Proceedings IEEE Workshop on Mobile Computing, Santa Cruz, California, December 1994.

Asynchronous Video Coding for Wireless Transport *

David G. Messerschmitt
Fellow IEEE
EECS Department
Univ. of California, Berkeley
messer@eecs.berkeley.edu

Johnathan M. Reason
Student Member IEEE
EECS Department
Univ. of California, Berkeley
reason@eecs.berkeley.edu

Allen Y. Lao
Student Member IEEE
EECS Department
Univ. of California, Berkeley
alao@eecs.berkeley.edu

Abstract

Wireless access to continuous-media services such as video, voice, and audio is becoming increasingly prevalent. Interactive video services such as video conferencing and multimedia editing is one such service, but existing compression standards (designed for wired, circuit-switched services) are unsatisfactory for wireless packet-video services. We propose a novel strategy for video transport using a layered source coder in conjunction with a variable-QOS, multiple-substream abstraction for the transport. This abstraction addresses specifically the need to obtain simultaneously high spectral efficiency, good subjective quality, and low perceptual delay on a wireless channel. It also addresses the heterogenous transport resulting from the concatenation of a wireless access link with a broadband backbone network.

We use asynchronous video (ASV) reconstruction, running counter to current techniques, which use strictly synchronous (frame-by-frame) video processing. By doing so, we hope to achieve a perceptual delay that is much lower than the worst-case transport delay. (By "perceptual delay", we refer to the effective end-to-end latency observed by the user, for example as represented by the audio delay required to maintain lip synchronization.) By identifying packets to the transport with relaxed reliability and/or delay requirements (through the substream identifier), the transport (particularly wireless) can achieve high traffic capacity. Reasonable and promising simulation results are achieved, although much work remains on achieving significant video compression in this environment.

1 Introduction

Wireless access to wired backbone networks will be a common model for the future, resulting in a heterogeneous network infrastructure. Typically the wireless access will be the bottleneck for such a heterogeneous network, due to its scant bandwidth allocation, serious multipath fading and propagation impairments, and interference. Thus, our focus in this research is the provision of video services over this heterogeneous network, with emphasis on the issues raised by the wireless access.

In continuous-media services such as video, the only meaningful criterion for evaluation of the service quality is the subjective quality. The primary subjective impairments will be time jitter, latency or delay, and artifacts introduced by loss mechanisms in the compression and the transport. Another important consideration will be cost, as represented in part by the traffic capacity consumed for the service, particularly on the wireless access link.

In this mix of issues, we believe that delay has been largely overlooked. For interactive applications such as video conferencing and multimedia editing, low delay is crucial to user acceptance, and yet current trends run directly counter to achieving low delay. Existing examples of wireless audio/voice access, such as digital cellular, use transcoders (conversions from one audio compression standard to another) in the basestation, which add significant delay. A similar approach for video would have the same problem. On variable-delay transport, typical of packet/cell networks, existing video compression algorithms synchronously reconstruct video frames, implying a delay that includes the *worst-case* transport delay plus any compression/decompression delay.

A primary goal of this research is to minimize the *perceptual delay* for video transport. By perceptual delay, we mean the delay perceived by the user, for example as would be indicated by the audio delay required to achieve lip synchronization in a video conferencing application. We reduce this perceptual delay by two related techniques. First, we code the video in such a way that transcoders are not required within the network; that is, we transport the video transparently from network edge to edge. Second, we do this in such a fashion that the perceptual delay is *less* than the transport worst-case delay plus compression/decom-

*This research was supported by the Advanced Research Projects Agency Contract J-FBI-93-153, Tektronix, the University of California MICRO Program Grant #94-052, and Asahi Semiconductor.

pression delay.

The second objective is achieved by reconstructing the video presentation at the receiver asynchronously. That is, we identify information within the video that can be delayed from its original frame to later frames, without significant artifacts being introduced. For example, information elements in areas of the screen with low motion are more tolerant of delay than areas of high motion. As another example, the high-resolution information in areas with little motion can be delayed relative to the low-resolution information. (This is closely related to the well-known technique of progressive image transmission.) When the video is asynchronously reconstructed, there is no longer a clear *objective* definition of delay, but rather we have to rely on the vague notion of perceptual delay, a quantity that can be determined only by subjective testing.

Decoupling perceptual delay from worst-case transport delay leads to another opportunity. By allowing the worst-case transport delay to increase (without affecting perceptual delay), we can actually increase the network traffic capacity. This will be particularly advantageous on wireless access links.

