Maintaining Intelligibility of ASL Video in the Presence of Data Loss

Sam Whittle
Honors Thesis
The Department of Computer Science and Engineering
University of Washington

June 5, 2008

Abstract

The goal of the MobileASL (American Sign Language) research project is to enable sign language communication over the US cellular network. The cellular network is low bandwidth and lossy. Data loss can greatly impact the quality of compressed video because of temporal and spatial error propagation. I investigate techniques to minimize the effect of error loss on the intelligibility of transmitted ASL video. As both computational power and bandwidth are limited on cellular devices, I concentrate on determining the best allocation of these resources. Specifically I focus on utilizing feedback to recover from data loss.

Contents

1 Introduction

The goal of the MobileASL (American Sign Language) research project is to enable sign language communication over the US cellular network. Though similar systems are available in other countries, such as Sweden, these systems run on 3G cellular networks which have greater bandwidth capacity than currently available in America [13].

American Sign Language (ASL) is the primary language of many Deaf individuals and is therefore preferred greatly to text for its expressiveness and ease. It is estimated that it is the preferred language of more than 500,000 Deaf people in the U.S [8]. Signs are composed of the location, orientation, and shape of the hands and arms, along with facial expressions. Though ASL usually uses two hands, one handed signing is not uncommon. Additionally signs usually occur within the region around the speaker's head and torso [2].

It is an unsolved problem to automatically interpret signs using computer vision and artificial intelligence. There have been only modest achievements despite the use of motion capture technology and trying to recognize a confined dictionary [11]. Therefore we focus on transmitting intelligible video and not translation or further analysis of signs.

However, developing a video conversational service for the US cellular network presents many challenges. Some of these challenges are due to the modest capabilities of cellphones. Our system needs to be able to encode/decode in real time with limited processing power. We also need to be mindful of memory and battery usage constraints. Other difficulties are introduced by the cellular network. The network is of limited bandwidth and there are no guarantees that packets sent will be successfully received.

It is beyond the scope of this paper to present all aspects of the MobileASL research project [9]. Instead the focus of this paper is to develop and evaluate computationally feasible techniques for maintaining intelligibility of ASL video despite losses of data. An overview of video compression is presented first followed by approaches to prevent or conceal data loss.

2 Video Encoding

Video compression is generally achieved through motion compensation. Videos have spatial correlation and high temporal correlation between frames and

thus portions of a frame can be effectively predicted within a frame or from previous frames. This often requires fewer bits, for the same quality, than would be used by independent encoding.

The video standard used by the MobileASL project is H.264 [10], which achieves much better coding efficiency than previous video codecs through features such as adaptive block size, intra coding with spatial prediction and in-loop deblocking. Also important is that H.264 was designed with transmitting over packet networks in mind; the standard includes a network abstraction layer (NAL) and produces NAL units that are suitable for packet transmission.

H.264 also contains many features that can increase the compressed video's robustness to data loss. These features include slice-structured coding, multiple reference frames, and data partitioning [14]. Unfortunately these features are not well-suited to our goal of real-time, low-complexity encoding and decoding. An overview of video compression will be given and then these features will be further described.

2.1 Encoding Decisions

To simplify the process of encoding a video, the abstraction of a macroblock is introduced. Each frame of the video is first subdivided into 16 by 16 blocks of pixels, known as macroblocks. If it is beneficial, in terms of bits, each macroblock can be further subdivided for encoding. A macroblock can be encoded as an intra-block, inter-block, or a skip-block. In normal encoding, these decisions are made based upon rate-distortion optimization.

An intra-block is encoded referencing only previous blocks of the current frame (see Figure 1). Thus encoding a macroblock as an intra-block breaks temporal dependencies but still takes advantage of spatial dependencies. Intra-blocks are often only used for macroblocks that do not resemble portions of previous frames.

An inter-block is encoded with respect to some portion of a previously determined reference frame (see Figure 2). Each inter-block is encoded as a motion vector indicating the referenced area and the difference of the two regions. Because the decoding of an inter-block depends on arbitrary regions of other frames, temporal and spatial dependencies are introduced. However, this often means that, for a given quality, encoding a macroblock as an inter-block uses fewer bits than encoding it as an intra-block. To ensure that the frames can be encoded in real-time, the search used for motion vectors is not

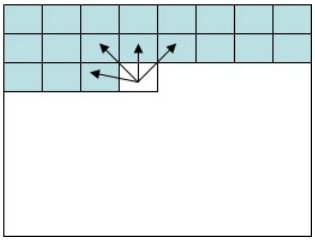


Figure 1: Encoding a intra-block.

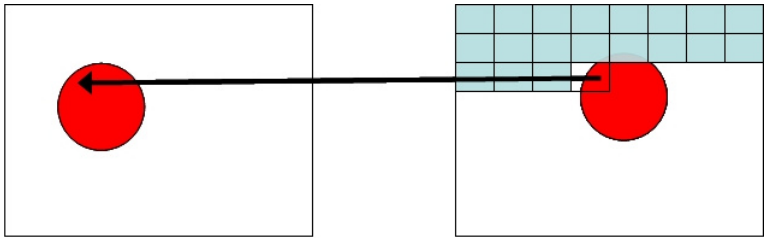


Figure 2: Encoding a inter-block.

exhaustive and thus the motion vector used may not be optimal.

If a macroblock remains unchanged from the reference frame, it can be encoded as a skip-block. Skip-blocks require the minimum number of bits and therefore the successful detection of possible skip-blocks greatly reduces the bits necessary to encode a frame.

