap [102]. Using COG plus the bounding ellipse of the hand, they obtain hand shape, location, and orientation information. In Chapter 5, I describe my skin-based features, which include the center of gravity, the bounding box, and the area of the skin.

2.3.2 Machine learning for sign recognition

Many of the researchers in sign language recognition use neural networks to train and test their systems [28, 29, 35, 49, 72, 111, 116, 122]. Neural networks are quite popular since they are simple to implement and can solve some complicated problems well. However, they are computationally expensive to train and test; they require many training examples lest they overfit; and they give a "black-box" solution to the classification problem, which does not help in identifying salient features for further refinement [93].

Decision trees and rules-based classifiers present another method for researchers to recognize sign language [89, 43, 51, 58, 94, 105]. These are quite fast, but sensitive to the rules chosen. Some works incorporate decision trees into a larger system that contains some other, more powerful machine learning technique, such as neural networks [75]. That idea holds promise; for instance, it makes sense to divide signs into two-handed and one-handed using some threshold, and then apply a more robust shape recognition algorithm.

The majority of research in sign language recognition uses hidden Markov models for sign classification [5, 8, 15, 29, 35, 50, 102, 106, 115, 117, 123]. Hidden Markov models are promising because they have been successfully applied to speech recognition. Support vector classifiers, another popular machine learning technique, are not used for sign language recognition, because they work best when distinguishing between a small number of classes. I describe experiments with both support vector classifiers and hidden Markov models in Chapter 4. In the next chapter, I motivate my activity analysis work by describing a user study that measured the effect of varying the frame rate on intelligibility.

Chapter 3

PILOT USER STUDY

My thesis is that I can save resources by varying the frame rate based on the activity in the video. My first step toward proving my thesis is to confirm that the variable frame rate does save resources and ensure that the videos are still comprehensible. To better understand intelligibility effects of altering the frame rate of sign language videos based on language content, I conducted a user study with members of the Deaf Community with the help of my colleague Anna Cavender [16]. The purpose of the study was to investigate the effects of (a) lowering the frame rate when the signer is not signing (or "just listening") and (b) increasing the frame rate when the signer is finger spelling. The hope was that the study results would motivate the implementation of my proposed automatic techniques for determining conversationally appropriate times for adjusting frame rates in real time with real users.

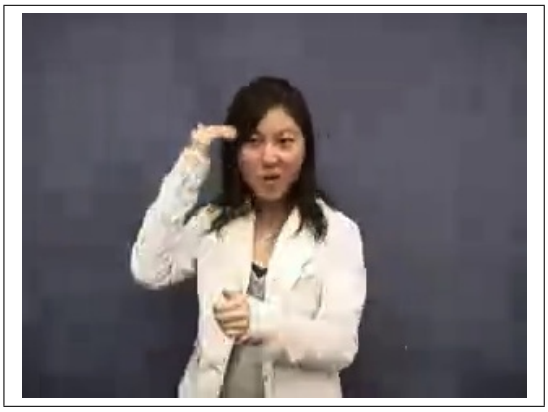
3.1 Study Design

The videos used in our study were recordings of conversations between two local Deaf women at their own natural signing pace. During the recording, the two women alternated standing in front of and behind the camera so that only one person is visible in a given video. The resulting videos contain a mixture of both signing and not signing (or "just listening") so that the viewer is only seeing one side of the conversation. The effect of variable frame rates was achieved through a "Wizard of Oz" method by first manually labeling video segments as signing, not signing, and finger spelling and then varying the frame rate during those segments.

Figure 3.1 shows some screen shots of the videos. The signer is standing in front of a black background. The field of view and "signing box" is larger than on the phone, and the signer's focus is the woman behind the camera, slightly to the left. Notice that the two

signing frames differ in the largeness of motion for the hands. While Figure 3.1(a) is more easily recognizable as signing, these sorts of frames actually occur with less frequency than the smaller motion observed in Figure 3.1(b). Moreover, the more typical smaller motion is not too far removed from the finger spelling seen in Figure 3.1(c).

(a) Large motion signing



(b) Small motion signing

(c) Finger spelling

Figure 3.1: Screen shots depicting the different types of signing in the videos.

We wanted each participant to view and evaluate each of the 10 encoding techniques described below without watching the same video twice and so we created 10 different videos, each a different part of the conversations. The videos varied in length from 0:34 minutes to 2:05 minutes (mean = 1:13) and all were recorded with the same location, lighting conditions, and background. The x264 codec [3], an open source implementation

of the H.264 (MPEG-4 part 10) standard [118], was used to compress the videos.

Both videos and interactive questionnaires were shown on a Sprint PPC 6700, PDA-style video phone with a 320 × 240 pixel resolution (2.8" × 2.1") screen.

3.1.1 Signing vs. Not Signing

We studied four different frame rate combinations for videos containing periods of signing and periods of not signing. Previous studies indicate that 10 frames per second (fps) is adequate for sign language intelligibility, so we chose 10 fps as the frame rate for the signing portion of each video. For the non-signing portion, we studied 10, 5, 1, and 0 fps. The 0 fps means that one frame was shown for the entire duration of the non-signing segment regardless of how many seconds it lasted (a freeze-frame effect).

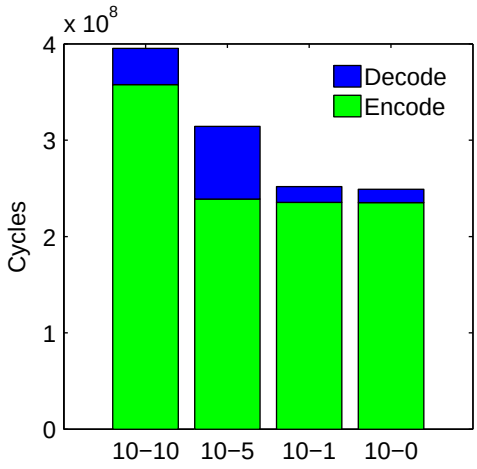
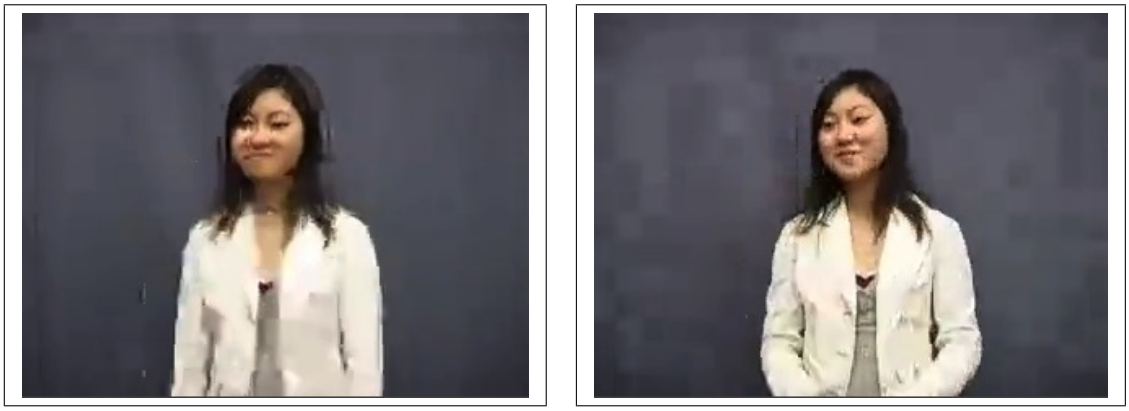


Figure 3.2: Average processor cycles per second for the four different variable frame rates. The first number is the frame rate during the signing period and the second number is the frame rate during the not signing period.

Even though the frame rate varied during the videos, the bits allocated to each frame were held constant so that the perceived quality of the videos would remain as consistent as possible across different encoding techniques. This means that the amount of data transmitted would decrease with decreased frame rate and increase for increased frame rate. The maximum bit rate was 50 kbps.

Figure 3.2 shows the average cycles per second required to encode video using these four techniques and the savings gained from reducing the frame rate during times of not signing. A similar bit rate savings was observed; on average, there was a 13% savings in bit rate from 10-10 to 10-5, a 25% savings from 10-10 to 10-1, and a 27% savings from 10-10 to 10-0.

The degradation in quality at the lower frame rate is clear in Figure 3.3. On the left is a frame sent at 1 fps, during the "just listening" portion of the video. On the right is a frame sent at 10 fps.



(a) Screen shot at 1 fps

(b) Screen shot at 10 fps

Figure 3.3: Screen shots at 1 and 10 fps.

3.1.2 Signing vs. Finger spelling

We studied six different frame rate combinations for videos containing both signing and finger spelling. Even though our previous studies indicate that 10 fps is adequate for sign language intelligibility, it is not clear that that frame rate will be adequate for the finger spelling portions of the conversation. During finger spelling, many letters are quickly produced on the hand(s) of the signer and if fewer frames are shown per second, critical letters may be lost. We wanted to study a range of frame rate increases in order to study both the effect of frame rate and *change* in frame rate on intelligibility. Thus, we studied 5, 10, and 15 frames per second for both the signing and finger spelling portions of the videos resulting in six different combinations for signing and finger spelling: (5,5), (5, 10), (5, 15),

(10, 10), (10, 15), and (15, 15). For obvious reasons, we did not study the cases where the frame rate for finger spelling was lower than the frame rate for signing.

3.1.3 Study Procedure

Six adult, female members of the Deaf Community between the ages of 24 and 38 participated in the study. All six were Deaf and had life-long experience with ASL; all but one (who used Signed Exact English in grade school and learned ASL at age 12) began learning ASL at age 3 or younger. All participants were shown one practice video to serve as a point of reference for the upcoming videos and to introduce users to the format of the study. They then watched 10 videos: one for each of the encoding techniques described above.

Following each video, participants answered a five- or six- question, multiple choice survey about her impressions of the video (see Figure 3.5). The first question asked about the content of the video, such as "Q0: What kind of food is served at the dorm?" For the Signing vs. Finger spelling videos, the next question asked "Q1: Did you see all the finger-spelled letters or did you use context from the rest of the sentence to understand the word?" The next four questions are shown in Figure 3.4.

The viewing order of the different videos and different encoding techniques for each part of the study (four for Signing vs. Not Signing and six for Signing vs. Finger spelling) was determined by a Latin squares design to avoid effects of learning, fatigue, and/or variance of signing or signer on the participant ratings. Post hoc analysis of the results found no significant differences between the ratings of any of the 10 conversational videos. This means we can safely assume that the intelligibility results that follow are due to varied compression techniques rather than other potentially confounding factors (e.g. different signers, difficulty of signs, lighting or clothing issues that might have made some videos more or less intelligible than others).

3.2 Results

For the variable frame rates studied here, we did not vary the quality of the frames and so the level of distortion was constant across test sets. Thus, one would expect to see higher ratings for higher frame rates, since the bit rates are also higher. Our hope was that

During the video, how often did you have to guess about what the signer was saying?

not at all $\frac{1}{4}$ time $\frac{1}{2}$ time $\frac{3}{4}$ time all the time

How easy or how difficult was it to understand the video?
(where 1 is very difficult and 5 is very easy).

1 2 3 4 5

Changing the frame rate of the video can be distracting. How would you rate the annoyance level of the video?
(where 1 is not annoying at all and 5 is extremely annoying).

1 2 3 4 5

If video of this quality were available on the cell phone, would you use it?

definitely probably maybe probably not definitely not

Figure 3.4: Questionnaire for pilot study.

the ratings would not be statistically significant meaning that our frame rate conservation techniques do not significantly harm intelligibility.

3.2.1 Signing vs. Not Signing

For all of the frame rate values studied for non-signing segments of videos, survey responses did not yield a statistically significant effect on frame rate. This means that we did not detect a significant preference for any of the four reduced frame rate encoding techniques

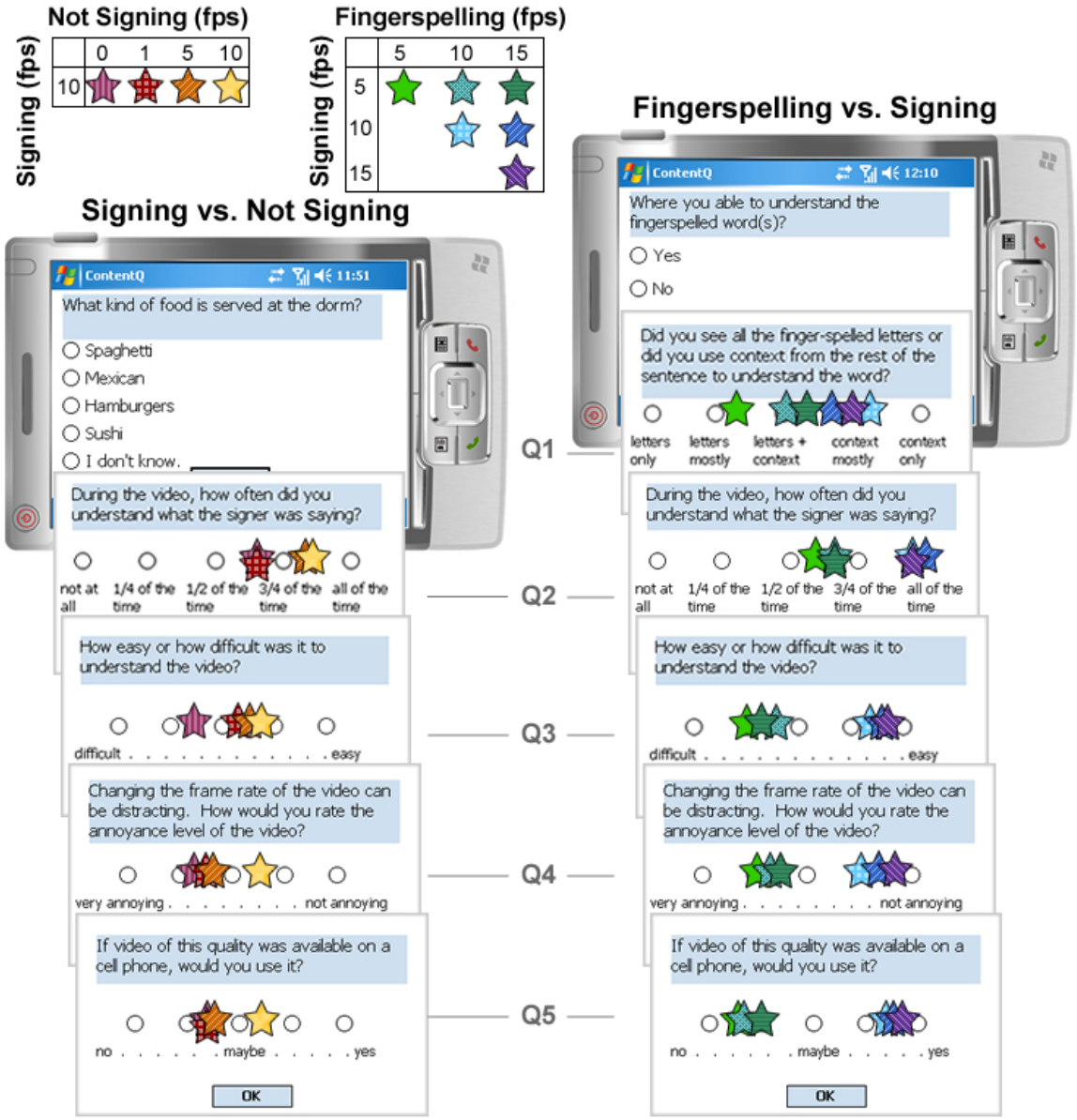


Figure 3.5: Average ratings on survey questions for variable frame rate encodings (stars).

studied here, even in the case of 0 fps (the freeze frame effect of having one frame for the entire non-signing segment). Numeric and graphical results can be seen in Table 3.1 and Figure 3.5. This result may indicate that we can obtain savings by reducing the frame rate during times of not signing without significantly affecting intelligibility.

Signing v Not Signing (fps)	10 v 0 {SD}	10 v 1 {SD}	10 v 5 {SD}	10 v 10 {SD}	Significance ($F_{3,15}$)
Q2					
0 not at all	0.71	0.71	0.79	0.83	1.00
1 all the time	{1.88}	{0.10}	{0.19}	{0.20}	*n.s.*
Q3					
1 difficult	2.50	3.17	3.50	3.83	1.99
5 easy	{1.64}	{0.98}	{1.05}	{1.17}	*n.s.*
Q4					
1 very annoying	2.17	2.50	2.83	3.67	1.98
5 not annoying	{1.33}	{1.05}	{1.33}	{1.51}	*n.s.*
Q5					
1 no	2.33	2.33	2.50	3.33	1.03
5 yes	{1.75}	{1.37}	{1.52}	{1.37}	*n.s.*

Table 3.1: Average participant ratings and significance for videos with reduced frame rates during non-signing segments. Standard deviation (SD) in {}, *n.s.* is not significant. Refer to Figure 3.4 for the questionnaire.

Many participants anecdotally felt that the lack of feedback for the 0 fps condition seemed conversationally unnatural; they mentioned being uncertain about whether the video froze, the connection was lost, or their end of the conversation was not received. For these reasons, it may be best to choose 1 or 5 fps, rather than 0 fps, so that some of feedback that would occur in a face to face conversation is still available (such as head nods and expressions of misunderstanding or needed clarification).

3.2.2 Signing vs. Finger spelling

For the six frame rate values studied during finger spelling segments, we did find a significant effect of frame rate on participant preference (see Table 3.2). As expected, participants preferred the encodings with the highest frame rates (15 fps for both the signing and finger

Signing v Finger spelling (fps)	5 v 5 {SD}	5 v 10 {SD}	5 v 15 {SD}	10 v 10 {SD}	10 v 15 {SD}	15 v 15 {SD}	Sig ($F_{5,25}$)
Q1							
1 letters only	2.17	3.00	3.33	4.17	3.67	4.00	3.23
5 context only	{0.75}	{1.26}	{1.37}	{0.98}	{1.21}	{0.89}	*n.s.*
Q2							
0 not at all	0.54	0.67	0.67	0.96	1.00	0.96	7.47
1 all the time	{0.19}	{0.38}	{0.20}	{0.10}	{0.00}	{0.10}	$p < .01$
Q3							
1 difficult	2.00	2.67	2.33	4.17	4.67	4.83	13.04
5 easy	{0.63}	{1.37}	{1.21}	{0.41}	{0.82}	{0.41}	$p < .01$
Q4							
1 very annoying	2.00	2.17	2.33	4.00	4.33	4.83	14.86
5 not annoying	{0.89}	{1.36}	{1.21}	{0.89}	{0.82}	{0.41}	$p < .01$
Q5							
1 no	1.67	1.83	2.00	4.17	4.50	4.83	18.24
5 yes	{0.52}	{1.60}	{0.89}	{0.98}	{0.84}	{0.41}	$p < .01$

Table 3.2: Average participant ratings and significance for videos with increased frame rates during finger spelling segments. Standard deviation (SD) in {}, *n.s.* is not significant. Refer to Figure 3.4 for the questionnaire.

spelling segments), but only slight differences were observed for videos encoded at 10 and 15 fps for finger spelling when 10 fps was used for signing. Observe that in Figure 3.5, there is a large drop in ratings for videos with 5 fps for the signing parts of the videos. In fact, participants indicated that they understood only slightly more than half of what was said in the videos encoded with 5 fps for the signing parts (Q2). The frame rate during signing most strongly affected intelligibility, whereas the frame rate during finger spelling seemed to have a smaller effect on the ratings.