2 Substream Abstraction of Transport

In [1] a basic architecture for heterogeneous continuous-media networks is proposed, one of the key features of which is the *substream abstraction* for the transport network. While constructs similar to substreams (typically called *flows*) have been proposed for future Internet protocols, [1] argues that substreams are crucial to obtaining high traffic efficiency in networks with wireless access links. Figure 1 illustrates the substream abstraction.

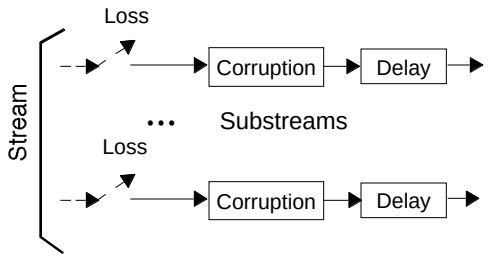

Figure 1: Substream Abstraction of Transport

For a particular audio or video *stream*, the transport provisions *substreams (*through a link-layer protocol). Each substream has a quality of service (QOS) specification of loss, corruption, and delay characteristics. The QOS of the substreams are different, as negotiated at session establishment. In addition, the collection of substreams comprising the stream has a *joint* specification of rate parameters (substreams have correlated rates since they originate from the same continuous-media source). The substreams provide a mechanism by which a continuous-media service can specify different QOS characteristics for different information elements. This in turn becomes the mechanism by which the benefits of *joint source/channel coding* to increase traffic capacity are achieved without a close coupling of source and channel coding design. Since substreams can be transported intact across heterogeneous subnetwork segments, joint source/channel coding on downstream links does not require transcoders. Further, encryption does not interfere with joint source/channel coding, as long as the substreams are independently encrypted. All these considerations are discussed in more detail in [1].

For typical transport subnetworks, there is a coupling of reliability (loss and corruption) and delay QOS, for fixed resource utilization. This is because available techniques for reducing loss and corruption, such as larger buffers, forward error-correction coding, interleaving, retransmission, etc., all increase delay. This is illustrated by the simple example of an *M/M/1/K* queue with K waiting positions (this doesn't represent any real network, but does illustrate the point). For this queue, the loss probability P_L (probability that an arriving packet is lost because it encounters a full buffer) is

$$P_L = \frac{(1-a)a^K}{1-a^{K+1}} \qquad \text{(EQ 1)}$$

where $0 \le a < 1$ is the offered load (or traffic utilization) [2]. The maximum delay through the queue is proportional to K. Thus, for constant offered load a, P_L decreases exponentially with the maximum permissible delay.

More important for present purposes is the observation that for typical networks with fixed loss/corruption, a relaxed worst-case delay requirement results in higher traffic capacity. This characteristic is observed in the *M/M/1/K* queuing example. More relevant for our application is an interference environment characteristic of wireless access links. If a given packet has more flexibility in its transmission time, we can choose an opportune time when the interference is temporarily low, giving an additional mechanism to smooth the traffic and increase traffic capacity.

In the absence of substreams, the delay characteristics of all data in a continuous-media stream is the same. With substreams, data can be segregated into parts with different delay characteristics, and the network can exploit the relaxed delay characteristics of some substreams to achieve higher overall traffic capacity (this is one manifestation of joint source/channel coding, albeit one not previously emphasized). From the perspective of the continuous-media coder, there is control over which data arrives earlier than the worst-case delay, and thus this low-delay data can be exploited.

3 Overview of Asynchronous Video

Video coding for substream transport requires the partitioning of semantic components of the compressed video into substreams with different QOS characteristics. Relaxing the QOS of each substream to the extent possible consistent with subjective quality objectives allows the transport to achieve the highest traffic capacity, providing it allocates its internal resources (buffer capacity, power, bandwidth, redundancy, etc.) to provide no higher QOS than necessary for each substream.

Since substreams have different delay characteristics, they arrive at the receiver asynchronously. It is possible to re-synchronize substreams at the receiving terminal, but to do so will result in transport delay (including re-synchronizing delay) that is uniform and characteristic of the highest delay (typically highest reliability) substream. We choose a different approach, in which the video is asynchronously reconstructed at the receiving terminal. Our goal is to achieve a perceptual delay that is representative of the *lowest-delay* substream, rather than the highest-delay substream. We define *asynchronous video* (ASV) as video coding for substream transport, where we decode the video presentation for display without re-synchronization of the substreams.