Encoding decisions on the macroblock level are guided by frame types. There are three possible frame types: I-frames, P-frames, and B-frames. An I-frame is a frame in which every macroblock is encoded as an intra-block. I-frames therefore break all temporal dependencies and only have spatial dependencies within the frame. P-frames contain intra-blocks, inter-blocks and skip-blocks. The encoding method for each macroblock is chosen independently to minimize the bits necessary for a set quality. B-frames are bi-directional frames that include inter-blocks predicted from both a frame in the future and a frame in the past. Since future frames are unavailable in real-time encoding, B-frames cannot be used in our situation without buffering and additional delay.

After the decision on how to encode a macroblock is made, it is encoded at a set quality. This quality is determined by the selection of the quantization parameter (QP) value. A lower QP value means that the encoding is of

higher quality. The encoder adjusts the QP throughout the encoding process to maintain a desired bit rate. Thus using a large number of bits in one frame means the encoder reduces the quality of subsequent frames to ensure that the average bit rate constraint is met.

2.2 Resiliency to Data Loss

The U.S. cellular network is a best-effort network; there are no quality-of-service guarantees that packets sent will be received successfully or in order. Unfortunately, due to the spatial and temporal dependencies of motion-compensated compressed video, a single loss can corrupt large portions of future frames (see Figure 7(a)). This is known as error propagation.

For ASL conversations to remain intelligible, we must successfully overcome the effects of lost packets. There are two general approaches to dealing with data loss: preventing the loss or concealing the loss of data. The H.264 standard includes several several enhancements to aid in utilizing either of these approaches [14].

2.2.1 Preventing Data Loss

To prevent loss of data, some type of redundancy needs to be present in the system, as losing packets is unavoidable. TCP or some other protocol with automatic retransmission can ensure that all data are received correctly. However retransmitted frames arrive after a delay greater than the round trip time of the connection. This delay is unacceptable because frames will arrive after they should have been displayed.

Forward error correction (FEC) is another approach for preventing data loss [7]. By using error correcting codes, for every k packets we calculate and transmit $n > k$ packets with the property that if any k of n packets are received, the original data can be reconstructed. This approach could be effectively paired with the data-partitioning feature of H.264, which arranges sections of the compressed video based upon their relative importance to decoding. However error correcting codes are designed based upon the selection of n and k. For such a code to be effective despite bursts of packet losses, the ratio of n to k would have to be large. Thus the amount of data transmitted would increase greatly. Since the bandwidth of the cellular network is already limited, this additional bit cost is prohibitive. Additionally, delay

is necessary for FEC because k packets must be received before decoding can begin.

2.2.2 Concealing Data Loss

Unlike preventing data loss, concealing data loss does not necessitate introducing delay into video playback. Basic error concealment can take place at the decoder; if regions of a frame are detected as lost, they can be estimated from neighboring blocks that were successfully received.

Error concealment can also take place at the encoder by making decisions to make the stream resilient against error propagation. One tool available in the H.264 standard is slice structured coding. Slices are regions of consecutive macroblocks and each slice is encoded independently from the others. Thus if a decoding error occurs within a slice, the effects of error propagation are limited to that slice. Unfortunately this benefit of slices is nonexistent in our situation; the low bit rate and small size of our video means that entire frames are lost at once.

Other techniques have been proposed to encode resilient videos by predicting the error of the decoder's reconstructed video using packet-loss probability models [15, 16, 20]. In these approaches, the distortion predicted by the model for each macroblock is used for rate-distortion optimization. Such techniques improve the robustness of the encoded video to data loss but require many calculations and additional memory. Also these techniques rely upon knowledge of how the decoder conceals loss, which could limit our ability to utilize an unmodified hardware decoder on the phone.

Previous work has also explored the benefits of multiple reference frames, another feature of H.264 [19, 5]. Video compression is improved by storing multiple reference frames from which to predict inter-blocks. Multiple reference frames can also be beneficial in recovering from loss if feedback is given by the decoder to the encoder. In such cases, a reference frame used for recovery must temporally proceed the frame lost. Therefore if n frames can be sent in the round trip time, we must store n reference frames in a cyclical array. This is a nontrivial use of the cell phone's limited memory resources. Though these techniques are not directly applicable, utilizing feedback is a useful strategy that is further developed in the next section.

3 Low-Complexity Feedback-Based Encoding

Several observations particularly motivate the use of feedback to recover from data loss in cellphone ASL conversations. First, because our system is a real-time conversation, participants can ask for clarifications if necessary. Thus short disruptions in quality are acceptable, similar to how short bursts of static do not greatly hinder normal cellphone conversations. Such disruptions would occur between the time of the loss and when the results of feedback are received.

Another ramification of two-way communication is that it is simple to use feedback to indicate frames dropped. This requires minimal additional complexity because this feedback can piggyback upon video packets we already send. Lost frames can be detected by numbering the packets or noting the frame number contained in a packet.

Since our frames are small and at encoded at a low bit rate, each packet loss corresponds to the loss of an entire frame. Thus decoder based error concealment is difficult as no successfully received neighboring blocks are available to predict from. Therefore we focus on several strategies utilizing feedback to encode for recovery. Frames sent in response to feedback will be called recovery or refresh frames.

3.1 Simple Intra-frame Refresh

A simple approach of encoding for robustness is to encode with intra-frames at regular intervals. Since an intra-frame breaks all dependencies with previous frames, receiving an intra-frame successfully halts error propagation, though at the cost of increasing the bits used in the frame. However it is difficult to determine the appropriate interval between intra-frames, known as the max group of pictures (GOP) length. If this value is large we risk long intervals of corrupted video. However it cannot be small because then many I-frames would be sent and the video quality would degrade due to bit constraints.