This result is confirmed by the anecdotal responses of study participants. Many felt that

the increased frame rate during finger spelling was nice, but not necessary. In fact many felt that if the higher frame rate were available, they would prefer that during the entire conversation, not just during finger spelling. We did not see these types of responses in the Signing vs. Not Signing part of the study, and this may indicate that 5 fps is just too low for comfortable sign language conversation. Participants understood the need for bit rate and frame rate cutbacks, yet suggested the frame rate be higher than 5 fps if possible.

These results indicate that frame rate (and thus bit rate) savings are possible by reducing the frame rate when times of not signing (or "just listening") are detected. While increased frame rate during finger spelling did not have negative effects on intelligibility, it did not seem to have positive effects either. In this case, videos with increased frame rate during finger spelling were more positively rated, but the more critical factor was the frame rate of the signing itself. Increasing the frame rate for finger spelling would only be beneficial if the base frame rate were sufficiently high, such as an increase from 10 fps to 15 fps. However, we note that the type of finger spelling in the videos was heavily context-based; that is, the words were mostly isolated commonly fingerspelled words, or place names that were familiar to the participants. This result may not hold for unfamiliar names or technical terms, for which understanding each individual letter would be more important.

In order for these savings to be realized during real time sign language conversations, a system for automatically detecting the time segments of "just listening" is needed. The following chapter describes some methods for real-time activity analysis.

Chapter 4

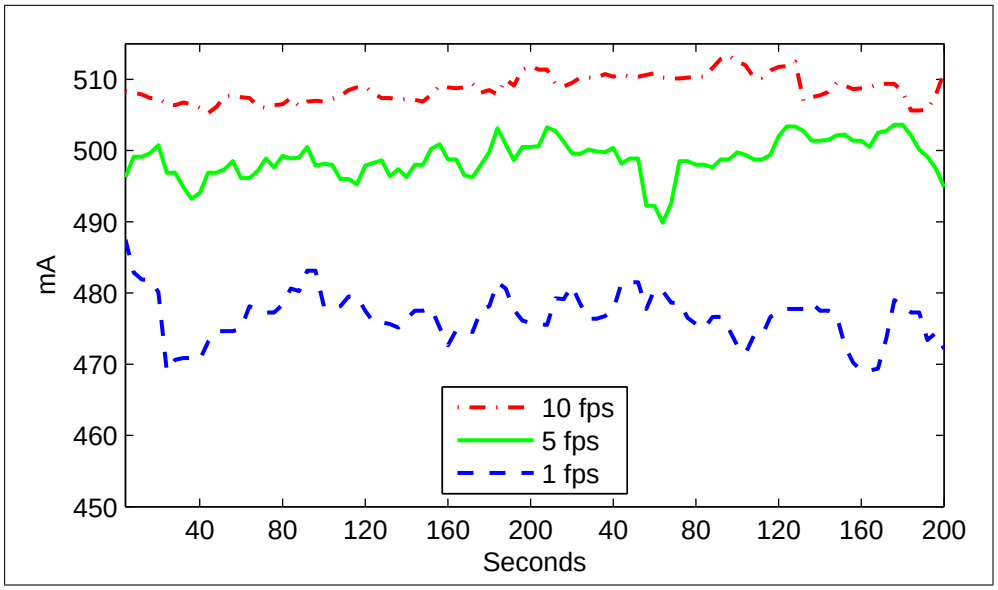
REAL-TIME ACTIVITY ANALYSIS

The pilot user study confirmed that I could vary the frame rate without significantly affecting intelligibility. In this chapter I study the actual power savings gained when encoding and transmitting at different frame rates. I then explore some possible methods for recognizing periods of signing in real-time on users that wear no special equipment or clothing.

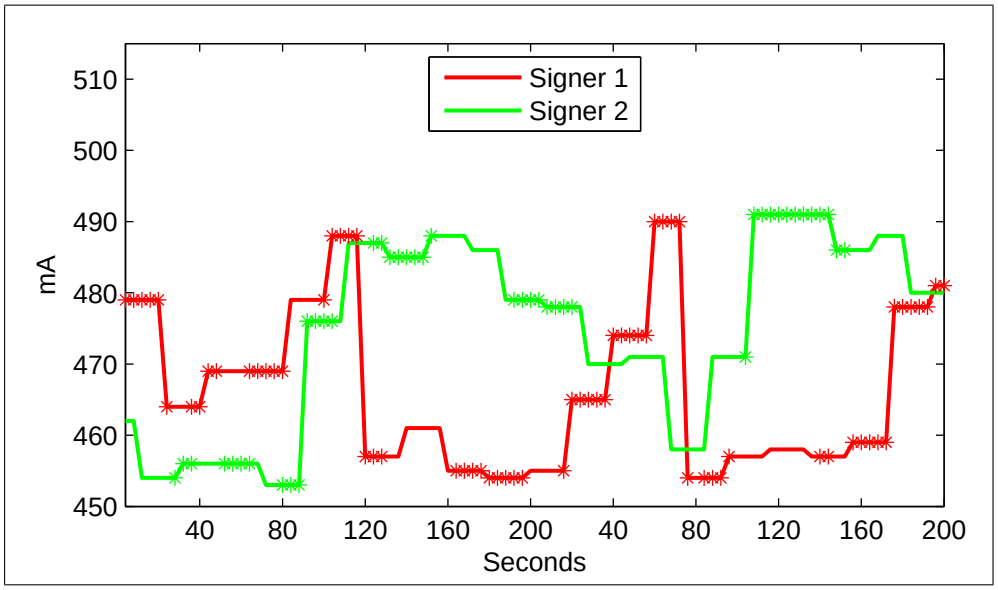
4.1 Power Study

Battery life is an important consideration in software development on a mobile phone. A short-lived battery makes a phone much less useful. In their detailed study of the power breakdown for a handheld device, Viredaz and Wallach found that playing video consumed the most power of any of their benchmarks [113]. In deep sleep mode, the device's battery lasted 40 hours, but it only lasted 2.4 hours when playing back video. Only a tiny portion of that power was consumed by the LCD screen. Roughly 1/4 of the power was consumed by the core of the processor, 1/4 by the input-output interface of the processor (including flash memory and daughter-card buffers), 1/4 by the DRAM, and 1/4 by the rest of the components (mainly the speaker and the power supply). The variable frame rate saves cycles in the processor, a substantial portion of the power consumption, so it is natural to test whether it saves power as well.

In order to quantify the power savings from dropping the frame rate during less important segments, I monitored the power use of MobileASL on a Sprint PPC 6700 at various frame rates [17]. MobileASL normally encodes and transmits video from the cell phone camera. I modified it to read from an uncompressed video file and encode and transmit frames as though the frames were coming from the camera. I was thus able to test the power usage at different frame rates on realistic conversational video.



(a) Average power use over all videos.



(b) Power use at 1 fps for one conversation. Stars indicate which user is signing.

Figure 4.1: Power study results.

The conversational videos were recorded directly into raw YUV format from a web cam. Signers carried on a conversation at their natural pace over a web cam/wireless connection. Two pairs recorded two different conversations in different locations, for a total of eight

videos. For each pair, one conversation took place in a "noisy" location, with lots of people walking around behind the signer, and one conversation took place in a "quiet" location with a stable background. I encoded the videos with x264 [3].

I used a publicly available power meter program [1] to sample the power usage at 2 second intervals. We had found in our pilot study that the minimum frame rate necessary for intelligible signing is 10 frames per second (fps), but rates as low as 1 fps are acceptable for the "just listening" portions of the video. Thus, I measured the power usage at 10 fps, 5 fps, and 1 fps. Power is measured in milliamps (mA) and the baseline power usage, when running MobileASL but not encoding video, is 420 mA.

Figure 4.1 shows (a) the average power usage over all our videos and (b) the power usage of a two-sided conversation at 1 fps. On average, encoding and transmitting video at 10 fps requires 17.8% more power than at 5 fps, and 35.1% more power than at 1 fps. Figure 4.1(b) has stars at periods of signing for each signer. Note that as the two signers take turns in the conversation, the power usage spikes for the primary signer and declines for the person now "just listening." The spikes are due to the extra work required of the encoder to estimate the motion compensation for the extra motion during periods of signing, especially at low frame rates. In general the stars occur at the spikes in power usage, or as the power usage begins to increase. Thus, while we can gain power saving by dropping the frame rate during periods of not signing, it would be detrimental to the power savings, as well as the intelligibility, to drop the frame rate during any other time.

4.2 Early work on activity recognition

My methods for classifying frames have evolved over time and are reflected in the following sections.

4.2.1 Overview of activity analysis

Figure 4.2 gives a general overview of my activity recognition method for sign language video. The machine learning classifier is trained with labeled data, that is, features extracted from frames that have been hand-classified as signing or listening. Then for the actual recognition

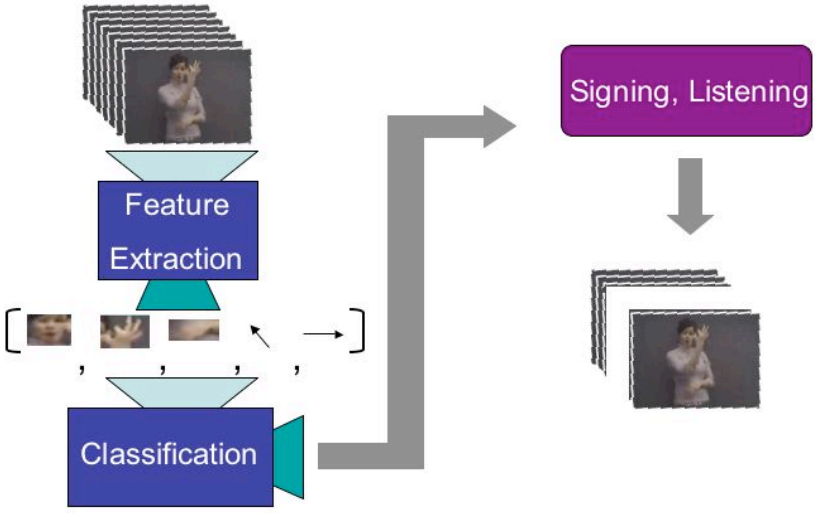


Figure 4.2: General overview of activity recognition. Features are extracted from the video and sent to a classifier, which then determines if the frame is signing or listening and varies the frame rate accordingly.

step, I extract the salient features from the frame and send it to the classifier. The classifier determines if the frame is signing or listening, and lowers the frame rate in the latter case.

Recall that for the purposes of frame rate variation, I can only use the information available to me from the video stream. I do not have access to the full video; nor am I able to keep more than a small history in memory. I also must be able to determine the class of activity in real time, on users that wear no special equipment or clothing.

For my first attempt at solving this problem, I used the four videos from the user study in the previous chapter. In each video, the same signer is filmed by a stationary camera, and she is signing roughly half of the time. I am using an easy case as my initial attempt, but if my methods do not work well here, they will not work well on more realistic videos. I used four different techniques to classify each video into signing and not signing portions. In all the methods, I train on three of the videos and test on the fourth. I present all results as comparisons to the ground truth manual labeling.

4.2.2 Differencing

A baseline method is to examine the pixel differences between successive frames in the video. If frames are very different from one to the next, that indicates a lot of activity and thus that the user might be signing. On the other hand, if the frames are very similar, there is not a lot of motion so the user is probably not signing. As each frame is processed, its luminance component is subtracted from the previous frame, and if the differences in pixel values are above a certain threshold, the frame is classified as a signing frame. This method is sensitive to extraneous motion and is thus not a good general purpose solution, but it gives a good baseline from which to improve. Figure 4.3 shows the luminance pixel differences as the subtraction of the previous frame from the current. Lighter pixels correspond to bigger differences; thus, there is a lot of motion around the hands but not nearly as much by the face.

Formally, for each frame k in the video, I obtain the luminance component of each pixel location (i, j). I subtract from it the luminance component of the previous frame at the same pixel location. If the sum of absolute differences is above the threshold τ, I classify the frame as signing. Let $f(k)$ be the classification of the frame and $I_k(i, j)$ be the luminance component of pixel (i, j) at frame k. Call the difference between frame k and frame $k - 1$ $d(k)$, and let $d(1) = 0$. Then:

$$d(k) = \sum_{(i,j)\in I_k} |I_k(i, j) - I_{k-1}(i, j)| \tag{4.1}$$

$$f(k) = \begin{cases} 1 & \text{if } d(k) > \tau \\ -1 & \text{otherwise} \end{cases} \tag{4.2}$$

To determine the proper threshold τ, I train my method on several different videos and use the threshold that returns the best classification on the test video. The results are shown in the first row of Table 4.1. Differencing performs reasonably well on these videos.

Figure 4.3: Difference image. The sum of pixel differences is often used as a baseline.

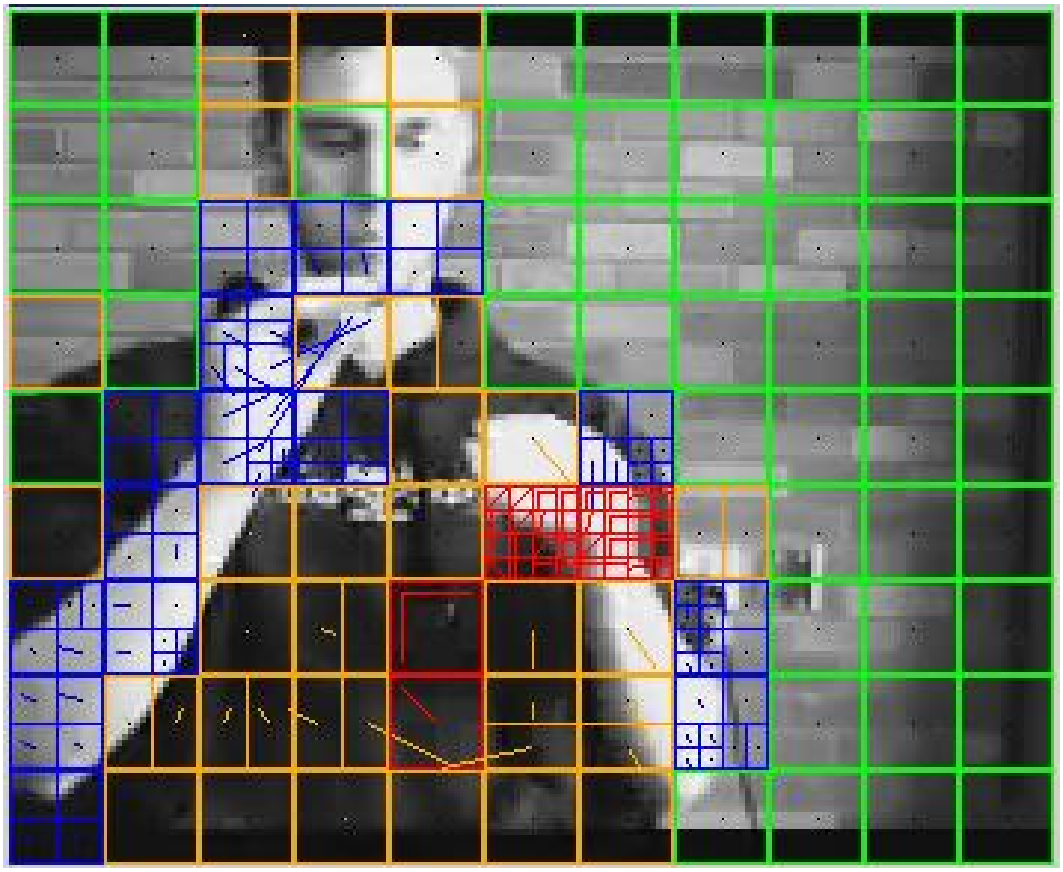


Figure 4.4: Visualization of the macroblocks. The lines emanating from the centers of the squares are motion vectors.

4.2.3 SVM

The differencing method performs well on these videos, because the camera is stationary and the background is fixed. However, a major weakness of differencing is that it is very sensitive to camera motion and to changes in the background, such as people walking by. For the application of sign language over cell phones, the users will often be holding the camera themselves, which will result in jerkiness that the differencing method would improperly classify. In general I would like a more robust solution.

I can make more sophisticated use of the information available to us. Specifically, the H.264 video encoder has motion information in the form of motion vectors. For a video

encoded at a reasonable frame rate, there is not much change from one frame to the next. H.264 takes advantage of this fact by first sending all the pixel information in one frame, and from then on sending a vector that corresponds to the part of the previous frame that looks most like this frame plus some residual information. More concretely, each frame is divided into macroblocks that are 16×16 pixels. The compression algorithm examines the following choices for each macroblock and chooses the cheapest (in bits) that is of reasonable quality:

1. Send a "skip" block, indicating that this macroblock is exactly the same as the previous frame.

2. Send a vector pointing to the location in the previous frame that looks most like this macroblock, plus residual error information.

3. Subdivide the macroblock and reexamine these choices.

4. Send an "I" block, or intra block, essentially the macroblock uncompressed.

Choices 2 and 3 have motion vectors associated with them; choice 4 does not. Choice 1 means no motion at all; choice 2 might indicate a big, sweeping motion, while choice 3 might indicate small, rapid movements. Choice 4 usually indicates the most motion of all, since the encoder only resorts to it when it cannot find a section of the previous frame that matches this macroblock. Figure 4.4 shows a visualization of the macroblocks, with the subdivisions and motion vectors. The green blocks are skip blocks, corresponding to choice 1; the orange and blue blocks have motion vectors associated with them, emanating from their center, and are subdivided; and the red blocks are I blocks. Note that the I blocks are centered around the left hand moving rapidly toward the right, while there are motion vectors associated with the slower motions of the right hand. As expected, the skip blocks are in the background. The encoder chooses the cheapest of these in terms of bits and sends it.