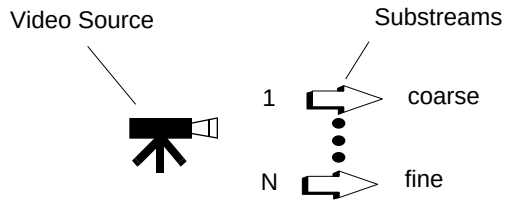

Figure 2: Substream Abstraction for Video

Figure 2 illustrates an abstracted view of ASV. The video source is segmented into blocks (rectangular regions of pixels). Each block is further segmented into a layered representation with different resolutions. These resolutions are transported by different substreams, using motion estimation in addition to resolution in determining the appropriate substream. Within this framework, we can make the following intuitive observations:

- For blocks in high motion regions, relatively low resolution is required, because the motion will subjectively mask the resulting quantization error artifacts. For the same reason, these blocks can tolerate a relaxed transport loss/corruption requirement.
- The perceptual delay will be dominated by the transport delay of the high motion regions. For example, in video conferencing the high motion area will typically be the person's lips, and the audio delay for lip synchronization (which for this application we take to be the perceptual delay) will be representative of the delay of this high-motion area.
- Blocks in low-motion areas will be more sensitive to loss/corruption in both coding and in transport, but less sensitive to delay. In particular, we hope that these blocks can be moved to later frames (relative to the high-motion blocks) without introducing undue artifacts, because they are changing little from frame to frame.

Nature is kind to us, since blocks with tighter delay requirements have relaxed transport loss/corruption requirements, and *vice versa*. This is precisely the typical trade-off between delay and loss in subnetworks operating with constant resource utilization.

To reiterate, with this segmentation of video blocks into substreams, we expect intuitively that the low-motion, full-resolution blocks can be delayed relative to the high-motion, low-resolution blocks, and high-motion, high-resolution blocks can be discarded entirely. Further, we expect that the perceptual delay will be dominated by the high-motion blocks, which are transported with the lowest delay. Thus, the perceptual delay will be dominated by the lowest-delay substream, rather than the worst-case transport delay. To the extent that the other substreams' delay requirements can be relaxed, the network traffic capacity can be increased. We believe that this opportunity has the greatest significance on wireless access links, although as noted before we propose to use this video coding technique transparently across the entire edge-to-edge connection, including backbone network, to avoid the added delay (and other problems [1]) introduced by transcoders.

We have not specified the way in which the layered resolution is achieved. There are many ways this could be done, but in the present realizations we have chosen a simplistic subband decomposition. Further, our current ASV coder does not utilize other known compression techniques like motion compensation, but it may be possible to introduce these concepts later. The video coding simulations reported here are very simple, designed to illustrate the perceptual delay benefits of ASV.

4 ASV Codec Design

We begin this section with a general description of our ASV codec, emphasizing the substream partitioning and asynchronous reconstruction of video, rather than the specific signal processing of each component. We subsequently discuss the details of the transmitter coder and receiver decoder.

4.1 General description

Figure 3 shows a block diagram of the coder design. In the top forward-path, frames are first processed through a *transformation* component, which transforms the image into a suitable domain (e.g., frequency domain via subband coding) and representation (i.e., transformed-image blocks). The *substream coder* then performs quantization of the transformed blocks to provide layered resolutions and, most importantly, partitions the semantic content of the compressed video-stream into substreams. This partitioning is controlled by a bank of motion-driven, *finite-state machines* (FSM), one for each spatial block-region of the image. *Motion detection* is performed in the bottom forward-path, where each motion estimate is quantized into one of three motion levels: no motion, low motion, or high motion.