Utilizing feedback, we can improve this method by only encoding intra-frames when we detect that a frame has been lost. By doing this, we avoid having to choose a GOP length and instead adjust to network conditions. This also guarantees that periods of corruption last for less than the round-trip time of the connection.

3.2 Bursty I-frame Refresh

The U.S. cellphone network experiences losses in bursts. Bursty losses are particularly devastating to the intelligibility of decoded videos [4]. It is important to adapt our techniques to perform well in the presence of bursty losses.

We can improve our approach of using I-frames to refresh by noticing that a single intra-frame can recover from multiple lost frames. Consider the first lost frame of a burst of lost frames. When this loss is detected, the decoder responds with feedback indicating that an intra-frame should be encoded. However before this intra-frame is received by the decoder, additional frames are lost and generate feedback. The encoder can safely ignore some of these further requests to encode an intra-frame because only one intra-frame is required per round trip time. Even though the initial intra-frame is sent when the encoder has no knowledge of the future losses, it is able to do recover from them because intra-frames are unselective in what they fix.

See Figure 3 for an illustrative example. The left column represents the frames produced by the encoder and the right column is the state of frames shown by the decoder, with time running vertically. Arrows indicate the transmission through the network of frames and feedback due to losses of frames. Note how the single intra-frame recovers from multiple corruptions. In this example, the round trip time is only 4 frames. As the round trip time increases, the benefits of this method grow equally.

3.3 Bursty P-frame Refresh

Most times that feedback is received, many of the decoder's displayed macroblocks are still uncorrupted. Thus sending an entire intra-frame may be refreshing areas of the frame unnecessarily.

We can safely reduce the number of intra-blocks necessary to refresh the video if we calculate macroblocks that are uncorrupted from the loss. These macroblocks include blocks that have been encoded as skip-blocks for longer than the round trip time. This ensures that the block has remained unchanged from before the loss occurred until the recovery frame is received. Macroblocks such as this are therefore unaltered by the data loss. In our ASL videos, this often occurs in the background of a conversation, especially when users set the phone down to allow themselves to sign with both hands.

Now, when encoding the refresh frame in response to feedback indicating

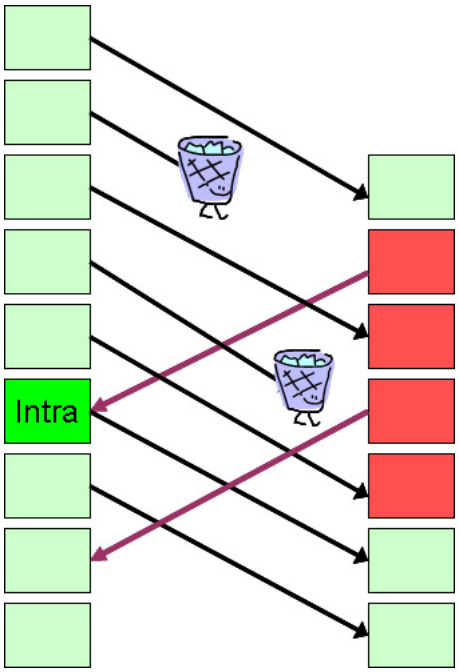
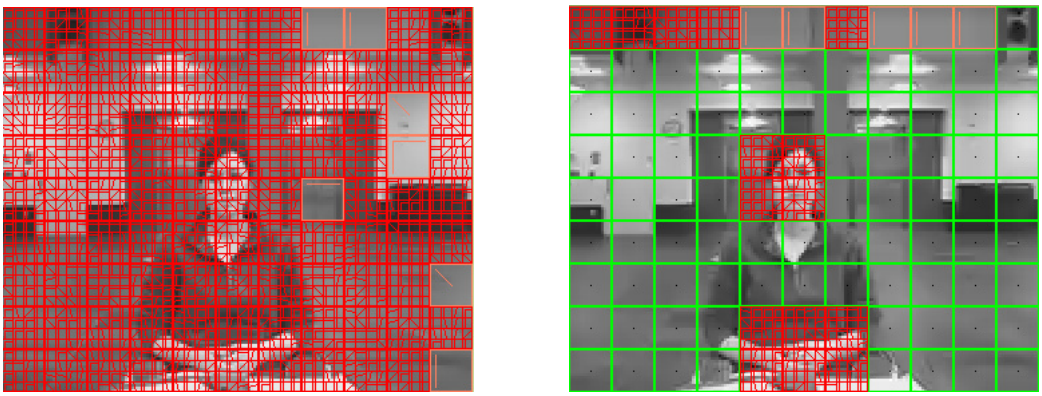


Figure 3: Demonstration of single intra frame recovering from multiple losses. The state of the encoder and decoder are on the left and right respectively. Time advances toward the bottom of the page.



(a) I-frame (b) P-frame

Figure 4: Comparison of number of intra-blocks contained in an I-frame and a P-frame used for refreshing. Intra-blocks are drawn in red and skip-blocks are indicated with green. Most gains will not be as extreme.

a loss, we can instead use a P-frame that includes only intra-blocks and skip-blocks. Skip-blocks are sent for those macroblocks identified as safe to skip by the above criteria. The resulting refresh frames often contain fewer intra-blocks and therefore make better use of bits (Figure 4). Decoded quality also improves because regions of the background remain constant and do not flicker because of unnecessary intra-blocks, which are usually encoded at lower quality. This technique requires little computation and only one integer per macroblock to store how many times that macroblock has been skipped in a row. These data are stored and calculated by the encoder.