For each frame, I can obtain either motion vector information for each macroblock or an indication that the encoder gave up. This is quite useful for determining what kind of

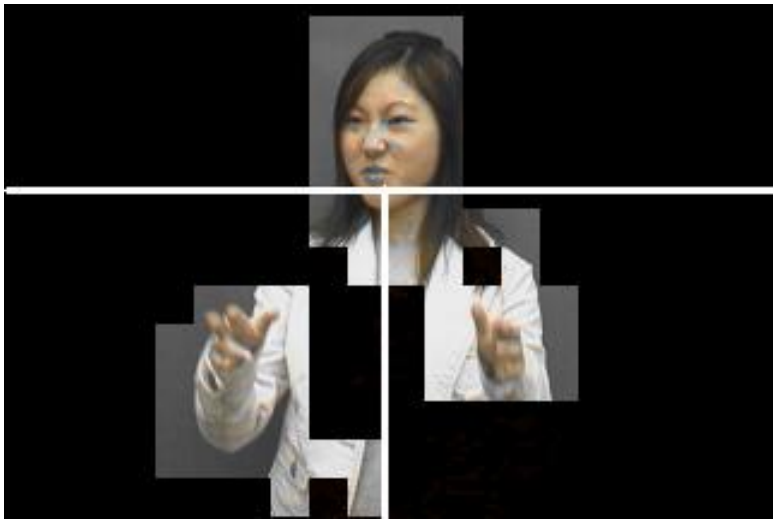


Figure 4.5: Macroblocks labeled as skin and the corresponding frame division.

activity is taking place in the video. If I know the hands are involved in big motions, I can classify the frame as a signing frame; conversely, if the hands and face are not moving very much, I can classify the frame as not signing.

I do not need all of the motion vector information from all of the macroblocks. Instead, I focus on the face and hands. I perform skin-detection on the video to determine the macroblocks most likely to contain the face and hands. I detect skin via a simple and well-known RGB-based algorithm [85] that works for many different skin tones and can be performed in real-time. I then divide the frame into three parts: the top third, corresponding to the face, and the bottom two thirds divided in half, corresponding to the left and right hands. Any macroblock with majority skin pixels we classify as skin. For those macroblocks, I calculate a summary motion vector for the face, right hand, and left hand. As an additional feature, I count the overall number of I-blocks in the frame. Figure 4.5 shows the macroblocks classified as skin and the frame division. Note that this simple method won't always correspond to the face and hands, but it mostly yields reasonable results. The total feature vector is (face_x, face_y, right_x, right_y, left_x, left_y, #I blocks) where the subscripts x and y are the components of the motion vector for that region, and I is the number of I blocks.

A well-known solution to the classification problem is Support Vector Machines (SVM)

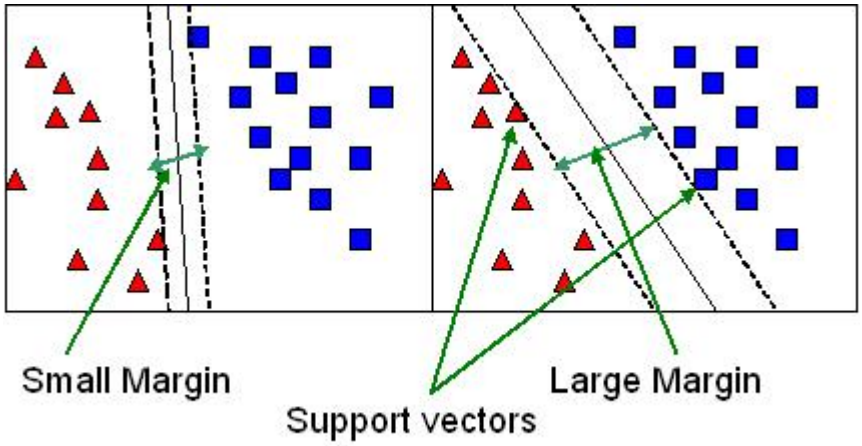


Figure 4.6: Optimal separating hyperplane.

[20]. An SVM is an algorithm that, given labeled training data in the form of features and their classes, determines the optimal separating hyperplane (see Figure 4.6). The hyperplane is not necessarily in the same dimension as the feature space; in fact, it is usually transformed nonlinearly to a higher dimensional space in which greater separation may be achieved. Often, the data is not actually separable. In this case, the SVM uses error terms and optimizes for the minimum error.

I use libsvm [14], a publicly available software package, to train and test our data. As with differencing, I train on three of the videos and test on the fourth. Row 3 of Table 4.1 contains the results. On the first two videos, the differencing method does better, but on the last two, SVM is superior.

4.2.4 Combination

Given these results, it would be nice to have the best of both worlds; that is, to combine the results of SVM with the results of the differencing method to make the best classification choice possible.

The SVM returns a classification based on which side of the hyperplane the test feature vector is on. Furthermore, it also returns the distance between the hyperplane and the feature vector. The distance can be viewed as a confidence value. If a feature vector is far

from the dividing hyperplane, we are very certain of its classification. On the other hand, if a feature vector is close to the hyperplane, we are unsure if the classification is correct.

I can use a similar measure of distance for the differencing method. If the difference is close to the threshold on either side, I am not very confident of my classification, but if the difference is much bigger or much smaller than the threshold, I can be sure I am correct.

I combine differencing and SVM as follows. When the SVM strongly classifies a vector, I use its classification. Otherwise, I determine the classification by weighting the percent threshold, comparing it to the SVM distance and choosing the classification of the larger one. Recall the definition of $d(k)$, $f(k)$, and τ from Equations 1 and 2. Let $g(k)$ be the classification returned by the SVM and $p(k)$ be the distance from the hyperplane. Let ω be the weighting factor. Then

$$h(k) = \begin{cases} f(k) & \text{if } p(k) < \omega \left| \frac{d(k)-\tau}{\tau} \right| \\ g(k) & \text{otherwise} \end{cases}$$

I empirically determined ω to be 3.01; this weighting factor resulted in the best possible value for most of the videos, and close to the best for the remainder.

4.2.5 Hidden Markov models

The majority of research in sign language recognition uses hidden Markov models for sign classification [5, 8, 15, 29, 35, 50, 102, 106, 115, 117, 123]. Hidden Markov models are appealing because they have been successfully applied to speech recognition.

A Markov model is simply a finite state machine in which the probability of the next state depends only on the current state, that is:

$$Pr(X_{t+1} = x | X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_1 = x_1) = Pr(X_{t+1} = x | X_t = x_t).$$

In a regular Markov model, the state is known; in a hidden Markov model, the state is not known, but there is an observation at each step and some probability that each state produced the observation. Intuitively, suppose there are two friends, Bob and Alice. Every day Alice calls Bob and tells him what she did that day. Alice either goes shopping, goes to a movie, or plays baseball, depending on the weather (see Figure 4.7). So, if it's sunny,

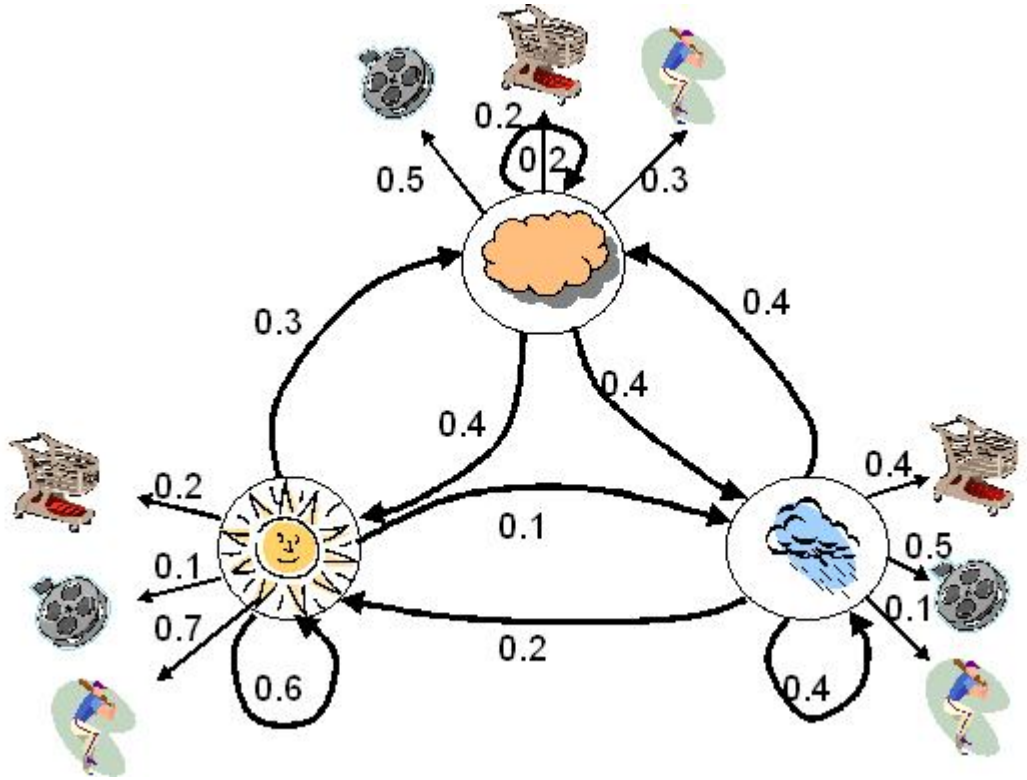


Figure 4.7: Graphical representation of a hidden Markov model. The hidden states correspond to the weather: sunny, cloudy, and rainy. The observations are Alice's activities.

she's likely to play baseball, and if it's rainy, she'll probably go to a movie or go shopping. Additionally, when it's sunny one day, there's a good chance it will be sunny the next day; but if it's cloudy, it could rain the next day. For Bob, the weather represents the hidden states, and based on Alice's activities over a series of days, he can guess what the weather has been.

There are three canonical problems for hidden Markov models [88]. First, given an observation sequence and a model, find the probability that the model produced the observation sequence. Second, given an observation sequence and a model, find the corresponding state sequence that best explains the observation sequence. Third, given an observation sequence, adjust the parameters of the model to maximize the probability of the observation sequence given the model. This last canonical problem is equivalent to training the models.

The first canonical problem is the one used to solve speech recognition. In speech recognition, a model is created for each phoneme. These are tied together into models representing words. Then, an utterance is the observation and the word model that most likely produced that utterance is the recognized word.

Starner et al. use the second canonical problem for sign recognition [102]. They apply a hidden Markov model for continuous sign recognition, so that the model implicitly segments the signs. The model is sensitive to transitions between signs, because the starting point of a sign will change depending on the previous sign. They mitigate this by imposing a strong grammar, that is, only allowing certain signs to come before or after each other.

Following the first canonical problem, I can train two models, one for signing and one for not signing, and then recognize the test data by selecting the model that maximizes the probability of the observations. This is the approach used in speech recognition and by Vogler [115] for sign recognition. Another possibility is to attempt to represent the entire video sequence with one model, and then recover the state sequence; presumably, the states will correspond to "signing" and "not signing." Starner et al. [102] used this technique for general sign recognition, but imposed a strong grammar so that their underlying model would be accurate and they would be able to recognize the signs by knowing the state. For my purposes, it is unclear how to model signing and not signing except with two states, and this is too few for the HMM to work well. Also, I would have to impose a delay since the entire sequence is not available to me. Thus I choose to follow the first approach.

The feature data must be more than one frame, to capture the temporal nature of the problem. If there is only one observation per "word," the model degenerates into a single instance of flipping a biased coin, or having some probability that the frame is signing or not signing, and doesn't harness any of the useful information about what came before the observation. I start with a three frame sliding window. The feature vector x is thus 3×7, where each column consists of the motion vector components used in the SVM: (face_x, face_y, right_x, right_y, left_x, left_y, #I blocks). The first column is the components of the frame two before this one, the second column is the components of the frame immediately previous to this one, and the last column is the components of the current frame. I then try my technique on a four frame and five frame sliding window. Two three-state hidden Markov

Classification method	Video 1	Video 2	Video 3	Video 4	Average
Differencing	88.1%	87.2%	88.8%	86.0%	87.5%
SVM	87.8%	85.2%	90.6%	86.6%	87.6 %
Combination	**89.9%**	88.5%	91.1%	87.2%	**89.2%**
HMM-3	87.3%	85.4%	87.3%	82.6%	85.7%
HMM-4	88.4%	86.0%	88.6%	82.5%	86.4%
HMM-5	88.4%	86.8%	89.2%	81.4%	86.5%
SVM-3	88.8%	87.4%	**91.3%**	87.1%	88.7%
SVM-4	87.9 %	**90.3%**	91.1%	**87.6%**	**89.2%**
SVM-5	88.7 %	88.3%	**91.3%**	**87.6%**	89.0%

Table 4.1: Results for the differencing method, SVM, and the combination method, plus the sliding window HMM and SVM. The number next to the method indicates the window size. The best results for each video are in bold.

models are trained on labeled feature data. The first is trained only on signing examples, and the second is trained only on not signing examples. The widely used, publicly available HTK package [104] is used for the hidden Markov model implementation.

4.2.6 Preliminary Results

In addition to the above methods, I improved the results for the SVM by also considering the classification of frames immediately previous to the one to be classified. I look at the classification returned by the SVM for the two frames before this one, plus the current classification, and output the majority vote. I also tried a four frame and five frame window. I experimented with different weightings on each frame, but found weighting them equally worked best.

My results are in Table 4.1. The top three rows correspond to the first three methods detailed (differencing, SVM without a sliding window, and combination). The next three are the hidden Markov model results for different sliding windows, and the final three are the

SVM results for different sliding windows. The recognition accuracy increased for the most part as the sliding window size increased. The one exception was on Video 4, which had poor results overall for the HMM. SVM with a sliding window of 4 tied with Combination for the best average results; this indicates that SVM is a good machine learning technique for this problem, but that differencing is also quite useful.

4.3 Feature improvements

The preliminary results are promising but leave room for improvement. It would be better to more naturally incorporate the differencing technique into the system. Also, the skin features are very coarse, since they simply divide the frame into different sections regardless of the picture. Finally, I am not harnessing the conversational nature of the video. Moreover, the videos themselves involve a stationary background and a standing signer, which is not what the mobile phone videos will look like. In the following sections I describe my feature improvements and new tests.

4.3.1 Differencing and skin features

Given the positive results in the previous section, it is natural to incorporate differencing into the feature set for the SVM. This way, we can implicitly take advantage of the baseline technique, instead of using an ad hoc method.

In addition to differencing, I calculate more specific skin features. I detect the skin with the same Gaussian map as the previous section. After applying a 4x4 smoothing filter to the skin map, I use a Matlab library to gather data on the skin map. The library returns the connected components of the skin map, which correspond to blobs of skin in the original frame. I eliminate all blobs below a certain area, and find the area, center of gravity, and bounding box of the three biggest blobs. Figure 4.8 shows the center of gravity and bounding box of the three biggest blobs; I ignore the small blocks above and to the left of the main boxes. I order the blobs first based on vertical height, then rightmost and leftmost. Thus they correspond roughly to the face, the right hand, and the left hand, though often the skin-detection is noisy. When the user is not signing, or signing close to her torso, there is

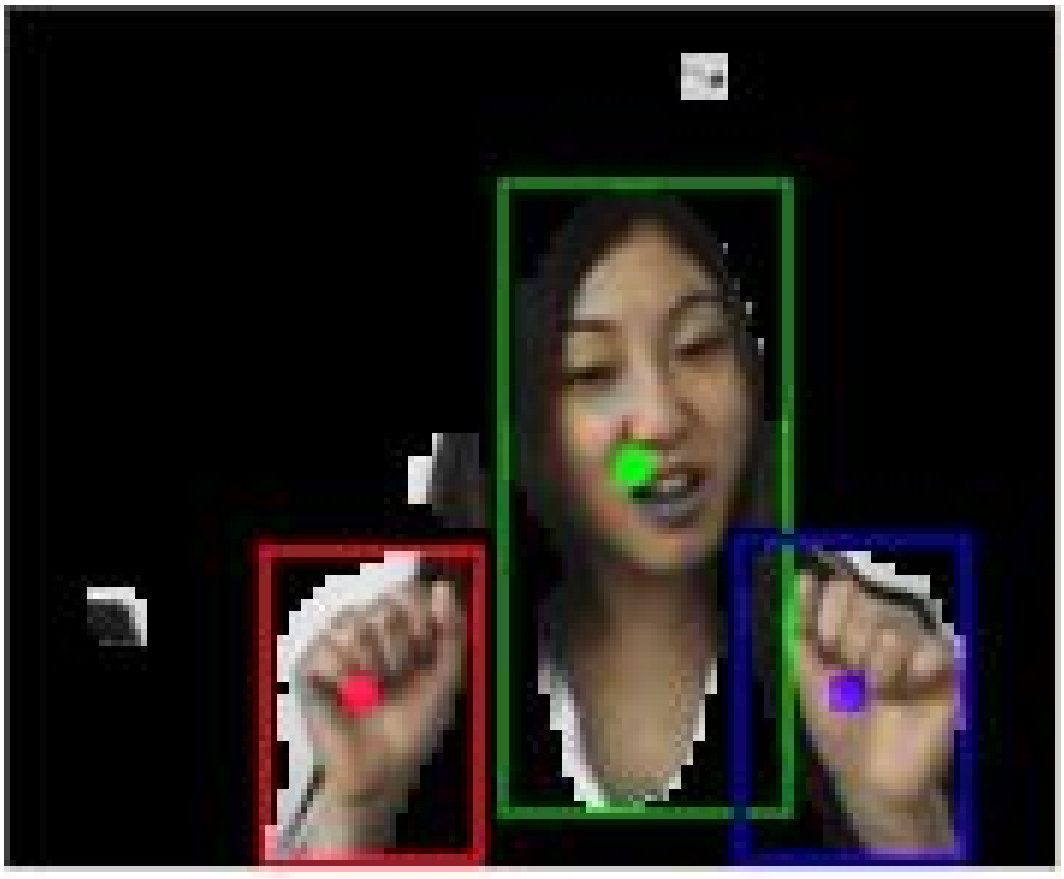


Figure 4.8: Visualization of the skin blobs.

Feature abbreviations	
sad	Sum of absolute differences
mv_x	x component of the summary motion vector
mv_y	y component of the summary motion vector
I	Number of I blocks
Following for each existing skin blob	
a	Area of skin
c_x	x component of the center of gravity of the skin
c_y	y component of the center of gravity of the skin
bb_x	x component of the bounding box of the skin
bb_y	y component of the bounding box of the skin
bb_w	Width of the bounding box of the skin
bb_h	Height of the bounding box of the skin

Table 4.2: Feature abbreviations

often no component corresponding to the left or right hands. In this case I send negative numbers, since that information is useful to the SVM classifier.

For the motion vectors, instead of dividing the frame into three pieces, I calculate an overall summary motion vector in the x and y directions. This makes the feature set for one frame a tuple of 25:

$$\{sad, mv_x, mv_y, I,$$

$$a^1, c_x^1, c_y^1, bb_x^1, bb_y^1, bb_w^1, bb_h^1,$$

$$a^2, c_x^2, c_y^2, bb_x^2, bb_y^2, bb_w^2, bb_h^2,$$

$$a^3, c_x^3, c_y^3, bb_x^3, bb_y^3, bb_w^3, bb_h^3\}$$

See Table 4.3.1 for the abbreviations.