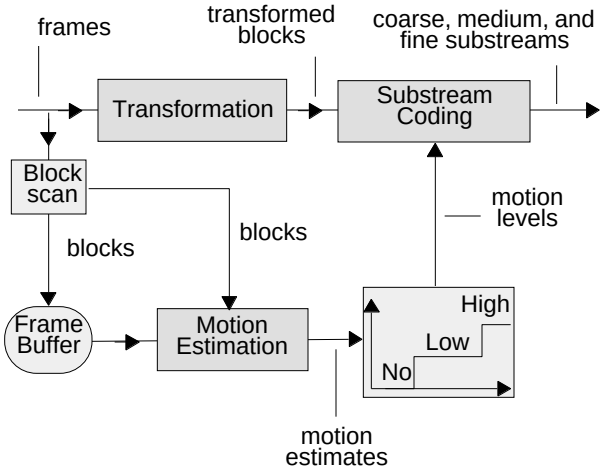

Figure 3: Block Diagram of ASV Coder

Figure 4 illustrates the substream-coder FSM for a single block. The FSM updates its state according to the level of motion determined for a given block region. In particular, the FSM makes a transition to the COARSE state whenever a high level of motion is detected since fine details are less visible in a high-motion scene. For low levels of motion, the FSM makes a transition into the MEDIUM state since some fine details become visibly more noticeable when the scene is moving slower. When virtually no motion is detected, then the FSM makes a transition to the FINE state, since most fine details are visible when the scene is not moving. If there is still no detected motion in this block at the next frame, then the FSM makes a transition to the IDLE state, where no data is transmitted. The FSM remains in IDLE until motion is detected again.

For each block, a message to the decoder is formulated with the following information:

- *Substream identifier*, which identifies to the transport the QOS expected for this data and to the decoder the type of data (coarse, medium, fine, etc.). (Yun and Messerschmitt in [5] discuss how QOS can be provided on a per-substream basis for a wireless subnetwork using a code-division multiple access transport.)
- *Temporal locator*, in the form of a *frame sequence number*.
- *Spatial locator*, specifying the X-Y coordinates of the block.
- Representation of the pixels in either a fine, medium, or coarse format.

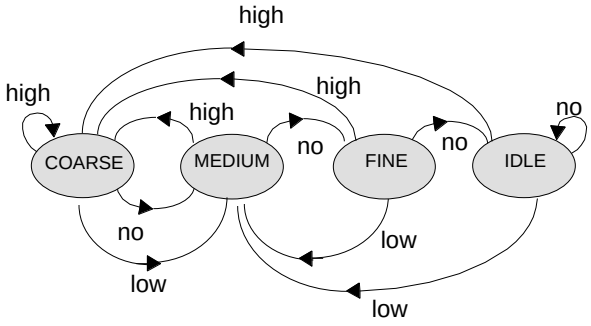

Figure 4: FSM of Substream Coder

The specific representation of pixel information is flexible. Often, video data transmitted at a certain resolution naturally decomposes into distinct parts. For example, data transmitted via a substream of high resolution might consist of corresponding low-resolution data in addition to fine-resolution enhancement information. In our ASV codec (see Section 4.2), each substream is mapped to a set of subbands where substreams of increasingly fine resolution are assigned additional subbands on top of those allotted for substreams of coarse resolution. In packetizing ASV for transmission over a specific substream, it may be natural to fragment information among multiple messages according to whatever natural decomposition may exist for that information.

Figure 5 shows a block diagram of the ASV decoder. First, the substream-decoding component reconstructs the video stream from the incoming substreams by maintaining certain ordering relationships. While we do not attempt to maintain the exact temporal relationship of the blocks, nor do we re-synchronize the substreams, it is desirable to maintain certain ordering relationships among blocks to minimize visible artifacts. Like the partitioning process of the substream coder, the reconstruction process is controlled by a bank of FSMs to impose ordering relationships. The inverse transformation component then produces the actual image representation.