We can improve the performance of this algorithm for bursts of losses by recognizing how the first P-refresh frame affects which macroblocks are safe to skip. After the first P-refresh frame is received, the decoder's state will be completely pristine because each macroblock is either a skip-block identified as uncorrupted or is an intra-block that is predicted from skip-blocks or previous intra-blocks, which are inductively uncorrupted (see Figure 1. Therefore these P-refresh frames break all temporal dependencies nd we can use them similar to I-frames, only needing to send one refresh frame per round trip time. This approach is therefore particularly applicable to bursty networks.

3.4 Tracking P-frame Refresh

Several papers develop a method known as error tracking, that utilizes feedback indicating lost macroblocks [6, 1]. Error tracking consists of creating a tree of dependencies by examining reference blocks used when encoding inter- and intra-blocks. Using these data, it is possible to detect all macroblocks dependent upon each macroblock that feedback indicates as lost. This technique works well but uses a significant amount of memory for the cell phones. However since we can assume the entire frame is corrupted by the loss, the process of error-tracking is simplified; we replace the tree data structure with three arrays while maintaining similar functionality.

In order to determine which macroblocks are uncorrupted, we need to store how long ago the block was based on a block we determined as uncorrupted. This count is updated differently depending upon how the macroblock is encoded. One array *curr* holds this count for each macroblock of the current frame; a second array *prev* holds the same values but for the previous frame; and a third array maintains a count of consecutive skip blocks. If a macroblock (x, y) is encoded as an intra-block, we set

$$curr[x, y] = \max\{curr[x-1, y-1], curr[x, y-1], curr[x+1, y-1], curr[x-1, y]\} \tag{1}$$

(using 0 for those indexes that are out of bounds). These are the neighboring blocks from which (x, y) can possibly be predicted (see Figure 1). If (x, y) is encoded as an inter-block, we set

$$curr[x, y] = \max\{prev[a, b]\} + 1 \tag{2}$$

over all macroblocks (a, b) from which (x, y) is predicted, as determined by the motion vectors (see Figure 2). If (x, y) is a skip-block, we set

$$curr[x, y] = prev[x, y] + 1 \tag{3}$$

if the skip count is less than the round trip time in frames and we set it to zero otherwise. This captures the dependencies desired with efficient use of memory. This is the most computationally intense approach, although the most intensive calculation, inter-block dependencies, could be combined with the motion vector search.

When feedback is received, we are now able to send a P-frame that includes intra-, inter-, and skip-blocks. A macroblock (x, y) can be safely skipped if $prev[x, y]$ is less than the round trip time. A macroblock can be

encoded as an inter-block only if $prev[a, b]$ is less than the round trip time for all macroblocks (a, b) from which it is predicted. A macroblock can always safely be encoded as an intra-block. Currently the motion vector search for refresh frames is standard and could be enhanced by trying to avoid making predictions from unclean macroblocks. Thus it is currently necessary to switch inter-blocks to intra-blocks if they are unsafely predicted. The downside of including inter-blocks in the refresh frame is that we need to send a tracking P-frame refresh for every frame lost.

4 Results

This section contains simulation results of our techniques. We present comparisons between using no recovery and our approaches to demonstrate the effectiveness of utilizing feedback.

4.1 Methodology

For our experiments we use a simulation that is not real-time. We determine sequences of losses with a Gilbert-Elliot model and use these sequences to simulate losses [3] (Figure 5). The sequences we generated were designed to include bursts of loss with an overall loss rate of up to 5%, the predicted loss rate of the cell phone network [12]. Three sequences, generated by our model, were used for the simulations producing the following data. Sequence three is notable because it contains longer bursts of lost frames than sequences one and two.

We also assume a fixed round trip time, measured in frames, denoted by $rttf$. For our results the round trip time was set equal to 7 frames. This is reasonable because the videos run at 15 frames per second and the latency of the cell-phone network is estimated to be $500ms$. In practice, the round trip time in frames could be calculated based upon actual network conditions, thus calibrating the recovery to be most effective and efficient.

We encoded each video assuming feedback was received $rttf$ frames after the losses indicated by the sequence, where $rttf$ is the round trip time of the connection. When the resulting video is decoded, the predetermined frames are "lost," resulting in data corruption that should be handled by our recovery methods. Changing the simulation to implementation should only involve replacing the simulated loss and feedback with actual network

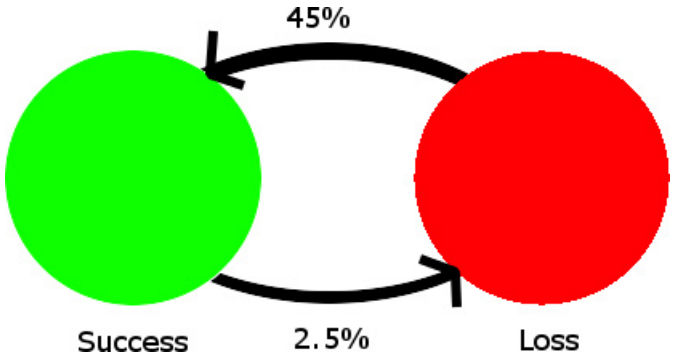


Figure 5: Gilbert-Elliot two state loss model. Transition probability of success state to loss state is 2.5% and probability to transition from loss state to success state is 45%.

activity.