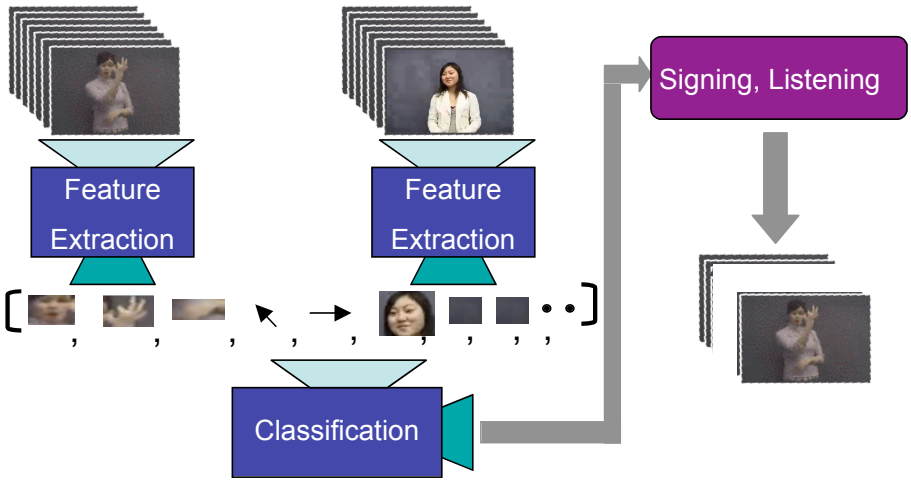


Figure 4.9: Activity recognition with joint information. Features are extracted from both sides of the conversation, but only used to classify one side.

4.3.2 Joint information

The conversational aspect of our videos allows me to incorporate additional information into my training and test sets. Namely, I am able to take features from both streams to aid in our classification. Suppose that two participants, Alice and Bob, are signing to each other over mobile phones. To classify Alice's next frame, I use the feature data from her previous frame plus the feature data from Bob's previous frame. Alice's features make up the first part of the vector and Bob's make up the second part, and I use Alice's label for training purposes. To classify Bob's next frame, I use the same data, except that Bob's features are in the first part of the vector and Alice's are in the second part, and I use Bob's label for training purposes. Thus, the feature vector is doubled in size. See Figure 4.9.

4.3.3 Recognition results Signing/Listening

I compare the results of the SVM on single stream and joint stream data to the simple baseline differencing technique. I determine the optimal threshold above which we will classify the frame as signing by training. I can combine data from both streams by simply subtracting the difference of the other stream from the difference of the current stream. Intuitively, this works because if the current user is signing and the other user is not signing, the joint difference will be large, but if the current user is not signing and the other user is signing, the joint difference will be small (negative, in fact). If both users are signing or not signing, the joint difference will be higher if the current user is signing more vigorously.

Recall the previous definition of $f(k)$ in Equation 4.2. Let $I_k(i,j)$ be the luminance component of pixel (i,j) at frame k in the primary side of the conversation, and $J_k(i,j)$ be the luminance component of pixel (i,j) at frame k in the secondary side of the conversation. Then

$$d_1(k) = \sum_{(i,j)\in I_k} |I_k(i,j) - I_{k-1}(i,j)| \tag{4.3}$$

$$d_2(k) = \sum_{(i,j)\in J_k} |J_k(i,j) - J_{k-1}(i,j)| \tag{4.4}$$

$$f(k) = \begin{cases} 1 & \text{if } \alpha d_1(k) - \beta d_2(k) > \tau \\ -1 & \text{otherwise} \end{cases} \tag{4.5}$$

I experimentally determined that values of $\alpha = 1$ and $\beta = 0.43$ resulted in the most accurate threshold τ.

I extracted features and trained and tested on eight conversational videos, from four different conversations. The conversational videos are the same as those used in the power study described in Section 4.1. I divided each video into four parts, trained on three out of the four, and tested on the fourth. I report the overall accuracy on the entire video using this leave-one-out testing method.

Table 4.3 shows that the single stream methods were all outperformed by the joint stream methods. Furthermore, my SVM technique, using the improved skin features, outperformed the baseline method.

Video		Single baseline	Joint baseline	Single SVM	Joint SVM
Noisy	Signer 1	36.3 %	85.5 %	**94.5** %	93.3 %
	Signer 2	65.1 %	85.9 %	94.7 %	**94.9** %
Quiet	Signer 1	76.9 %	82.6 %	91.6 %	**92.9** %
	Signer 2	52.3 %	83.2 %	90.3 %	**90.9** %
Noisy	Signer 3	63.2 %	85.7 %	**84.6** %	84.4 %
	Signer 4	77.8 %	89.1 %	90.5 %	**91.7** %
Quiet	Signer 3	49.8 %	80.4 %	86.3 %	**90.8** %
	Signer 4	46.0 %	85.4 %	**93.7** %	92.5 %
Weighted Average		58.9 %	84.6 %	90.7%	**91.4**%

Table 4.3: Recognition results for baseline versus SVM. The best for each row is in bold. The average is weighted over the length of video.

While the results are promising, this study is necessarily preliminary. A robust, generalized system would have to cope with complex backgrounds, differences in skin tones, and a non-stationary camera. Furthermore, different signers have different styles. Some are more likely to have big, energetic gestures and lots of movement even while not signing, whereas others are "quieter." There are also dialect variations from region to region, similar to accents in spoken languages. A truly robust system would have to cope with all these concerns.

Moreover, the above methods, while real-time, have not been implemented on the phone itself; nor has the power savings of been quantified when changing between different frame rates. The next chapter describes the variable frame rate on the phone itself, including the power savings, the challenges posed by the lack of processing power, and the classification accuracy of the reduced techniques.

Chapter 5

PHONE IMPLEMENTATION

The preceding chapters detail techniques for overcoming the challenges of low bandwidth and low processing power while producing intelligible sign language video. All of the methods thus far, though designed to be real-time and work efficiently, have not been implemented on the phone itself.

There are two major limiting factors for MobileASL on the phone: the processing power and the bit rate. My goal is to save processing power (and thus power overall) by varying the frame rate during periods of not signing, but my methods cannot be so computationally expensive that they affect the ability of the processor to maintain a reasonable frame rate. Initially, this turned out to be quite limiting, because the processor on the phones is only capable of 7-8 fps during periods of signing for full QCIF-sized video (176×144). Thus, there is no "wiggle room"; my methods must essentially be free in terms of processor power.

I address the bit rate by using the state-of-the-art video encoder, and by focusing bits on the face and hands of the user. Much of the grammar of sign language is found in the face [110]. I encode the skin at higher quality at the expense of the rest of the frame.

In the following sections, I detail the power savings obtained on the phone when employing our variable frame rate technique, and then describe compromises made to produce a phone implementation of the variable frame rate and the skin-based encoding.

5.1 Power savings on phone

MobileASL uses HTC TyTN-II phones (Windows Mobile 6.1, Qualcomm MSM7200, 400 MHz ARM processor), chosen because they have a front camera on the same side as the screen. The video size is QCIF (176×144). In order to verify the power savings on the phone, I simulated a sign language conversation and monitored the power usage for an hour on two phones with the variable frame rate on and with the variable frame rate off. The

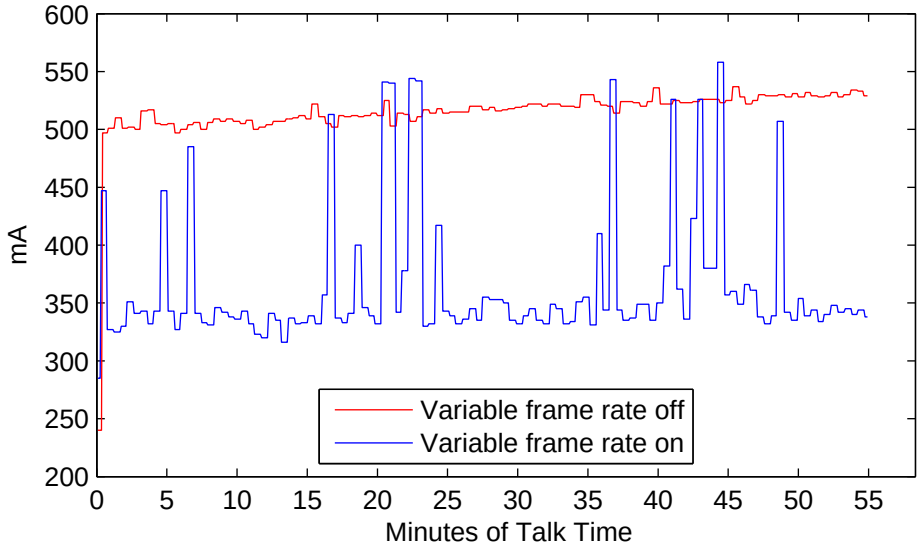


Figure 5.1: Snap shot of the power draw with variable frame rate off and on.

simulated conversation consisted of motion resulting in the higher frame rate every other minute, as though each person were signing for a minute, then listening for a minute, and so on. From observations of users on the phones, this is a reasonable scenario.

Figure 5.1 shows a snap shot of the power draw when the phone utilizes a variable frame rate versus when it does not. The power draw dips when the frame rate is lowered, due to the smaller processing power required to encode and transmit at 1 fps. Over an hour of talk time, the average power draw is 32% less with variable frame rate on than with it off.

In terms of battery life, the power savings is dramatic. Over the course of an hour, phone A lost 39% of battery life without the variable frame rate, versus 25% when the variable frame rate was on. Similarly, phone B lost 36% in regular mode and 26% with variable frame rate on. Regression analysis shows that the rate of loss over time for battery life on the phones is linear, with correlation coefficients of greater than 0.99. The average slope of the power drain on the battery every 5 seconds with the variable frame rate off is -0.0574, versus -0.0391 with it on. This corresponds to 68 extra minutes of talk time, or a 47% power gain over the battery life of the phone (see Figure 5.2).

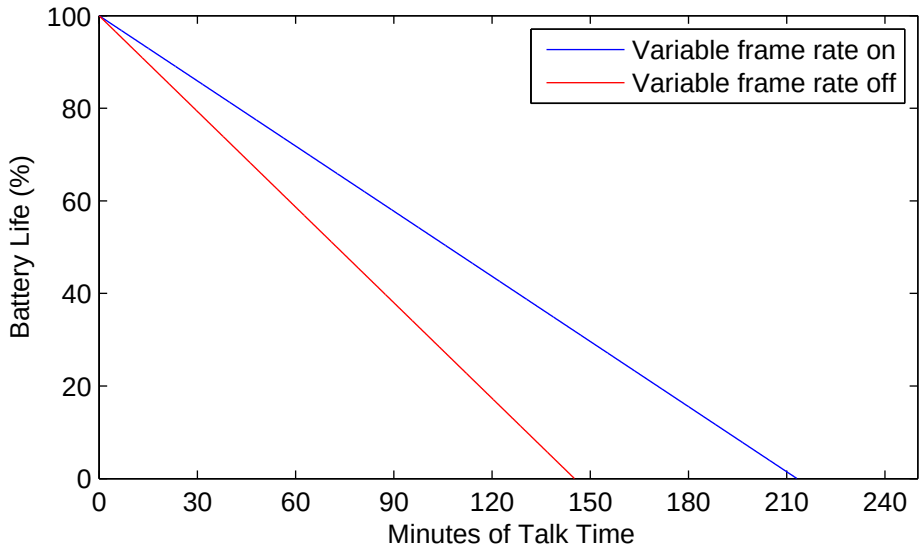


Figure 5.2: Battery drain with variable frame rate off and on. Using the variable frame rate yields an additional 68 minutes of talk time.

5.2 Variable frame rate on phone

5.2.1 Assembly considerations

H.264 is the current state-of-the-art video compression standard (MPEG Part 10)[118]. The MobileASL implementation is based on the Open Source x264 H.264 codec [3, 68]. The x264 encoder was compared with the JM reference encoder (ver 10.2) [70] and was shown to be 50 times faster, while providing bit rates within 5% for the same PSNR [68]. This makes it a good choice for H.264 video compression on low-power devices.

There are two ways to increase the processing speed of compression. The team performed assembly optimization using the ARMv6 single instruction multiple data assembly set, and converted the most computationally intensive operations into assembly. The lowest possible x264 settings are also used, changing the code when necessary (see Table 5.1). Even with these settings, the phones are only able to encode at a maximum rate of 7-8 fps for QCIF-sized video; the bottleneck in this case is not the bandwidth, but the processor. Later, the team achieved a speed-up in frame rate by down-sampling the video by 4 before compressing.

However, for the user study described in the next chapter, the processor limitations of encoding QCIF video had not yet been overcome.

5.2.2 Differencing

In previous chapters, I used advanced feature extraction and machine learning techniques to determine the class of the frame. However, I also found that a baseline differencing method performed quite well, with recognition rates averaging 84.6% versus 91.4% for SVM. Given the processor constraints on QCIF-sized video, I employ differencing to distinguish signing from not signing. I calculate the sum of absolute differences between successive raw frames:

$$d(k) = \sum_{i,j \in I(k)} |I_k(i,j) - I_{k-1}(i,j)|$$

I then check this value against a previously determined threshold τ arrived at by training on conversational sign language video. If $d(k) > \tau$, the frame is classified as signing. The videos originally used for training are those described in Chapter 4, and τ was determined to be $70,006$.

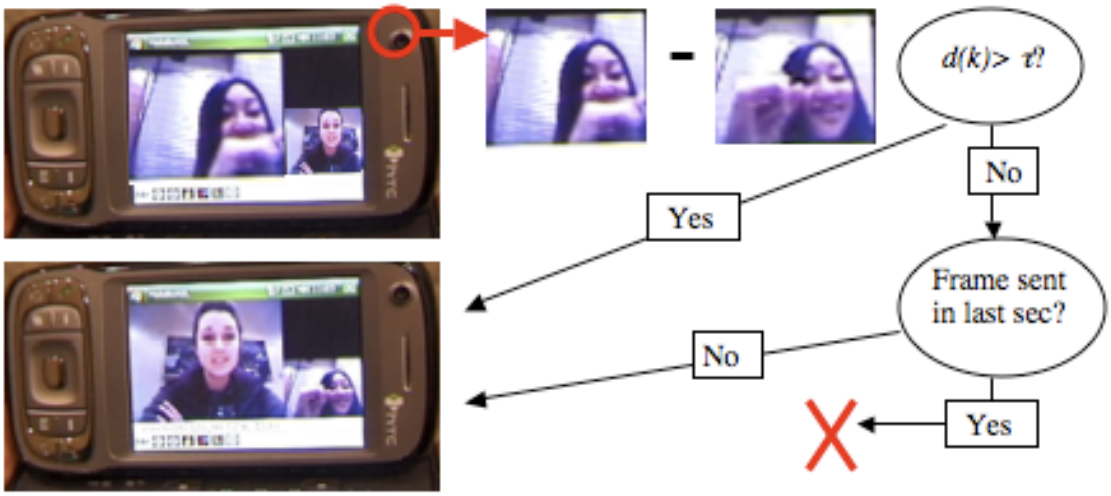


Figure 5.3: The variable frame rate architecture. After grabbing the frame from the camera, we determine the sum of absolute differences, $d(k)$. If this is greater than the threshold τ, we send the frame; otherwise, we only send the frame as needed to maintain 1 fps.

Figure 5.3 shows the architecture of my variable frame rate as implemented on the phone

Assembler optimizations	
usada8	Sum of absolute differences and accumulate
uqsub8	Quad 8-bit unsigned saturating subtraction
smulbb	Two signed 16-bit multiply
qsubaddx	16-bit unsigned saturating subtract and add with exchange
pkhbt	Pack halfword bottom top
x264 parameters	
–bitrate 30	Bit rates 30 kbps
–media	Integer pixel motion estimation: diamond search, radius 1
–8x8dct	8x8 DCT
–direct spatial	Direct MV prediction mode
–ref 1	One reference frame
–trellis 0	Trellis RD quantization disabled
–no-cabac	No CABAC
–bframes 0	No B-frames
–no-b-adapt	Adaptive B-frame decision disabled
-A none	16x16 inter motion vectors only
direct code change	No subpixel motion estimation and partition
direct code change	16x16 intra motion vectors only

Table 5.1: Assembler and x264 settings for maximum compression at low processing speed.

for the user study described in the next chapter. If the classifier determines that the user is signing, the frame is encoded and sent. If the classifier determines that the user is not signing, the frame is only sent to maintain a 1 fps rate.

5.2.3 Joint differencing with linear programming

I also implemented the joint differencing technique described in Chapter 4. Recall that I trained for the best τ according to Equation 4.5, but the parameters α and β were determined experimentally. I would like a more rigorous way to use the sum of absolute differences from both sides of the conversation to classify the frame. I pose the problem as a linear program and use Matlab's built-in Simplex algorithm to optimize my parameters.

I want to choose α, β, and τ such that the following is true as much as possible:

$\alpha d_1(k) - \beta d_2(k) > \tau$ when k is a signing frame, $\alpha d_1(k) - \beta d_2(k) \leq \tau$ when k is not.

Let $C = \{c_1, ..., c_n\}$ be a vector of indicator variables where 1 indicates signing and -1 indicates not signing, $c_k \in \{-1, 1\}$. Then:

$$\begin{aligned} \alpha d_1(k) - \beta d_2(k) > &\quad \tau \quad \forall k | c_k = 1 \\ \alpha d_1(k) - \beta d_2(k) \leq &\quad \tau \quad \forall k | c_k = -1 \end{aligned}$$

Assume τ is positive. Let $\mu = \alpha/\tau$ and $\gamma = \beta/\tau$. Write:

$$\begin{aligned} \mu d_1(k) - \gamma d_2(k) > &\quad 1 \quad \forall k | c_k = 1 \\ \mu d_1(k) - \gamma d_2(k) \leq &\quad 1 \quad \forall k | c_k = -1 \end{aligned}$$

This is equivalent to:

$$\begin{aligned} -\mu d_1(k) + \gamma d_2(k) \leq &\quad -1 \quad \forall k | c_k = 1 \\ \mu d_1(k) - \gamma d_2(k) \leq &\quad 1 \quad \forall k | c_k = -1 \end{aligned}$$

Thus the training problem is to choose μ and γ so that

$$-\mu d_1(k) c_k + \gamma d_2(k) c_k \leq -c_k \tag{5.1}$$

is true as much as possible. The optimal solution would minimize the number of k for which Equation 5.1 is not true. To approximate this, I subtract an error term per frame and minimize the sum. The linear program is:

$$\min \sum_{k=1}^{n} \epsilon_k$$

subject to

$$\begin{aligned} -\mu d_1(1)c_1 + \gamma d_2(1)c_1 - \epsilon_1 &\leq -c_1 \\ -\mu d_1(2)c_2 + \gamma d_2(2)c_2 - \epsilon_2 &\leq -c_2 \\ \vdots &\quad \vdots \\ -\mu d_1(n)c_n + \gamma d_2(n)c_n - \epsilon_n &\leq -c_n \\ \mu, \gamma, \epsilon_k &\geq 0 \end{aligned}$$

The variables in the linear program are μ, γ, and $\epsilon_k, 1 \leq k \leq n$. I normalize the $d_1(k)$ and $d_2(k)$ so that they are between 0 and 1 and run Simplex to find the settings for μ and γ that minimize the error. The classification of an unknown frame p is "signing" if $-\mu d_1(p) + \gamma d_2(p) \leq -1$ and "listening" otherwise.

Though this is not an optimal protocol, in practice the error values ϵ_k are quite small. Figure 5.4 shows a histogram of ϵ values. The majority, over 66 percent of the total, are equal to 0, and the mean is 0.19.

5.2.4 SVM

There are several challenges to making the support vector classifier work on the phones. As stated in earlier chapters, the feature extraction and classification is real-time and designed to work well on the phones themselves, but the processor is a major bottleneck - much more of one than originally expected. Before, the skin features were extracted with Matlab code that automatically found the connected components. Recall that the features for one frame are in a tuple of 25 (abbreviations defined in Table 4.3.1):

$$\{sad, mv_x, mv_y, I,$$

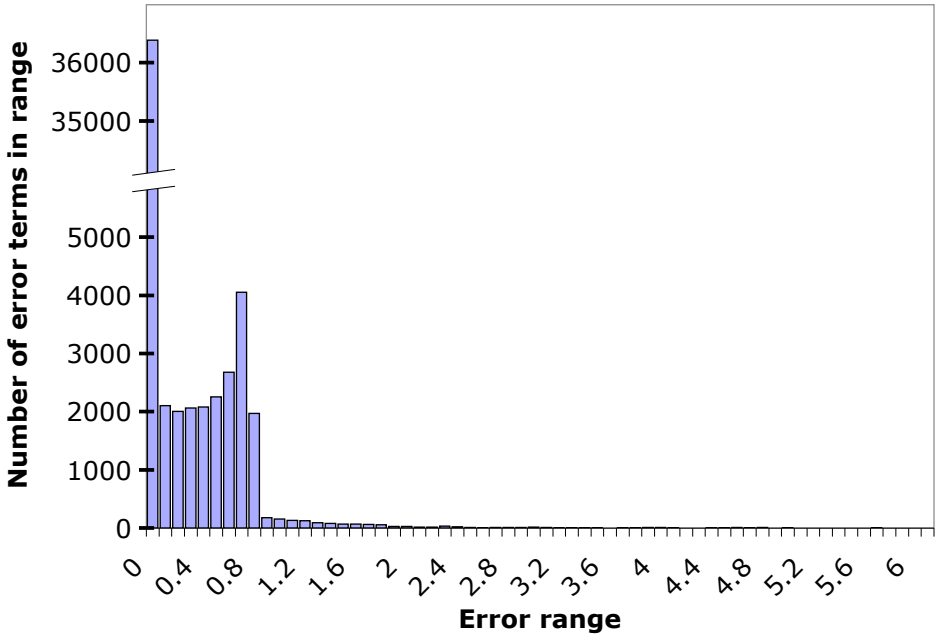


Figure 5.4: Histogram graph of the number of error ϵ_k terms with certain values. The vast majority are 0.

$$a^1, c_x^1, c_y^1, bb_x^1, bb_y^1, bb_w^1, bb_h^1,$$
$$a^2, c_x^2, c_y^2, bb_x^2, bb_y^2, bb_w^2, bb_h^2,$$
$$a^3, c_x^3, c_y^3, bb_x^3, bb_y^3, bb_w^3, bb_h^3\}$$

I implement differencing on the phone by buffering the previous image within the encoder and subtracting its luminance component from the current image's on a pixel-by-pixel basis. The next three features I obtain in the same way as before. For the skin features, I need to detect the skin rapidly, filter and threshold, and determine the three largest connected components and their area, center of gravity, and bounding box.

In the previous chapter I describe using a Gaussian skin map for the skin detection.

Though computationally cheap in general, on the phone the floating point operations are expensive enough to affect the frame rate. Thus, I modify the code so that the transformation is performed with only integer operations. Since the skin detection is ultimately a threshold decision, I can eliminate lower order bits without affecting the accuracy of the detection. I then apply a 4x4 averaging filter on the binary skin map, to eliminate small holes. I detect the connected components by using the classical two-pass labeling algorithm that employs a union-find data structure [92]. As I label the components, I keep track of their current area, center of gravity, and bounding box, so that after I've found the connected components, my task is finished; I do not need to iterate over the skin map again. This version of the feature extraction is cheap enough to not affect the frame rate of the encoding.

I posited that the salient feature might not be the area or centroid itself, but the change in it from the previous frame. Using the change in features rather than their absolute value increased classification accuracy, so this is the result I report.

Joint information, in the form of the pixel differences from the other side of the conversation, is transmitted in packets whether or not the frame itself is sent. Since that data is so small, transmitting it does not affect the bit rate or encoding time.

Though the feature extraction is not too expensive, the floating point operations are, and unfortunately the SVM relies heavily on floating point operations to classify the tuple (the tuple itself is scaled and normalized to be between -1 and 1, so all operations are necessarily floating point). The feature extraction and SVM are currently implemented on the phone, but they cause the encoder to process fewer frames per second, an unreasonable trade off given how low the frame rates are already.

5.3 Results

To test the classification accuracy of my algorithms, I captured YUV video from the phone camera and hand labeled the frames as signing or not. I recorded four conversations with six different users, for a total of 8 videos. I tested the accuracy of my methods in two ways: across all videos, training on seven and testing on the eighth (or inter-video); and within the videos, in the same manner as before (or intra-video). For intra-video, I divide the

video into four parts, train on three and test on the fourth. I do this for all four pieces and report the average. Inter-video training would make sense to have the most general baseline for the classifier, but it takes a long time and may overfit to characteristics of my training videos. More videos will help inter-video training increase in accuracy. Intra-video training is equivalent to a user having a training session on his or her phone. Since phones are personal devices, it would be natural to add this feature. I experimented with several different temporal windows, as in the previous chapter, and found the three frame window to be the best across all methods.

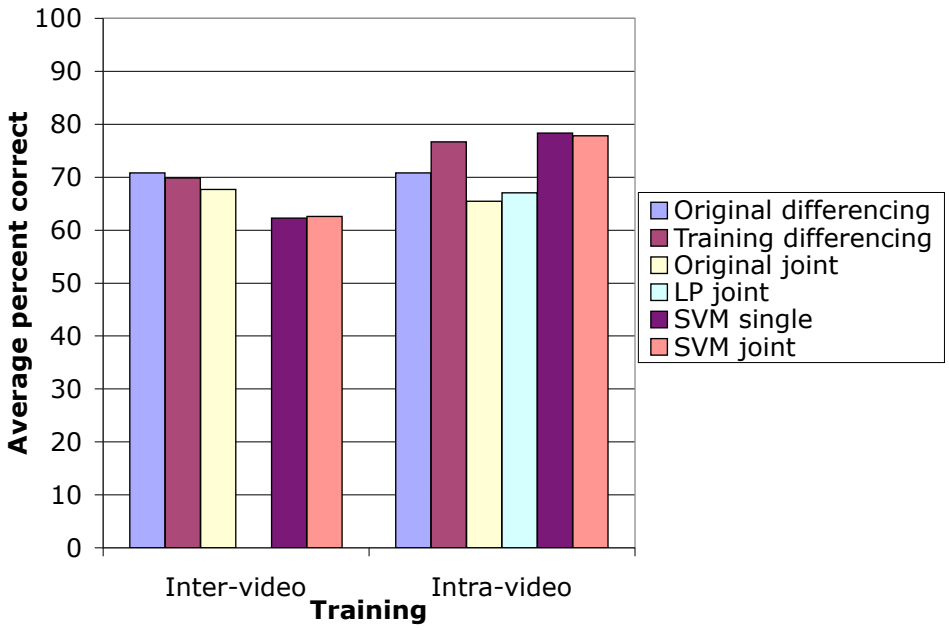


Figure 5.5: Comparison of classification accuracy on the phone of my methods.

Figure 5.5 displays a comparison of the average classification accuracy of the different methods. I show the average accuracy results of both inter- and intra-video training. For

inter-video training, the classifier was trained on seven of the videos and tested on the eighth; the overall average is shown. For intra-video training, each video was cross-tested by training on 3 parts of the video and testing on the fourth. These four tests were averaged for the result displayed.

The original differencing method used τ trained on non-phone videos and was the method in use for the study in the next chapter. The training differencing method trained on the phone data for the appropriate τ. It was not possible to inter-train for linear programming, since the number of variable became too large for the Simplex algorithm; but on intra-training, the joint linear programming method out performed the original joint method. However, using joint information does not result in higher classification accuracy in either the original (α, β) method or the new linear programming method. For SVM, using joint information helped when training between videos but did not for intra-video training, and the difference was small in both cases. The most salient feature appears to be the primary differencing; differencing from the second side of the conversation does not add much.

Overall, intra-video training outperformed inter-video training. Support vector machines performed the best, but only by a very small amount. The second best method was the simplest: training differencing. This is good news, since this method is feasible for phone implementation.

5.4 Skin Region-of-interest

Earlier studies with MobileASL have shown that signers tend to focus on the face of the person signing. One way to increase intelligibility while maintaining the same bit rate is to shift the bits around, so that more are focused on the face and less are focused on the background.

The H.264 encoder works by dividing the uncompressed frame in 16x16 pixel macroblocks. It then performs a series of operations to compress these macroblocks, essentially by finding parts of the previous frame that match the current frame. The encoder aims to produce the highest possible quality frame at a given bit rate (in our case, 30 kbps). The quality of each macroblock is determined by the quantizer step size, or QP. Lower QPs indicate higher quality but also require a higher bit rate.

I will encode skin macroblocks at a higher bit rate and non-skin macroblocks at a lower bit rate. To determine what macroblocks have a majority of skin pixels, I can use the Gaussian skin map from the previous section. However, because I am working at the level of macroblocks, I don't need to be as accurate as before and can thus employ an even simpler skin detection technique.

The image is captured in YUV format, where Y is the luminance component and U and V are the chrominance components. I examine the U and V values and determine if they are in the appropriate range for skin: U must be between and V must be between. I then check each 16x16 pixel macroblock and deem it skin if the majority of pixels in the block are skin. Figure 5.6 shows the skin pixel classification. This method works as well as the Gaussian and requires essentially zero computational power.

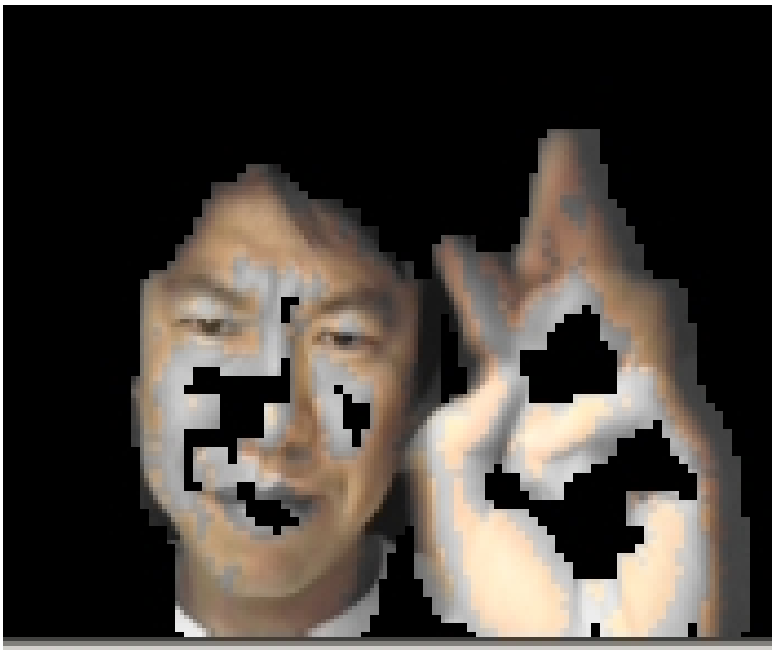


Figure 5.6: Skin-detected pixels as determined by our algorithm running on the phone.

I change the quality of the skin by adjusting the QP value for the skin macroblocks. In my experiments, ROI 0 corresponds to no reduction in quantizer step size; ROI 6 corresponds to a 6 step reduction in size; and ROI 12 corresponds to a 12 step reduction in size. Forced to

make the skin macroblocks of higher quality, the encoder must reduce the quality elsewhere in the frame to maintain the bit rate.

Figure 5.7 shows a comparison of ROI 0 (left) and ROI 12 (right). The face and hand are slightly clearer in the ROI 12 picture. However, in the ROI 0 picture there is a clear line between the shirt and the background around the shoulder area. In the ROI 12 picture, this line is smudged and blocky.

Figure 5.7: ROI 0 (left) and ROI 12 (right). Notice that the skin in the hand is clearer at ROI 12, but the background and shirt are far blurrier.

The dynamic skin ROI and variable frame rate are evaluated by users in the following chapter.

Chapter 6

USER STUDY ON PHONES

In this chapter, I describe an experiment that evaluated the user experience of our compression technique. I tested variable frame rate on and off together with dynamic skin ROI encoding at ROI 0, 6, and 12, for six different possible combinations. The order of the settings was changed between conversations according to a Latin Square. Differencing with τ trained on the earlier web camera videos was used to decide when to lower the frame rate. Down-sampling had not yet been implemented, and the frame rate of the phones was 7-8 fps.

Given that our application is mobile phone communication, I expect a variety of different conversations to take place between users. For example, a user might call an unfamiliar interpreter in order to reach his or her doctor. On the other hand, users will certainly call friends and family members. The conversations recorded represent this variety.

I gathered both subjective and objective measures. The subjective measures were obtained via a survey. For the objective measures, I wanted to see how the conversations were affected by our changes to the settings. I videotaped each conversation and analyzed the recording after.

6.1 Participants

Altogether, 15 participants fluent in ASL (age: 24-59, mean = 42, 5 male) recruited from the Seattle area took part in the study. For background information, they were asked how long they had known ASL; from whom they learned ASL; and which language they preferred for communication. See Table 6.1.

Eight participants preferred ASL for communication; four preferred English; and the remaining three chose both. Of the participants that chose English, three were interpreters with 7, 15, and 30 years experience, and one read lips but had known ASL for 21 years.

ID	*Time known*	*Taught by*	*Prefers*
P1	15+ years	colleagues	English
P2	7 years	teachers	English
P3	32 years	friends	ASL
P4	since birth	parents	ASL
P5	since birth	parents	ASL, English
P6	high school	friends/school	ASL
P7	21 years	friends	English
P8	32 years	friends/school	ASL
P9	since birth	parents	ASL
P10	30 years	friends/teachers	ASL
P11	since birth	parents	ASL
P12	since birth	parents	ASL
P13	34 years	friends/school	ASL
P14	20 years	friends	ASL, English
P15	38 years	friends/interpreters	ASL, English

Table 6.1: ASL background of participants

Five participants wore glasses for near-sightedness.

Nine separate conversations were recorded. Three conversations were with a research staff member fluent in ASL. Five of the conversations were between strangers and four were between people that knew each other well, including one husband/wife pair.

6.2 Apparatus

The participants sat on the same side of a table, separated by a screen. In the background was a black drape. The phones were on the table in front of them, and they were told to adjust their positioning and the phone location so that they were comfortable carrying on the conversation. See Figure 6.1.

Figure 6.1: Study setting. The participants sat on the same side of a table, with the phones in front of them.

6.3 Procedure

The participants were told to talk about whatever came to mind, and that they would be interrupted after five minutes and the settings changed on the phone. After each five minute period (subconversation), they filled out a questionnaire about the subconversation. Each

POST-VIDEO QUESTIONNAIRE

"Compression of American Sign Language Video"

During the video, how often did you have to guess about what the signer was saying?

not at all | 1/4 of the time | 1/2 the time | 3/4 of the time | all the time

On a scale from 1 to 5, how difficult would you say it was to comprehend the video? (where 1 is very easy and 5 is very difficult).

1 | 2 | 3 | 4 | 5

Changing the frame rate of the video can be distracting. How would you rate the annoyance level of the video? (where 1 is not annoying at all and 5 is extremely annoying).

1 | 2 | 3 | 4 | 5

The video quality over a cell phone is not as good as video quality when communicating via the Internet (i.e. by using a web cam) or over a set top box. However, cell phones are convenient since they are mobile. Given the quality of conversation you just experienced, how often are you to use the mobile phone for making calls versus just using your regular version of communication (go home to use the Internet or set top box, or just text)?

always use phone | mostly use phone | use phone 1/2 the time | mostly wouldn't use phone | never use phone

If video cell phones were available today with this quality of transmission, would you use them?

definitely | probably | maybe | probably not | definitely not

Figure 6.2: Study questionnaire for subjective measures.

conversation was video taped, and objective measures calculated on the recording.

6.3.1 Subjective Measures

The survey questions are shown in Figure 6.2. The participants were asked to subjectively rate the quality of the video, measured in how hard or easy it was to understand. The fourth question was poorly worded and often had to be explained. I was trying to capture the trade-off in the convenience of the mobile phone versus its quality. The video quality over a phone is not as good as video quality when communicating via the Internet (e.g., by using a web cam) or over a set top box. However, phones are convenient since they are mobile. In light of the quality of conversation the participants experienced during the study, I asked them to rate how often they would use the mobile phone for making video calls versus just using their other pre-existing modes of communication.

6.3.2 Objective Measures

My goal was to measure the comprehensibility of the conversation. A confusing speaking conversation might contain a lot of requests for repetitions (called *repair requests* [108]) or *conversational breakdowns*, where one person says "I can't hear you." In sign language, there is an additional feature to measure, which is finger spelling. Finger spelling is when someone spells out the name of something, and occurs mainly with proper names, titles, and technical words. However, some finger spelled words are lexicalized "loan signs," common words whose sign has become the stylized finger spelling (e.g., "bus," "back"). Since these act as regular signs, I do not count them in our finger spelling measure.

The objective measures were number of repair requests, average number of turns associated with repair requests, number of conversational breakdowns, and speed of finger spelling. These were all calculated with the help of a fluent ASL speaker. In sign language conversation, a repair request may mean forming the sign for "again" or "what?," or finger spelling in unison with the conversation partner. For each repair request, I counted the number of turns until the concept was understood; this is the number of times the requester had to ask for repetition before moving on. Conversational breakdowns were calculated as

the number of times the participant signed the equivalent of "I can't see you" (e.g. "frozen," "blurry," "choppy"). If repetition requests were made but never resolved, I counted it as a conversational breakdown. Finally, I measured the time it took to sign each finger spelled word and divided by the number of characters in that word, resulting in characters per second.