The simulation was carried out on three ASL videos. Each video is centered on an ASL signer, who occupies approximately the center third of the frame (see Figure 7 for an example). The Graduation video contains a complex but fairly static background, only interrupted by someone walking by. The Food video is taken against a white background that remains constant throughout the video. The Accident video is very busy, with many individuals moving behind the signer.

4.2 Measuring Intelligibility

ASL intelligibility is difficult to measure. To substitute for user-studies at this stage of research, we instead measure the peak signal to noise ratio (PSNR), a common measure of corruption due to image compression. The PSNR is calculated for corresponding frames between the simulated video and the source video. PSNR is inversely related to the mean squared error of the simulated video and thus a higher PSNR indicates a better reconstruction.

Although this metric does not directly correspond to the intelligibility of ASL, the average PSNR over all frames of a particular video can indicate the persistence of corruption introduced by data loss. In simulation, all of our techniques resulted in a higher average PSNR of the decoded video than without recovery methods (Figure 6 and Table 1).

The improvements gained from our recovery techniques are fairly equivalent. The simple I-frame method performs slightly worse than the others

because using so many I-frames results in the encoder selecting lower QP values. For each video, the improvement is more dramatic when using loss sequence three. This is the case because sequence three contains long bursts of lost frames which greatly corrupt the video if no recovery technique is used.

The the success of the techniques is best realized by watching the videos themselves. To capture this, several pictures comparing recovery techniques are also presented (see Figure 7). Notice the presence of artifacts when no recovery technique is used, despite that the most recent loss occurred 15 frames previously. Though our methods result in videos that remain briefly corrupted, these errors are limited to the round trip time.

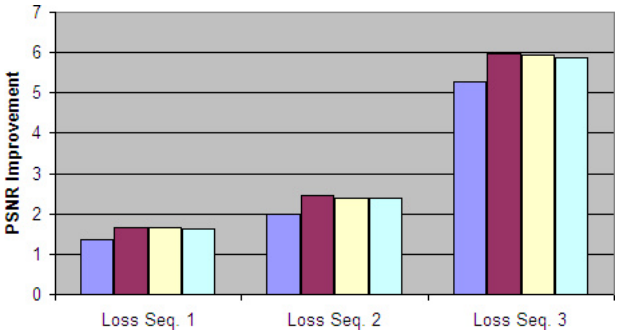
4.3 Analysis

From Figure 6 it appears that the bursty I-refresh and both P-refresh methods are fairly equivalent with respect to PSNR. Each method recovers from lost frames once feedback is received (Figure 8(a)). This is only part of our goal, however, since we are also concerned with each method's consumption of bits and added computational complexity.

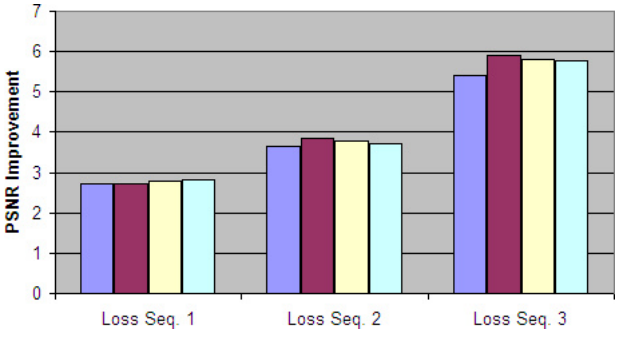
As indicated earlier, the number of intra-blocks used by a recovery method greatly affects the size in bits of a refresh frame. We expect the I-refresh method to produce refresh frames that use the most bits and the refresh frames produced by the tracking P-refresh method to be the most conservative with bits, since these frames can include all types of macroblocks. This is observed in Figure 8(b).

Note that for longer bursts, the bursty I-refresh method only sends one frame of large size, compared to several large frames with the simple intra-refresh method. This corresponds to a lower maximum bit rate as several consecutive intra-frames consume many bits (see Table 3). The tracking P-frame mode sends frames using the fewest bits but because it must send one for every burst, this mode can end up using more bits to recover than would be consumed by sending a single P-frame without inter-blocks. This is most evident when comparing the maximum bit rates observed with loss sequence three, the sequence containing several lengthy bursts.

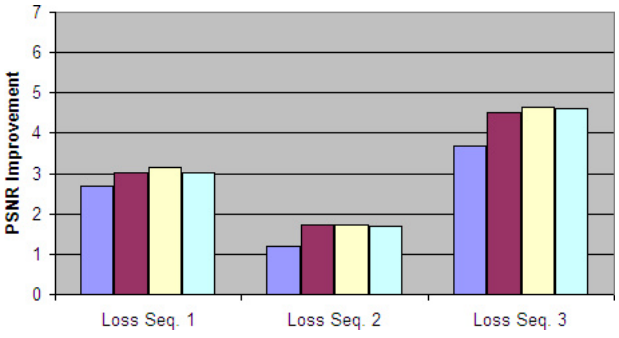
Another indication of the usage of bits is the QP value chosen by the encoder. The encoder adjusts the QP value in order to maintain a desired average bit rate. Higher QP values result in lower quality encodings. Thus by comparing the average QP values for different recovery techniques on the