6.4 Study Results

The results of the study were statistically significant for only two of the subjective measures. VFR affected the objective measures but ROI did not. There was no significant ROI*VFR interaction for any measure.

Question	ROI		VFR		Interaction	
	$\chi^2(2, N = 90)$	p	$\chi^2(1, N = 90)$	p	$\chi^2(2, N = 90)$	p
Guesses	11.11	$< 0.01^{***}$	4.44	$< 0.05^{**}$	0.78	0.68
Comprehension	5.33	0.07^{*}	2.87	0.091^{*}	2.75	0.25
Annoyance	3.07	0.22	0.79	0.37	2.26	0.32
Phone vs. Other	0.12	0.94	1.1	0.29	0.18	0.91
Would use	0.42	0.81	0.22	0.64	1.02	0.6

Table 6.2: Statistical analysis for the subjective measures questionnaire (see Figure 6.2). Statistical significance: *** = $p < 0.01$, ** = $p < 0.05$, * = $p < 0.10$.

6.4.1 Likert Scale Subjective Measures

Table 6.2 contains the χ^2 test and significance values for the five questions. Only the first two questions in the questionnaire yielded statistically significant results. The interaction results were all non-significant, indicating levels of ROI and VFR did not disproportionately affect one another.

The ROI had a significant effect on participants' Likert responses for how often they had to guess, with 1=not at all, 5=all the time. It had only a marginal effect on participants'

Likert responses for how difficult the video was to comprehend, with 1=very easy, 5=very difficult. Figures 6.3 and 6.4 show the means and standard deviation of their responses. For guessing, a Wilcoxon signed-rank test shows that ROI 0 and ROI 6 were not significantly different ($z = 0.50, p = 1.00$), but that ROI 0 and ROI 12 were different ($z = 35.00, p < .01$) and ROI 6 and ROI 12 were also different ($z = 35.50, p < .05$). Thus, perceptions of guessing frequency decreased when region-of-interest coding reached 12 from 0 and 6. There was also a trend towards easier comprehension as ROI increased, but these differences were statistically non-significant ($p < 0.10$). The variable frame rate increased perceptions of guessing frequency and decreased perception of comprehension. The effect was only marginal for comprehension.

The ROI and VFR did not cause a detectable difference in participants' Likert responses for how annoyed they were at the level of frame rate, how often they would prefer the phone to some other means of communication, or their potential future use of the technology. The overall means for preference of the phone and potential future use were 2.98 and 2.47, respectively, where 1 meant the participant would definitely use the phone and 5 meant the user definitely would not use the phone.

6.4.2 Quantitative Objective Measures

Question	ROI		VFR		Interaction	
	$\chi^2(2, N = 90)$	p	$\chi^2(1, N = 90)$	p	$\chi^2(2, N = 90)$	p
Repair requests	2.66	.26	5.37	$< 0.05^{**}$	1.99	.37
Number of turns	0.94	.62	4.01	$< 0.05^{**}$	0.96	.62
Breakdowns	3.38	.18	7.82	$< 0.01^{***}$	1.51	.47
	F(2,28)	p	F(1,14)	p	F(2,28)	p
Finger spelling speed	0.42	.66	0.19	.67	0.21	.81

Table 6.3: Statistical analysis for the objective measures. Statistical significance: *** = $p < 0.01$, ** = $p < 0.05$. ROI was not statistically significant, nor was the interaction. Finger spelling speed was amenable to ANOVA and was not statistically significant.

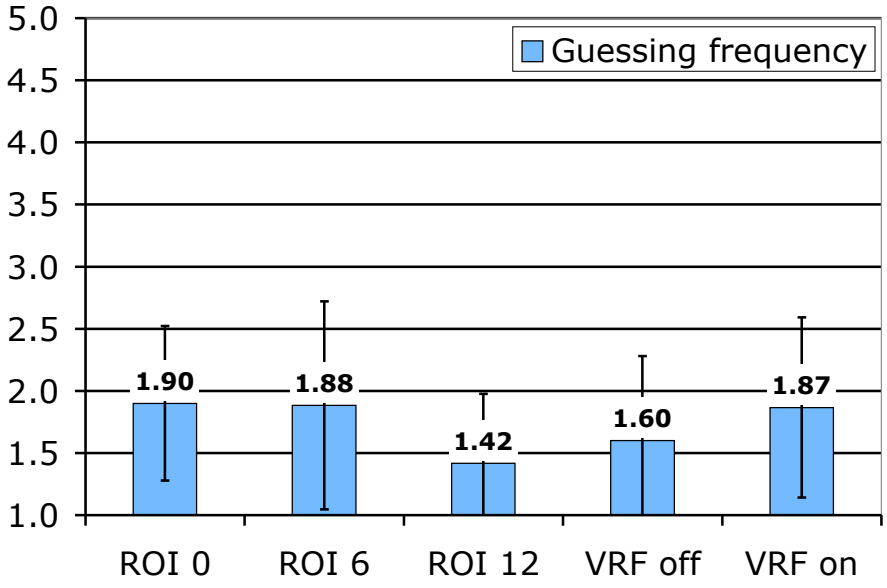


Figure 6.3: Subjective measures on region of interest (ROI) and variable frame rate (VRF). The participants were asked "How often did you have to guess?," where 1=not at all and 5=all the time.

Table 6.3 contains the statistical results for the objective measures. Repair requests, number of turns before a concept was understood, and conversational breakdowns were all affected by VFR but not by the ROI. Speed of finger spelling was not affected by either, and the interaction between ROI and VFR was not statistically significant.

The number of repair requests was highly skewed and according to a significant Shapiro-Wilk test ($W = 0.74, p < .0001$), not amenable to ANOVA. This was also true of the number of repetitions that transpired before the concept was understood ($W = 0.76, p < .0001$) and the number of conversational breakdowns ($W = 0.44, p < .0001$). Typical transformations were not an option, so I continued to employ ordinal logistic regression as I had for our Likert data, which showed an appropriate fit for all three measures. For finger spelling

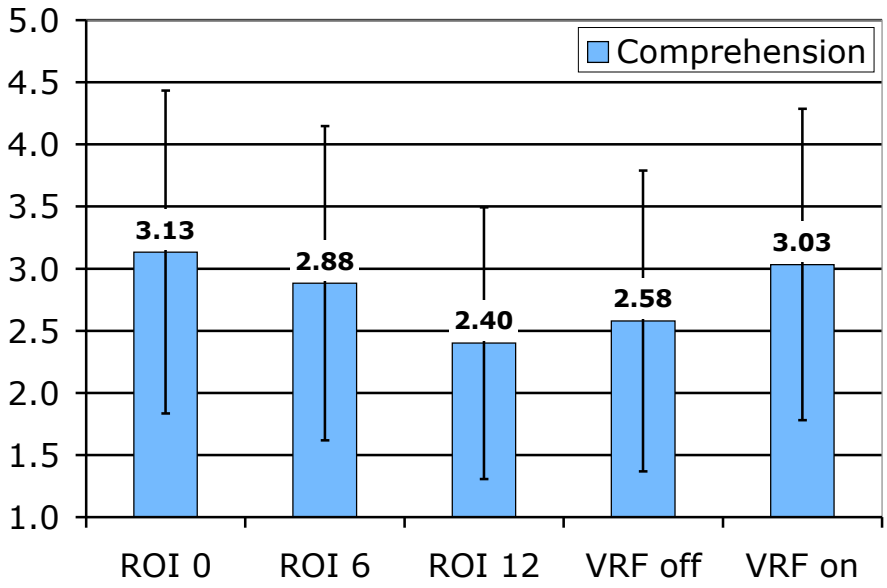


Figure 6.4: Subjective measures on region of interest (ROI) and variable frame rate (VRF). The participants were asked "How difficult was it to comprehend the video?," where 1=very easy and 5=very difficult.

speed, a non-significant Shapiro-Wilk test (W=0.98, p=.12) confirms these data are normal and suited to analysis with a repeated measures ANOVA. Finger spelling speed was not affected by ROI or VFR, and the mean finger spelling speed for all conditions was 3.28 characters per second.

The means and standard deviations for the significant objective measures are in Figure 6.5. VFR negatively affects the number of repeats, the number of repetitions, and the number of conversational breakdowns.

6.4.3 Participant Comments

Nearly all of the participants asked when the technology would be available for their use. They expressed disappointment that the software is not ready for wide distribution.

Several participants commented on the awkward angle of the camera when the phone is on the table. They separately suggested creating a stand so that the phone could be at the same level as the face.

Other participants disliked the eye strain caused by looking at the small screen. Participants 8 and 9 in particular were affected by an over bright room that made the LCD very difficult to see. They commented that they would only ever use the phone for emergencies or very short conversations.

Two of the interpreter participants separately commented on the speed of finger spelling. They noted that they were finger spelling at a pace unnaturally slower than usual and said this reminded them of video-relay interpreting. Video-relay interpreting occurs over a much higher bandwidth connection than the mobile phone, though it sometimes has connection problems.

6.5 Discussion

Our participants perceived guessing less frequently and understanding more of the conversation at a higher ROI (see Figure 6.3 and 6.4). ROI otherwise had no statistically significant effect on the participants. Recall that a high ROI encodes the skin at a higher quality at the expense of the rest of the frame; this means there is no extra cost to the encoder in terms of bandwidth. Since our algorithm is a simple range detection query, there is no extra burden on the processor. Thus, using a high ROI is a good way to save system resources and increase intelligibility.

The results on variable frame rate are more mixed. I expect variable frame rate to lead to some degradation in quality, since I am leaving out a lot of information in order to save resources. Indeed, participants perceived guessing more frequently and understanding less of the conversation (Figure 6.3 and 6.4). Moreover, their conversations were objectively affected. They made more repair requests, took more turns to correct those requests, and

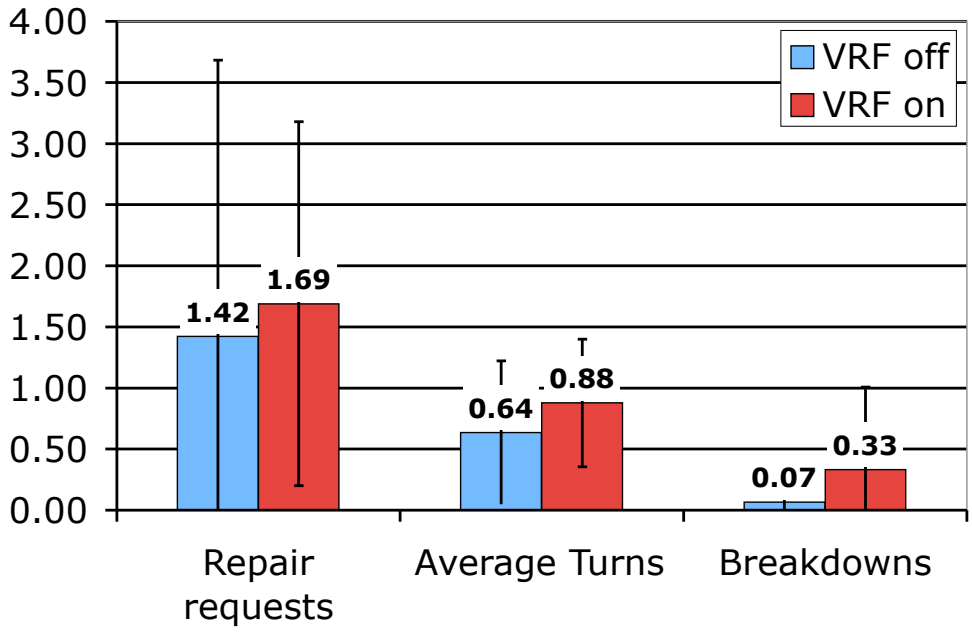


Figure 6.5: Objective measures: the number of repair requests, the average number of turns to correct a repair request, and the conversational breakdowns.

had more conversational breakdowns when variable frame rate was on (Figure 6.5). However, an examination of the means shows that overall, they were not making many repair requests or experiencing many conversational breakdowns. Even though there were relatively many more breakdowns with the variable frame rate, breakdowns still only occurred once every third conversation on average.

The results on three of the subjective measures were encouraging. The variable frame rate did not appear to affect participants' likelihood of using the phone over other means of communication or their overall adoption likelihood. Nor did it affect their perceived irritation with the frame rate changes. Because variable frame rate saves considerable

system resources, I expect it to affect conversations; it is encouraging that this does not mean users perceive that they are less likely to adopt the technology.

The results on finger spelling were surprising. Given the other objective measure results, I expected finger spelling to be measurably slower, but I saw no statistically significant result. It may be that the participants spelled more slowly overall and not just during the variable frame rate subconversations. However, when analyzing videos it seemed that conversational breakdowns occurred most often when one participant was finger spelling. I suspect this is because the differencing method would incorrectly label the frames and lower the frame rate, resulting in a "frozen" image.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Future Work

Technically speaking, several challenges remain. Most of these are generally about improving classification accuracy, exploring power use, and understanding the trade off.

7.1.1 Finger spelling activity recognition

We would like to further investigate finger spelling. Using our method developed for the variable frame rate, we want to automatically recognize finger spelling, so that we do not lower the frame rate during important periods of the video. It would also be interesting to know how using the mobile phone affects finger spelling as compared to other methods of video communication, such as video relay service.

7.1.2 Training on phone, per user

The classification accuracy was much better on the phone when the videos were intra-trained. That is, when the signer is known, the algorithm can more accurately recognize signing and not signing. A simple graphical user interface could be designed in which the user would be directed to start signing and stop signing. This would be used as the ground truth training for the classification algorithm. Open questions are: how long does the training have to go on for? how much would it improve classification accuracy? would users mind the training? how much time would they be willing to spend on it?

7.1.3 Skin improvements

A number of different engineering projects remain for the skin ROI. Currently we only use range testing. How accurate is it? Could we make our own Gaussian based on the phone videos? Would users notice the difference between our skin ROI and an extremely accurate

version? It would also be interesting to subjectively measure ROI without sign language, as in regular video conferencing, and see if users found that it improves the video.

7.1.4 Power exploration

We currently do not know where the power savings is coming from: the encoding or the sending (or what combination of the two). One idea that a team member is exploring is to reduce the spatial resolution during periods of not signing instead of the temporal resolution. We should see if this still leads to a power gain, and also explore what users think of the two different methods. Also, to accurately use the motion vectors, we need to encode the video even when we will not send it. This might result in no power gain. We should experiment with testing the power savings when the SVM is used with different features and see what the power/accuracy trade off is.

7.1.5 Small improvements

We should try two versions of a threshold, one for a "noisy" environment and one for a "quiet" one, which could be determined by the user. The threshold itself would be trained with noisy and quiet data. We should also implement code that fixes the "jitter" problem; that is, when the user holds the phone in one hand, we need to compensate for the movement of the camera itself. We should try to improve the features, by investigating using pixel differences per skin block instead of over the whole video. We should also find out which features are most salient to the classifier, by leaving some out and training and testing on reduced feature sets.

We currently do not know which frames are being misclassified. It is much worse for a signing frame to be misclassified as not signing than vice-versa; our current methods do not penalize for this. Again, there is a trade off, because too much penalty will result in no variable frame rate, but too little penalty means that important information is lost. We could test this in a lab study where we keep track of accuracy versus power and have users subjectively rate their conversations.

7.1.6 Modeling sign language

The classifier is currently looking at movement and not the deeper question of sign language versus movement. For example, it will misclassify a user drinking a cup of coffee as signing. We could model sign language in the same that speech is modeled, to create a more accurate picture of what signing is and use that in our classifier. One way to start would be by looking at the video data in the frequency domain and seeing if patterns emerge during sign language that don't during other movement.

7.1.7 Technology transfer

In the future, we will continue to improve MobileASL so that we may make it widely available. Our next step is to move out of the lab and into the field. We plan to give participants phones with MobileASL installed and have them use and comment on the technology over an extended period of time.

7.2 Conclusion

In this work, I described my sign language sensitive algorithms for aiding in the compression and transmission of sign language video. I create techniques that save system resources by focusing on the important parts of the video. I implement my methods on an off-the-shelf mobile phone. I then evaluate my techniques in a user study in which participants carry on unconstrained conversation over the phones in a laboratory setting. My results show that I can save processing power and bandwidth without negatively affecting the participants' likelihood of adopting the technology.

The most common question asked by my participants was "when will this be available?" When recruiting for my study, I received interested queries from all over the United States. My thesis brings us closer to the ultimate goal of our MobileASL project: to provide Deaf people full access to today's mobile telecommunication network.

BIBLIOGRAPHY

[1] acbPocketSoft. acbTaskMan Version 1.3, April 2007. http://www.acbpocketsoft.com.

[2] D. Agrafiotis, C. N. Canagarajah, D. R. Bull, M. Dye, H. Twyford, J. Kyle, and J. T. Chung-How. Optimized sign language video coding based on eye-tracking analysis. In *VCIP*, pages 1244–1252, 2003.

[3] L. Aimar, L. Merritt, E. Petit, M. Chen, J. Clay, M. R., C. Heine, and A. Izvorski. x264 - a free h264/AVC encoder. `http://www.videolan.org/x264.html`, 2005.

[4] S. Akyol and U. Canzler. An information terminal using vision based sign language recognition. In *ITEA Workshop on Virtual Home Environments*, pages 61–68, 2002.

[5] M. Assan and K. Grobel. Video-based sign language recognition using hidden markov models. In *Proceedings of Gesture Workshop*, pages 97–109, 1997.

[6] A. Bar-Noy, R. E. Ladner, and T. Tamir. Scheduling techniques for media-on-demand. In *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA '03)*, pages 791–800, 2003.

[7] A. Bar-Noy, R. E. Ladner, T. Tamir, and T. VanDeGrift. Windows scheduling of arbitrary length jobs on parallel machines. In *Proc. of the 17th annual ACM symposium on Parallelism in algorithms and architectures*, pages 56–65, 2005.

[8] B. Bauer and K.-F. Kraiss. Video-based sign recognition using self-organizing subunits. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 434–437, 2002.

[9] D. Bavelier, A. Tomann, C. Hutton, T. Mitchell, D. Corina, G. Liu, and H. Neville. Visual attention to the periphery is enhanced in congenitally deaf individuals. *The Journal of Neuroscience*, 20(RC93):1–6, 2000.

[10] R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *European Conference on Computer Vision*, volume 1, pages 390–401, 2004.

[11] A. Buzo, A. Gray Jr., R. Gray, and J. Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(5):562–574, October 1980.