(a) Accident Video



(b) Food Video



(c) Graduation Video

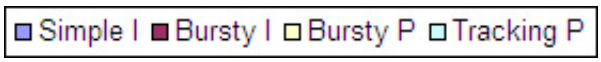
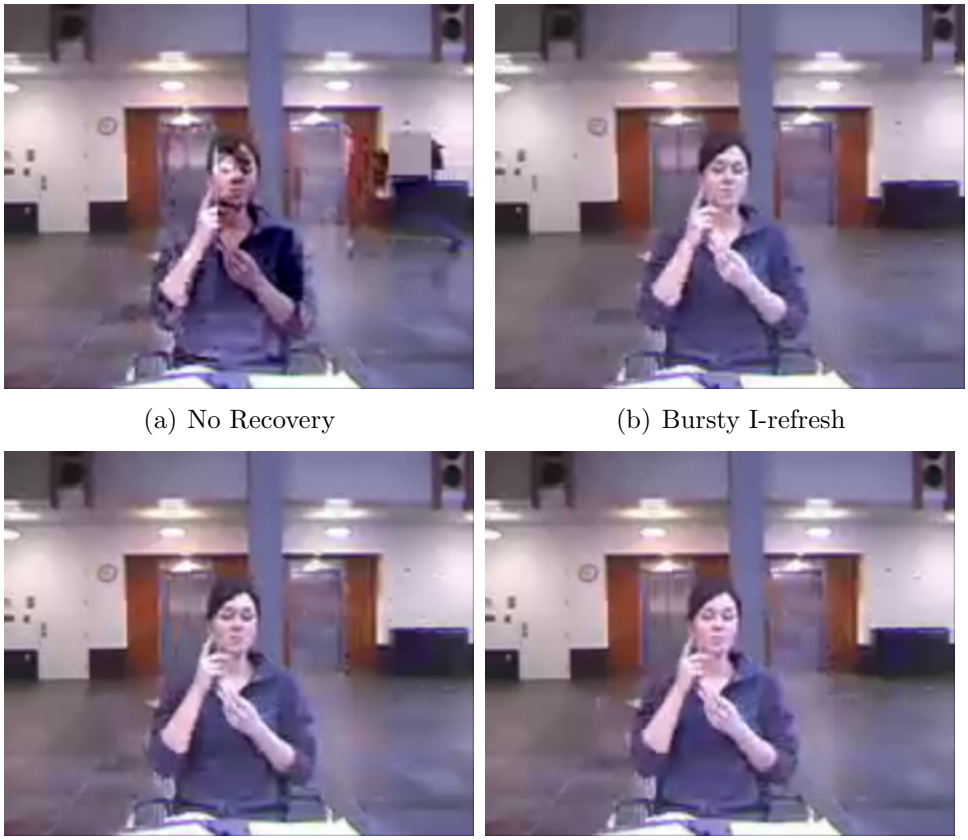


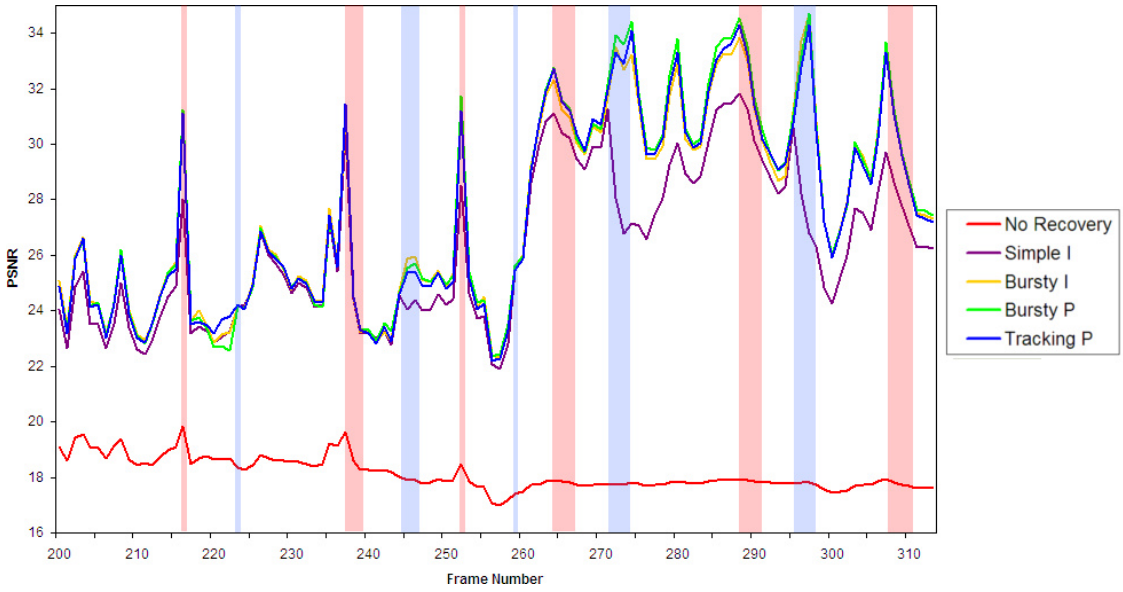
Figure 6: Improvement in average PSNR using various techniques compared against no recovery method used, determined with three sequences of determined losses.



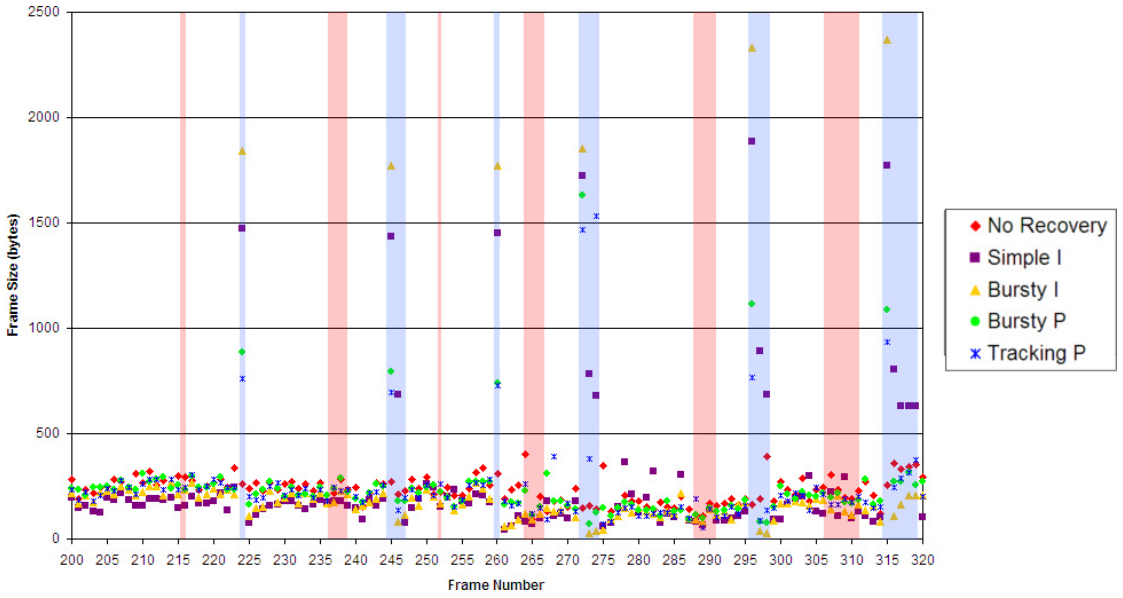
(a) No Recovery (b) Bursty I-refresh

(c) Bursty P-refresh (d) Tracking P-refresh

Figure 7: Visual comparison of recovery techniques. Shown is frame 362 of the Graduation video using loss sequence 1.



(a) PSNR



(b) Frame Size

Figure 8: Resulting PSNR and frame size for different recovery techniques on the Accident video using loss sequence 3. Pink bars indicate frames that were lost and blue bars indicate frames for which recovery strategies were prompted due to feedback.

same video and loss sequence, some knowledge is gained about how well the available bits were allocated. Table 2 supports that the bursty P-frame method and the tracking P-frame method are the best techniques and fairly equivalent. The small difference between these two methods most likely is a result of the burst lengths and the visual complexity of frames encoded as refresh frames.

Intra-frames are easier to encode than P-frames because they avoid the costs of searching for motion vectors. Thus the bursty I-frame approach may require less computation than even normal encoding. The additional complexity of implementing the bursty P-refresh method is small, as it only involves updating two values per macroblock.

As currently implemented, the tracking P-refresh method requires significantly more computation. To populate the *curr* array, every motion vector must be followed to determine which macroblocks the current macroblock depends on. It is necessary to calculate where each corner of the region predicted from lies, since this region does not necessarily align with macroblock borders. Additional complexity is added by the possible subdivision of macroblocks, resulting in even more motion vectors. Though much of this complexity could possibly be reduced by integrating these calculations with the actual encoding of inter-blocks, this method is currently the most computationally intensive.

5 Conclusion

The data loss concealment techniques developed recover from data loss with low computational complexity and use little memory. The bursty P-frame method is best suited for longer bursts while the tracking P-frame refresh handles a single frame loss with fewest bits used.

To minimize wasting limited transmission capacity, the encoder should intelligently switch between these two modes. If losses are usually isolated the tracking method would be best and otherwise the bursty method could be employed. The additional complexity of the tracking method may determine that adhering solely to the bursty P-frame method is the best course of action.

6 Future Work

There are several areas for future work:

- **Improve the performance of the the tracking P-refresh mode:**
  To reduce the number of intra-blocks, the motion vector search could be modified to only consider motion vectors that point to uncorrupted blocks. Computational improvements could also be made by integrating the calculation of *curr* array with inter-block encoding.

- **Implement the system on the cellphones:**
  This would involve replacing the simulation aspects with actual detection of lost frames, feedback, and calculation of round trip time of the network.

- **Experiment with adjusting quantization parameter for different regions:**
  Less bits would be used if we encoded the background and other areas at a lower quality in refresh frames, allowing the bit-rate to remain more constant. However, this may reduce intelligibility due to spatial and temporal predictions. This could involve incorporating other aspects of the MobileASL codec into loss recovery, such as skin detection.

7 Acknowledgments

I thank Prof. Richard Ladner, Prof. Eve Riskin and the rest of the MobileASL research group for guidance and ideas. This research uses the Open Source projects x264 and ffmpeg. I would like to thank the Mary Gates Endowment for a Mary Gates Research Scholarship. I would also like to acknowledge the NSF, Intel, Sprint and Microsoft.

8 Tables

Accident	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	23.93	25.29	25.59	25.57	25.56
Loss Seq. 2	23.01	25.01	25.48	25.40	25.39
Loss Seq. 3	19.29	24.55	25.25	25.23	25.16
Food	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	27.98	30.70	30.71	30.78	30.79
Loss Seq. 2	26.93	30.59	30.78	30.70	30.66
Loss Seq. 3	24.83	30.25	30.74	30.65	30.61
Graduation	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	23.66	26.34	26.66	26.80	26.69
Loss Seq. 2	24.76	25.97	26.50	26.49	26.45
Loss Seq. 3	21.76	25.45	26.26	26.39	26.37

Table 1: Average PSNR from simulations with all videos and loss sequences. The PSNR values for the Food video are generally higher because the video was filmed against a simple white background. Loss sequence three has a much greater effect on the video with no recovery because it contains some lengthy bursts of loss.