[12] A. Cavender, R. E. Ladner, and E. A. Riskin. MobileASL: Intelligibility of sign language video as constrained by mobile phone technology. In *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 71–78, New York, NY, USA, 2006. ACM Press.

[13] A. Cavender, R. Vanam, D. Barney, R. Ladner, and E. Riskin. Mobileasl: Intelligibility of sign language video over mobile phones. In *Disability and Rehabilitation: Assistive Technology*, pages 1–13, London, June 2007. Taylor and Francis.

[14] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[15] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image Vision Computing*, 21(8):745–758, August 2003.

[16] N. Cherniavsky, A. C. Cavender, R. E. Ladner, and E. A. Riskin. Variable frame rate for low power mobile sign language communication. In *Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, New York, NY, USA, 2007. ACM Press.

[17] N. Cherniavsky, R. E. Ladner, and E. A. Riskin. Activity detection in conversational sign language video for mobile telecommunication. In *Proceedings of the 8th international IEEE conference on Automatic Face and Gesture Recognition*. IEEE Computer Society, Sept 2008.

[18] F. Ciaramello and S. Hemami. 'Can you see me now?' An objective metric for predicting intelligibility of compressed American Sign Language video. In *Human Vision and Electronic Imaging 2007*, January 2007.

[19] F. Ciaramello and S. Hemami. Complexity constrained rate-distortion optimization of sign language video using an objective intelligibility metric. In *Proceedings of SPIE Vol. 6822, Visual Communication and Image Processing 2008*, San Jose, CA, January 2008.

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[21] Y. Cui and J. J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, May 2000.

[22] I. Dalgic and F. A. Tobagi. Characterization of quality and traffic for various video encoding schemes and various encoder control schemes. Technical Report CSL-TR-96-701, Stanford University, 1996.

[23] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand server with batching. In *Proc. of ACM Multimedia*, pages 15–23, 1994.

[24] A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic batching policies for an on-demand video server. *ACM Multimedia Systems Journal*, 4(3):112–121, 1996.

[25] D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for on-demand data delivery. In *Proc. of the 7th ACM Int'l Multimedia Conference*, pages 199–203, 1999.

[26] The Economist. In search of the perfect battery, March 2008.

[27] L. Engebretsen and M. Sudan. Harmonic broadcasting is optimal. In *Proc. of the 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA '02)*, 2002.

[28] R. Erenshteyn, P. Laskov, R. Foulds, L. Messing, and G. Stern. A recognition approach to gesture language understanding. In *International Conference on Pattern Recognition*, pages III: 431–435, 1996.

[29] G. Fang, W. Gao, X. Chen, C. Wang, and J. Ma. Signer-independent continuous sign language recognition based on SRN/HMM. In Ipke Wachsmuth and Timo Sowa, editors, *Gesture Workshop*, volume 2298 of *Lecture Notes in Computer Science*, pages 76–85. Springer, 2001.

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

[31] R. A. Foulds. Piecewise parametric interpolation for temporal compression of multijoint movement trajectories. *IEEE Transactions on information technology in biomedicine*, 10(1), January 2006.

[32] N. Furman, D. Goldberg, and N. Lusin. Enrollments in languages other than English in United States institutions of higher education, Fall 2006. Modern Language Association of America, November 2007.

[33] E. M. Gallaudet. *Life of Thomas Hopkins Gallaudet*. Henry Holt and Company, New York, 1888.

[34] L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. *Multimedia Systems*, 8(4):284–294, 2002.

[35] W. Gao, J. Ma, J. Wu, and C. Wang. Sign language recognition based on HMM/ANN/DP. *IJPRAI*, 14(5):587–602, 2000.

[36] L. Garber. Technology news: Will 3G really be the next big wireless technology? *Computer*, 35(1):26–32, January 2002.

[37] T. Gnoffo. How hot is vod? *The Philadelphia Inquirer*, page C01, Oct 2005.

[38] GSMA. General packet radio service. `http://www.gsmworld.com/technology/gprs/class.shtml`, 2006.

[39] N. Habili, C.-C. Lim, and A. Moini. Segmentation of the face and hands in sign language video sequences using color and motion cues. *IEEE Trans. Circuits Syst. Video Techn.*, 14(8):1086–1097, 2004.

[40] S. Hansell. Snip! nearly one-fifth of homes have no landline. The New York Times, September 2008.

[41] J. E. Harkins, A. B. Wolff, E. Korres, R. A. Foulds, and S. Galuska. Intelligibility experiments with a feature extraction system designed to simulate a low-bandwidth video telephone for deaf people. In *RESNA 14th Annual Conference*, pages 38–40, Kansas City, MO, 1991.

[42] V. Henderson-Summet, R. E. Grinter, J. Carroll, and T. Starner. Electronic communication: Themes from a case study of the deaf community. In *Human-Computer Interaction - INTERACT 2007*, volume 4662 of *Lecture Notes in Computer Science*, pages 347–360, 2007.

[43] H. Hienz, K. Grobel, and G. Offner. Real-time hand-arm motion analysis using a single video camera. In *International Conference on Automatic Face and Gesture Recognition*, pages 323–327, 1996.

[44] N. M. Hogg, C. S. Lomicky, and S. F. WEINER. Computer-mediated communication and the gallaudet university community: A preliminary report. *American Annals of the Deaf*, 153(1):89–96, 2008.

[45] S. Hooper, C. Miller, S. Rose, and G. Veletsianos. The effects of digital video quality on learner comprehension in an American Sign Language assessment environment. *Sign Language Studies*, 8(1):42–58, 2007.

[46] R. Hsing and T. P. Sosnowski. Deaf phone: sign language telephone. In *SPIE volume 575: Applications of digital image processing VIII*, pages 56–61, 1985.

[47] A. Hu. Video-on-Demand broadcasting protocols: a comprehensive study. In *Proc. of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-01)*, pages 508–517, 2001.

[48] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In *Proc. of the ACM SIGCOMM Conference : Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM-97)*, pages 89–100, 1997.

[49] C. L. Huang and W. Y. Huang. Sign language recognition using model-based tracking and a 3D hopfield neural-network. *Machine Vision and Applications*, 10(5-6):292–307, April 1998.

[50] C. L. Huang and S. H. Jeng. A model-based hand gesture recognition system. *Machine Vision and Applications*, 12(5):243–258, 2001.

[51] K. Imagawa, H. Matsuo, R. Taniguchi, D. Arita, S. Lu, and S. Igi. Recognition of local features for camera-based sign language recognition system. In *International Conference on Pattern Recognition*, pages Vol IV: 849–853, 2000.

[52] International Telecommunication Union. International Mobile Telecommunications-2000 (IMT-2000), 2000. http://www.itu.int/home/imt.html.

[53] International Telecommunication Union. Trends in Telecommunication Reform 2007: The Road to NGN, Sept 2007.

[54] C. L. James and K. M. Reischel. Text input for mobile devices: comparing model prediction to actual performance. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 365–371, 2001.

[55] B. F. Johnson and 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.

[56] L. Juhn and L. Tseng. Harmonic broadcasting for video-on-demand service. *IEEE Transactions on Broadcasting*, 43(3):268–271, Sept 1997.

[57] JupiterResearch. Jupiterresearch forecasts wireless subscriber growth will slow due to market saturation between 2008 and 2013, August 2008. http://www.jupiterresearch.com.

[58] M. W. Kadous. Machine recognition of auslan signs using powergloves: Towards large-lexicon recognition of sign language. In *Workshop on the integration of Gesture in Language and Speech*, pages 165–174, 1996.

[59] H. Kalva and B. Furht. Techniques for improving the capacity of video-on-demand systems. In *Proc. of the 29th Annual Hawaii Int'l Conference on System Sciences*, pages 308–315, 1996.

[60] T. Kobayashi and S. Haruyama. Partly-hidden markov model and its application to gesture recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 3081–3084, 1997.

[61] P. Letelier, M. Nadler, and J. F. Abramatic. The telesign project. *Proceedings of the IEEE*, 73:813–827, April 1985.

[62] F. Li and I. Nikolaidis. Trace-adaptive fragmentation for periodic broadcast of VBR video. In *Proc. of 9th Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '99)*, 1999.

[63] F. Li and I. Nikolaidis. An inherently loss-less and bandwidth-efficient periodic broadcast scheme for vbr video. Technical Report TR00-16, University of Alberta, Canada, Jun 2000.

[64] R. S. Ling. *The Mobile Connection*. Morgan Kaufmann, 2004.

[65] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proceedings of the British Machine Vision Conference (BMVC) 2002*. British Machine Vision Association, 2002.

[66] B. L. Loeding, S. Sarkar, A. Parashar, and A. I. Karshmer. Progress in automated computer recognition of sign language. In *Computers Helping People with Special Needs, 9th International Conference, ICCHP 2004*, volume 3118 of *Lecture Notes in Computer Science*, pages 1079–1087, 2004.

[67] M. D. Manoranjan and J. A. Robinson. Practical low-cost visual communication using binary images for deaf sign language. *IEEE Transactions on Rehabilitation Engineering*, 8(1), March 2000.

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

[69] R. E. Mitchell, T. A. Young, B. Bachleda, and M. A. Karchmer. How many people use ASL in the United States? Why estimates need updating. *Sign Language Studies*, 6(3):306–335, 2006.

[70] Joint Model. JM ver. 10.2. http://iphome.hhi.de/suehring/tml/index.htm.

[71] L. Muir and I. Richardson. Perception of sign language and its application to visual communications for deaf people. *Journal of Deaf Studies and Deaf Education*, 10(4):390–401, 2005.

[72] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *SIGCHI Conference Proceedings*, pages 237–242, 1991.

[73] K. Nakazono, Y. Nagashima, and M. Terauchi. Evaluation of effect of delay on sign video communication. In *International Conference on Computers Helping people with Special Needs*, pages 659–666, Linz, Austria, July 2006.

[74] I. Nikolaidis, F. Li, and A. Hu. An inherently lossless and bandwidth efficient periodic broadcast scheme for vbr video. In *Proc. of ACM SIGMETRICS 2000*, pages 116–117, 2000.

[75] E. J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *International Conference on Automatic Face and Gesture Recognition*, pages 889–894, 2004.

[76] S.C.W. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), June 2005.

[77] K. Pahlavan and A. H. Levesque. *Wireless Information Networks*. Wiley-Interscience, New York, NY, 1995.

[78] J.-F. Pâris. A broadcasting protocol for compressed video. In *Proc. of Euro-media '99 Conference*, pages 78–84, 1999.

[79] J.-F. Pâris, S. W. Carter, and D. D. E. Long. Efficient broadcasting protocols for video on demand. In *Proc. of the 6th Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '98)*, pages 127–132, 1998.

[80] J.-F. Pâris, S. W. Carter, and D. D. E. Long. A low bandwidth broadcasting protocol for video on demand. In *Proc. of the IEEE Int'l Conference on Computer Communications and Networks (IC3N '98)*, 1998.

[81] J.-F. Pâris, S. W. Carter, and D. D. E. Long. A hybrid broadcasting protocol for video on demand. In *Proc. of the 1999 Multimedia Computing and Networking Conference (MMCN '99)*, 1999.

[82] D. H. Parish, G. Sperling, and M. S. Landy. Intelligent temporal subsampling of american sign language using event boundaries. *Journal of Experimental Psychology: Human Perception and Performance*, 16(2):282–294, 1990.

[83] D. E. Pearson. Visual communication system for the deaf. *IEEE Transactions on Communication*, 29:1986–1992, December 1981.

[84] D. E. Pearson and J. A. Robinson. Visual communication at very low data rates. *Proceedings of the IEEE*, 73:795–812, April 1985.

[85] S. L. Phung, A. B., and D. Chai. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Trans. Pattern Anal. Mach. Intell*, 27(1):148–154, 2005.

[86] H. Poizner, U. Bellugi, and V. Lutes-Driscoll. Perception of American Sign Language in dynamic point-light displays. *Journal of Experimental Psychology: Human Perception & Performance*, 7(2):430–440, 1981.

[87] M.R. Power and D. Power. Everyone here speaks txt: Deaf people using sms in australia and the rest of the world. *Journal of Deaf Studies and Deaf Education*, 9(3):333–343, 2004.

[88] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.

[89] J. L. Hernandez Rebollar, N. Kyriakopoulos, and R. W. Lindeman. A new instrumented approach for translating american sign language into sound and text. In *International Conference on Automatic Face and Gesture Recognition*, pages 547–552, 2004.

[90] C. M. Reed, L. A. Delhorne, N. I. Durlach, and S. D. Fischer. A study of the tactual and visual reception of fingerspelling. *Journal of Speech and Hearing Research*, 33:786–797, December 1990.

[91] O. Rose. Statistical properties of MPEG video traffic and theft impact on traffic modeling in ATM systems. Technical Report 101, University of Wuerzburg, Germany, 1995. Trace available at `http://www3.informatik.uni-wuerzburg.de/MPEG/`.

[92] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.

[93] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, N.J., 1995.

[94] H. Sagawa and M. Takeuchi. A method for recognizing a sequence of sign language words represented in a japanese sign language sentence. In *International Conference on Automatic Face and Gesture Recognition*, pages 434–439, 2000.

[95] D. Saparilla, K. Ross, and M. Reisslein. Periodic broadcasting with VBR-encoded video. In *Proc. of IEEE Infocom '99*, pages 464–471, 1999.

[96] D. M. Saxe and R. A. Foulds. Robust region of interest coding for improved sign language telecommunication. *IEEE Transactions on Information Technology in Biomedicine*, 6:310–316, December 2002.

[97] J.D. Schein and M.T. Delk Jr. The deaf population of the United States. National Association of the Deaf, 1974.

[98] R. Schumeyer, E. Heredia, and K. Barner. Region of Interest Priority Coding for Sign Language Videoconferencing. In *IEEE First Workshop on Multimedia Signal Processing*, pages 531–536, 1997.

[99] P. Seeling, M. Reisslein, and B. Kulapala. Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial. *IEEE Communications Surveys and Tutorials*, 6(2):58–78, 2004. Trace available at `http://trace.eas.asu.edu/mirrors/h26l/1462.html`.

[100] G. Sperling. Video transmission of American Sign Language and finger spelling: Present and projected bandwidth requirements. *IEEE Transactions on Communication*, 29:1993–2002, December 1981.

[101] G. Sperling, M. Landy, Y. Cohen, and M. Pavel. Intelligible encoding of ASL image sequences at extremely low information rates. In *Papers from the second workshop Vol. 13 on Human and Machine Vision II*, pages 256–312, San Diego, CA, USA, 1986. Academic Press Professional, Inc.

[102] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[103] W. C. Stokoe. *Sign Language Structure: An Outline of the Visual Communication System of the American Deaf*. Studies in Linguistics: Occasional Papers 8. Linstok Press, Silver Spring, MD, 1960. Revised 1978.

[104] S.Young, J.Jansen, J.Odell, D.Ollason, and P.Woodland. *The HTK Book*. Entropic Cambridge Research Laboratory, Cambridge, England, 1995.

[105] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988.

[106] N. Tanibata, N. Shimada, and Y. Shirai. Extraction of hand features for recognition of sign language words. In *Proceedings of International Conference on Vision Interface*, pages 391–398, 2002.

[107] V. C. Tartter and K. C. Knowlton. Perception of sign language from an array of 27 moving spots. *Nature*, 289:676–678, February 1981.

[108] D. R. Traum and E. A. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8:575–599, 1992.

[109] C. Valli and C. Lucas. *Linguistics of American Sign Language: An Introduction*. Gallaudet University Press, 1992.

[110] C. Valli and C. Lucas. *Linguistics of American Sign Language: An Introduction*. Gallaudet University Press, Washington DC, 1995.

[111] P. Vamplew. *Recognition of Sign Language Using Neural Networks*. PhD thesis, University of Tasmania, 1996.

[112] 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 (DCC)*, Snowbird, UT, March 2007.

[113] M. A. Viredaz and D. A. Wallach. Power evaluation of a handheld computer. *IEEE Micro*, 23(1):66–74, 2003.

[114] S. Viswanathan and T. Imielinski. Pyramid broadcasting for video on demand service. In *IEEE Multimedia Computing and Networking Conference*, volume 2417, pages 66–77, 1995.

[115] C. Vogler. *American Sign Language Recognition: Reducing the Complexity of the Task with Phoneme-Based Modeling and Parallel Hidden Markov Models*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, December 2002.

[116] M. B. Waldron and S. Kim. Isolated ASL sign recognition system for deaf persons. *IEEE Transactions on rehabilitation engineering*, 3(3), 1995.

[117] C. Wang, W. Gao, and S. Shan. An approach based on phonemes to large vocabulary chinese sign language recognition. In *International Conference on Automatic Face and Gesture Recognition*, pages 393–398, 2002.

[118] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Techn*, 13(7):560–576, 2003.

[119] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems*, 13(7):560–576, Jul 2003.

[120] W. W. Woelders, H. W. Frowein, J. Nielsen, P. Questa, and G. Sandini. New developments in low-bit rate videotelephony for people who are deaf. *Journal of Speech, Language, and Hearing Research*, 40:1425–1433, December 1997.

[121] S. Wu. General frame level segmentation for periodic broadcast of vbr videos. In *Proc. of the 12th Int'l Conference on Computer Communications and Networks (ICCCN '03)*, pages 143–148, 2003.

[122] M. H. Yang, N. Ahuja, and M. Tabb. Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, August 2002.

[123] Q. Yuan, W. Gao, H. Yao, and C. Wang. Recognition of strong and weak connection models in continuous sign language. In *International Conference on Pattern Recognition*, pages I: 75–78, 2002.

Appendix A

WINDOWS SCHEDULING FOR BROADCAST

H.264 is currently the best way to compress media to achieve high quality at low bandwidth. Since its inception, technologies such as video-on-demand are increasingly realizable. Periodic broadcast is a popular way of implementing video-on-demand, yet most current methods do not work on H.264 because they assume a constant bit rate stream and do not account for B frames. In this appendix, I describe a new protocol for periodic broadcast of video-on-demand that works for variable bit rate streams with B frames. I map the periodic broadcast problem into the generalized windows scheduling problem of arbitrary length jobs on parallel machines. Our method is lossless and practical, in that it does not require channels of differing bandwidth. I apply our method to H.264 encoded video traces and achieve a delay of under 10 seconds on a 1.5 Mbps channel.

A.1 Introduction

With the advent of H.264, movies can be compressed at higher quality with far fewer bits than in earlier standards [119]. Technologies that were previously impractical can now be realized. Video-on-demand (VOD), a service whereby a customer interactively selects and downloads movies and other programming, is one such technology. A popular way to implement VOD is via periodic broadcast, where the customer incurs some delay and then can play the movie from start to finish. However, H.264 makes many current methods infeasible with its use of bidirectional (B) frames and constant quality video. B frames are predicted from previous and upcoming reference frames, and thus cannot be decoded until their reference frames are received. A constant quality video, in which each frame is encoded to nearly the same PSNR, requires a variable bit rate per frame. Most current VOD methods ignore the use of B frames and require a constant bit rate. The protocols that work for variable bit rate require dividing the stream into logical channels of differing

bandwidth, which is impractical for many applications.

In this paper, I introduce a new method for video-on-demand that is flexible enough to support H.264 compressed video, practically implementable, and yet reduces both bandwidth and delay compared to previous methods. The key insight is to model the problem as windows scheduling of arbitrary length jobs on parallel machines [7]. The jobs correspond to frames of the movie. The parallel machines correspond to multiple logical channels. In the same way that using multiple parallel machines can service more jobs than just using one fast machine, using multiple channels reduces delay compared to just increasing bandwidth.

For example, suppose there are two frames of different lengths, and suppose it takes 2 time slots to play a frame. Call the delay D. The goal is to schedule the frames so that, no matter when the user tunes in, she will only need to wait D time slots before playing the movie. I also must ensure that she experiences no further delay once the movie starts.

In order to achieve this, we need to schedule the first frame so that it is received in every window of D time slots and the second frame so that it is received in every window of $D+2$ time slots. With this schedule, no matter when the user tunes in, she will have completely received frame 1 after D time slots. She can play that frame and know that she will have completely received frame 2 once she has played the first frame.

Suppose the length of the first frame is 2 time slots and the length of the second frame is 3 time slots (based on the bandwidth of our channel). The minimum delay D for one channel is 5 time slots. The schedule would be a round robin between frames 1 and 2. To see why the minimum delay cannot be any smaller, consider Figure $A.1$ and suppose the delay were 4. If the user is lucky and tunes in at the beginning of the round robin at the first arrow, she will not experience any problems. But if she is unlucky and tunes in at the second arrow, frame 1 will not be completely received after 4 time slots. I cannot change the schedule from a round robin, or else she will not receive frame 2 in time. On the other hand, if the delay is 5, she will always receive each frame within the corresponding window.

Now suppose I send the frames over two logical channels. Each channel has half of the bandwidth of the original. This means that our lengths are doubled, because it takes twice as long to receive the same number of bits. The length of the first frame is now 4 time slots and the length of the second frame is now 6 time slots. The minimum delay on two logical

channels is $D = 4$. The schedule is simply to send the first frame on the first channel and the second frame on the second channel. The second frame takes 6 time slots to receive, but does not need to be received before the first frame is played, which takes 2 time slots. Thus, a delay of 4 time slots guarantees the second frame will be received in every window of size $D + 2$.

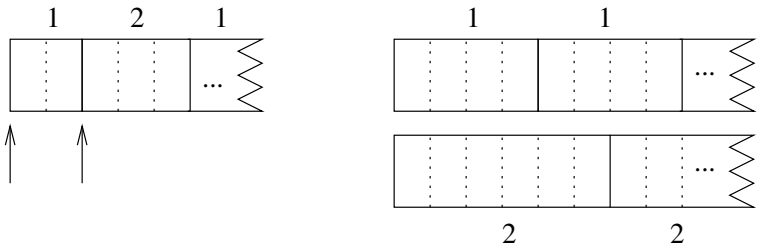


Figure A.1: Schedule on one channel and two channels

The rest of the paper is organized as follows. In the next section, I describe some related work on periodic broadcast VOD as well as the windows scheduling problem. In section 3 I present my algorithms. Section 4 contains the results, and section 5 the conclusion.

A.2 Related Work

Video-on-demand has been studied extensively since the early nineties. There are two main ways in which a video service can provide VOD. A *reactive* service responds to requests from users. Reactive methods include unicast, in which there is a stream between the service and each new user. This does not scale well with the number of users, so researchers have explored multicast methods such as batching users together [24] and merging streams [25]. A *proactive* service anticipates requests. Periodic broadcast is a proactive service that requires the users to wait for some delay before they can begin watching the movie. This is essentially what many VOD services do today. It is the only option for satellite television, which has high bandwidth available for transmitting downstream but none available for transmitting upstream [37].

The primary advantage of periodic broadcast is that it takes much less bandwidth than unicast, and that in turn could allow a video service to offer more movies. Furthermore,

research has shown that 80% of requests for movies are for the 20 or so most popular [23]. If these were sent over a periodic broadcast channel, the video service could meet the other 20% of requests via unicast, saving bandwidth. The main disadvantages of periodic broadcast are the need for a large buffer on the receiving end, lack of some VCR functionality, and the delay. Most periodic broadcast protocols assume that there is plenty of space available on the user's set top box, which is not unreasonable considering the cost of memory. Most protocols also do not allow for the VCR functionality of fast forward, though rewind and pause are easily implemented with a large buffer. Therefore, research in the field focuses on achieving a small delay with a minimal bandwidth. If the delay is on the order of minutes, then a user might turn to another service, such as a regular video rental store.

Perhaps the most natural way to implement periodic broadcast is via *Staggered Broadcast* [24], in which the movie is broadcast in its entirety over several channels at different intervals. If the length of the movie is N and there are k channels, the user experiences a maximum delay of N/k. A lower delay protocol is to send the movie in different segments over k channels, one for each segment. The channels may have differing bandwidth. Most of the research in the field uses this technique [114, 48, 56, 79, 81]. For a detailed survey, see [47]. These methods assume that the movie can be broken up in arbitrary places, or at the very least, along frame boundaries, which is not the case with any coding standard that includes B frames. Furthermore, they assume that the movie is received at some constant bit rate (often the playback rate). It is possible to encode a constant quality movie at a constant bit rate, but the bandwidth requirement will be much higher than for the variable bit rate version [22].

Methods for variable bit rate VOD include lossy methods that use smoothing, server buffering and client prefetching [95, 62]. There are also several lossless methods. *Variable Bandwidth Harmonic Broadcasting* [78] changes the bandwidth per channel depending on the movie; each channel has differing bandwidth. The *Loss-Less and Bandwidth-Efficient* (LLBE) protocol [74] divides the movies into segments that respect frame boundaries. Each segment is broadcast in its own channel, at differing bandwidths per channel. The segments' divisions are chosen based on a dynamic program that returns the division that gives rise to the minimum total bandwidth. *General Frame Level Segmentation* (GFLS) [121] modifies

LLBE to work on MPEG videos with B frames. The differing bandwidths used in these lossless methods are not co-factors, i.e., it is not generally the case that one channel's bandwidth is a multiple of another's. This renders them impractical for current video service providers.

The technique most closely related to ours is harmonic broadcasting[56, 79, 80]. Harmonic broadcasting divides the movie into segments and broadcasts segment i at bandwidth proportional to $1/i$. The worst case delay for bandwidth b asymptotically approaches $1/(e^b - 1)$. This is optimal for constant bit rate periodic broadcast [27, 34, 47]. Harmonic broadcasting has a nice mapping to windows scheduling, explored in [6]. That work showed that the optimal delay can be approached in the limit using windows scheduling techniques on channels of equal bandwidth.

A.3 Algorithm

A.3.1 Conversion to windows scheduling

I wish to schedule a periodic broadcast of a variable bit rate H.264 video so that no matter when users tune in to the broadcast, they experience a delay of D before they can begin playing the movie. I model the problem as windows scheduling of arbitrary length jobs on parallel machines.

The windows scheduling problem takes as input a sequence of n positive integer pairs $I = \langle (w_1, \ell_1), (w_2, \ell_2), \ldots, (w_n, \ell_n) \rangle$, representing n jobs. The ith job has length ℓ_i and must be executed within every window of size w_i. The goal is to schedule the jobs on the minimum number of parallel processors. Solving the problem optimally is NP-hard, but an 8-approximation is known, as well as a practical greedy algorithm [7].

I convert the video-on-demand problem to a windows scheduling problem as follows. The processors correspond to logical channels and the lengths correspond to the frame sizes. The window sizes correspond to our guarantee that the users experience only a small fixed delay before they can begin playing the movie.

Specifically, if L is the time it takes to play one frame, I must guarantee that frame 1 is received in every window of size D, that frame 2 is received in every window of size $D + L$,

and that frame i is received in every window of size $D + (i-1)L$. Since the window size is in units of time, the frame lengths must be converted to time, and this will depend on the bandwidth and the number of logical channels.

Let the bandwidth in bits per second be denoted B, the number of logical channels k, the delay in seconds D, and the time it takes to play one frame L. Let b_i be the size in bits per frame i. Then the bandwidth per channel is B/k and the length of each frame in seconds is $b_i k/B$. The job sequence is then

$$\langle (D, b_1 k/B), (D + L, b_2 k/B), (D + 2L, b_3 k/B), \ldots, (D + (n-1)L, b_n k/B) \rangle .$$

Each window must be at least as large as the length of the job scheduled on it, so in particular, $D \geq b_1 k/B$.

A.3.2 Solving windows scheduling

Though the windows scheduling problem is NP hard, there is a greedy algorithm that works quite well in practice [7]. The algorithm is easier to visualize with the help of the tree representation of a schedule. Each channel is represented by a directed tree. The nodes of the tree consist of (w, ℓ) pairs. A (w, ℓ) node represents a window of size w and a length of size ℓ, so any (w', ℓ') scheduled on this node must have $w \leq w'$ and $\ell \geq \ell'$. There are two ways to schedule jobs on (w, ℓ):

1. (w, ℓ) may be split into k children of size (wk, ℓ). This corresponds to a round robin schedule over the children.

2. (w, ℓ) may be split into k children of size $(w, \ell_1), (w, \ell_2), \ldots, (w, \ell_k)$ such that $\sum_{i=1}^{k} \ell_i = \ell$. This corresponds to subdividing the window and allocating the appropriate number of slots to jobs.

For example, suppose I had a playback rate of $L = 3$ and converted frames lengths of $\ell_1 = 2, \ell_2 = 1, \ell_3 = 1$. Then to play on one channel, our minimum delay would be 3, giving $\langle (3, 2), (6, 1), (9, 1) \rangle$. The tree representation of the schedule is in Figure A.2. In every

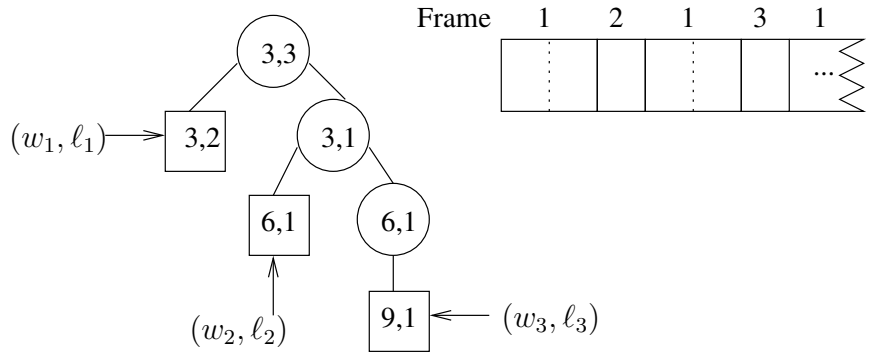


Figure A.2: Tree representation and corresponding schedule. Boxes represent jobs.

window of size 3, the schedule would visit the $(3, 2)$ node, then one of the $(6, 1)$ or $(9, 1)$ nodes, as shown in the figure.

To find the tree, the algorithm simply starts with one node of size (w_1, w_1) and schedules the first job (w_1, ℓ_1) as a child. That leaves a node of size $(w_1, w_1 - \ell_1)$. The next job, (w_2, ℓ_2), is scheduled as a child of this node by one of the two methods listed above. Note that our window sizes are always increasing. If the ith node cannot be scheduled, a new tree starts with (w_i, w_i) as its root.

My goal is different than that of windows scheduling; I want to minimize the delay or the bandwidth for a given set of frame lengths. To find the minimum delay, I set the bandwidth size and the number of channels and convert the frame lengths. I know that the minimum delay is at least the converted size of the first frame. I run the greedy algorithm with the delay set to this and the number of trees capped at the number of channels. If the algorithm reaches the cap, I run it again with double the delay. I continue doubling until I find a feasible delay given the number of channels; a delay equal to the sum of lengths is always feasible. Once I have a feasible delay, I perform a binary search between it and the last infeasible delay in order to determine the smallest delay that is feasible.

To find the minimum delay over any number of channels, I run the algorithm with the number of channels set from 1 to n, where n is the maximum number of channels. This adds a multiplicative factor of n to the running time. The algorithm for finding the minimum bandwidth is similar; I set the delay at the beginning and run the greedy algorithm for

differing values of bandwidth. In this case, I set a maximum bandwidth of 100 Mbps. Since a window can never be smaller that its length, I can throw out some infeasible bandwidths right away (e.g. those in which $D \leq \ell_1 k/B$ where k is the number of channels).

A.3.3 H.264 modification

The above model is too simplistic in that it does not take B frames into account. In H.264, there are three frame types: I frames, P frames, and B frames. I frames are coded independently of the other frames in the video. P frames are predicted from previous I or P frames. B frames are predicted from previous or future I or P frames. If I used the above model based only on frames, I could no longer guarantee that the user could play the movie without interruption.

Our solution is to combine the frames into segments that do not depend on future segments. For example, a common H.264 coding sequence is IBBPBBPBBPBBI... Our algorithm divides this into segments:

$$\text{I, BBP, BBP, BBP, BBI}, \ldots$$

Each segment no longer depend on future segments. The windows scheduling algorithm will guarantee that all preceding segments are received before the current segment, and that the current segment is received in its entirety before it must be played.

Since different segments will be scheduled on the same channel, I include a unique identifier at the beginning and end of each segment. The number of segments will certainly be smaller than 2^{32}, so I use a 32-bit integer for our unique identifier. This adds a total of 64 bits to each segment.

A.4 Results

In order to demonstrate the applicability of our algorithm, I ran it on several video traces. The first is from [99] and uses H.26L, the working version of H.264, to encode the movie "Starship Troopers" at constant quality. Here I use the trace file with QP=20. The coding sequence is IBBPBBPBBPBBI. The number of frames is 90,000 and the length of the movie is 60 minutes. Figure A.3 shows the minimum delay at different bandwidths and also the

minimum bandwidth at different delays, plotted against the number of channels. The most gain occurs when I move from one channel to two. Though the results continue to improve as I add more channels, the returns are diminishing. Since the complexity of reading and reassembling the movie increases as the number of channels increase, I limit the number of channels in the graph to 25.

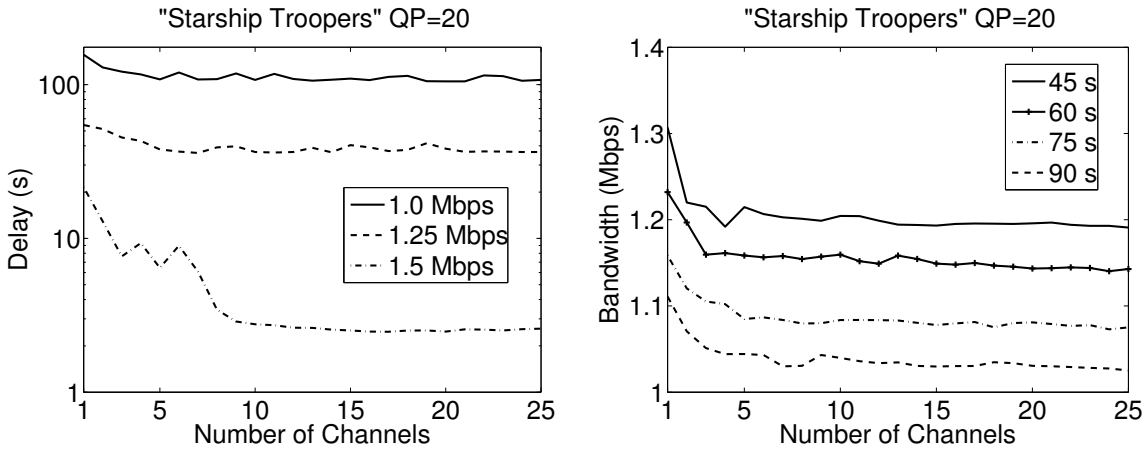


Figure A.3: Delay at varying bandwidths and bandwidth at varying delays for "Starship Troopers"

These results demonstrate that a practical VOD system using H.264 is feasible, even for satellite television, where interactivity is not possible. Note that the delay for a 1.5 Mbps logical channel is less than 10 seconds. Since the bandwidth available on one cable channel is at least 30 Mbps [59], the provider could divide the channel into 20 subchannels, each of which would play one movie. Each subchannel of 1.5 Mbps would be further divided into the number of logical channels that guaranteed a low delay; in the case of "Starship Troopers", 5 channels would suffice. The division process is simple since the bandwidth is distributed equally. When the user requests one of the 20 movies, the set top box tunes into the appropriate channel and the movie begins playing in less than 10 seconds.

I also compare our results to LLBE/GFLS (the results for the two algorithms are quite similar [121]). The results from LLBE are expanded upon in a technical report [63], in which they plot delay against optimal server bandwidth on 7 channels for the MPEG-1 traces from

[91]. In Table A.1, I compare our minimum bandwidth using windows scheduling (WS) on 7 logical channels to theirs for traces mtv_1, news_2, and soccer_1 (fuss). Because MPEG-1 is a poor compression standard compared to H.264, the bandwidth requirements are much higher. Though their method uses different bandwidths for different channels, our algorithm, with equal bandwidth channels, obtains comparable results.

	Delay (s)	LLBE	WS
mtv_1	15	3.8	3.3
	30	3.2	3.0
	60	2.6	2.6
	90	2.3	2.4
news_2	15	2.4	2.2
	30	2.0	1.9
	60	1.5	1.6
	90	1.4	1.5
soccer_1 (fuss)	15	4.1	3.6
	30	3.5	3.3
	60	2.8	2.9
	90	2.4	2.6

Table A.1: Minimum bandwidth (Mbps) for given delay

A.5 Conclusion

I have presented an algorithm for periodic broadcast of variable bit rate movies. My method is practical, in that it does not require channels of differing bandwidth to achieve low delay. In the future, I will look for better windows scheduling algorithms for this application. I would also like to explore the advantages of pre-caching the first few frames of a movie on the user's set top box at some point before the movie is requested. This could provide significant improvement in minimum bandwidth for a given delay, without taking up too

much storage space. Finally, I would like to explore other problems in the space, such as taking user bandwidth limitations into consideration.

VITA

Neva Cherniavsky received the B.S. in computer science and mathematics from Tufts University in 2001 and her M.S. in computer science from the University of Washington in 2004. She was awarded an NSF graduate research fellowship in 2001. She has published research in conferences and journals in diverse areas, including auction theory, accessibility, data compression, and multimedia. For her postdoctoral work, she will be a part of the INRIA Willow project, based in Paris, France.