Accident	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	34.89	36.69	35.89	35.24	35.32
Loss Seq. 2	34.89	36.76	35.84	35.19	35.29
Loss Seq. 3	34.89	38.16	36.45	35.50	35.75
Food	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	30.19	30.99	30.60	30.75	30.47
Loss Seq. 2	30.19	31.08	30.64	30.47	30.58
Loss Seq. 3	30.19	31.79	30.89	30.68	30.72
Graduation	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Loss Seq. 1	34.33	36.16	35.39	34.66	34.75
Loss Seq. 2	34.33	36.27	35.40	34.64	34.63
Loss Seq. 3	34.33	37.94	36.21	35.07	35.15

Table 2: Average QP from simulations with all videos and loss sequences. Lower QP means the video was encoded at higher quality, an indication of better use of available bits.

Loss Seq. 1	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Average	27.46	27.81	27.77	27.35	27.36
Max	36.84	74.58	61.84	39.66	39.06
Loss Seq. 2	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Average	27.46	28.07	27.90	27.50	27.45
Max	36.84	71.16	49.59	37.50	38.22
Loss Seq. 3	No Recovery	Simple I	Bursty I	Bursty P	Tracking P
Average	27.46	27.35	27.77	27.28	27.14
Max	36.84	64.90	52.50	40.88	46.59

Table 3: Resulting bit rates (in Kbps) from encoding the Accident video with a target average bit rate of 30 Kpbs. Max equals the maximum number of bits used in a second, equivalent to the most bits used by 15 consecutive frames for 15 fps videos.

References

[1] Chang, Pao-Chi, and Tien-Hsu Lee: *Precise and Fast Error Tracking for Error-Resilient Transmission of H.263 Video*, IEEE Transactions on Circuits and Systems for Video Technology, **10**(4), (2000), 600-607.

[2] Cokely, Dennis, and Charlotte Lee Baker-Shenk: *American Sign Language: A Student Text*, Silver Spring, Md: T.J. Publishers, 1980.

[3] Ebert, Jean-Pierre and Andreas Willig: *A Gilbert-Elliot Bit Error Model and the Efficient Use in Packet Level Simulation*, TKN Technical Reports Series of Technical University Berlin, March 1999.

[4] Liang, Y. J., J. G. Apostolopoulos and B. Girod: *Analysis of Packet Loss for Compressed Video: Does Burst-Length Matter?*, Proceedings of ICASSP2003, Apr. 2003.

[5] Leonardis, Athanasios, and Pamela C. Costman: *Video Compression for Lossy Packet Networks With Mode Switching and a Dual-Frame Buffer*, IEEE Transactions on Image Processing, **13**(7), (2004), 885-897.

[6] Girod, Bernd, and Niko Färber: *Feedback-Based Error Control for Mobile Video Transmission*, Proceedings of the IEEE, **87**(10), (1999), 1707-1723.

[7] Goshi, Justin, Alexander E. Mohr, Richard E. Ladner, Eve A. Riskin and Alan Lippman: *Unequal Loss Protection for H.263 Compressed Video*, IEEE Transactions on Circuits and Systems for Video Technology, **15**(3), (2005), 412-419.

[8] Mitchell, Ross E: *How many deaf people are there in the United States?*, 2005, http://gri.gallaudet.edu/Demographics/deaf-US.php.

[9] MobileASL Research Group: *Publications*, http://mobileasl.cs.washington.edu/publications.html.

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

[11] Ong, Sylvie C.W., and Surendra Ranganath: *Automatic Sign Language Analysis: A Survey and the Future beyond Lexical Meaning*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **27**(6), (2005), 873-891.

[12] Panpaliya, Monika: *Technical Question*, email correspondence with Sprint CTO.

[13] Richter, Andreas: *Mobile videotelephony: Test of 3G telephones*, Hjlpmedelsinstitutet (HI) / The Swedish Handicap Institute (SHI), URN-NBN:se:hi-2007-07335-pdf, 2007.

[14] Stockhammer, Thomas, Miska M. Hannuksela, and Thomas Wiegand: *H.264/AVC in Wireless Environments*, IEEE Transactions on Circuits and Systems for Video Technology, **13**(7), (2003), 657-673.

[15] Stockhammer, Thomas, Dimitrios Kontopodis, and Thomas Wiegand: *Rate-Distortion Optimization for JVT/H.26L Video Coding in Packet Loss Environment*, Proceedings of the International Packet Video Workshop (PV'02), Pittsburgh, Pa, USA, April 2002.

[16] Wan, Shuai, and Ebroul Izquierdo: *Rate-Distortion Optimized Motion Compensated Prediction for Packet Loss Resilient Video Coding*, IEEE Transactions on Image Processing, **16**(5), (2007), 1327-1338.

[17] Wang, Yao, and Qin-Fan Zhu: *Error control and concealment for video communication: A review*, Proceedings of the IEEE, **86**(5), (1998), 974-997.

[18] Wiegand, Thomas, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra: *Overview of the H.264/AVC Video Coding Standard*, IEEE Transactions on Circuits and Systems for Video Technology, **13**(7), (2003), 560-576.

[19] Yu, Hong-Bin, Ci Wang, and Songyu Yu: *A Novel Error Recovery Scheme for H.264 Video and Its Application in Conversational Services*, IEEE Transactions on Consumer Electronics, **50**(1), (2004), 329-334.

[20] Zhang, Rui: *Video Coding with Optimal Inter/Intra-Mode Switching for Packet Loss Resilience*, IEEE Journal on Selected Areas in Communications, **18**(6), (2000), 966-976.