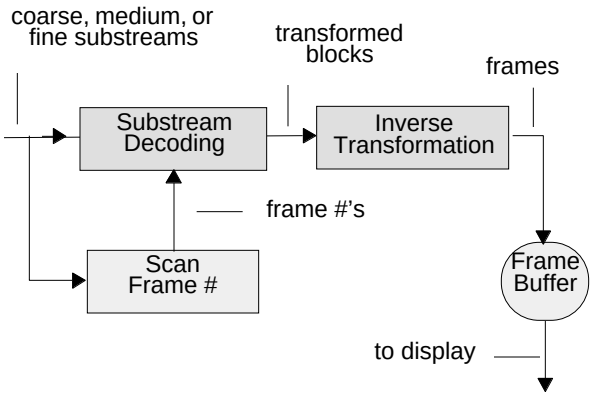

Figure 5: Block Diagram of ASV Decoder

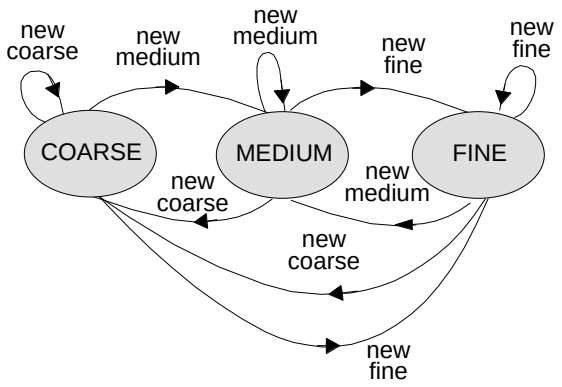

Figure 6: FSM of Substream Decoder

Figure 6 illustrates a substream-decoder FSM for a particular block. This FSM updates its state according to the frame sequence number for each block of that block region. In particular, the FSM makes a transition to the COARSE state whenever a *new* (higher frame sequence number) block for this region arrives on the coarse substream. Likewise, the FSM makes a transition to the MEDIUM or FINE states whenever a new block for this region arrives on the medium or fine substreams, respectively. To maintain proper ordering, the FSM retains its current state whenever a *stale* (smaller frame sequence number) block arrives on a given substream for this block. Stale blocks are simply discarded without display. In summary, each FSM in the substream decoder ensures that only the most recent image data for each block region is displayed; stale data is always discarded.

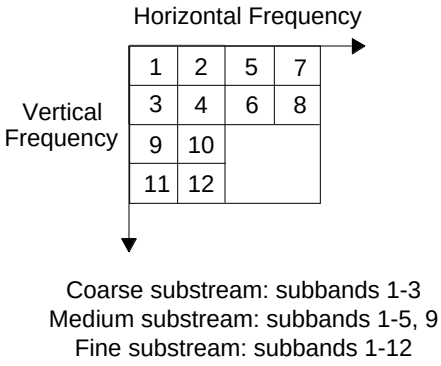

Figure 8: Mapping of Substreams to Subbands

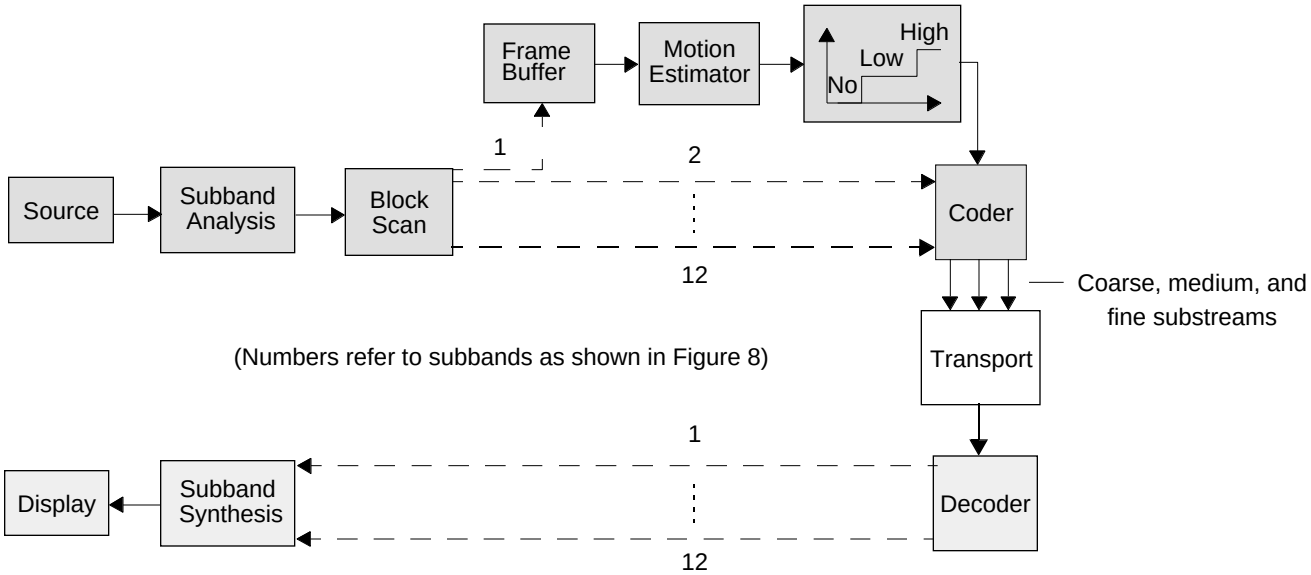

Figure 7: ASV Codec Implementation Using Subband Filtering

4.2 Codec implementation

We now describe an implementation for our codec that concretely illustrates this general approach. Results of experimentation performed on this codec design follow in the next section.

Figure 7 shows a top-level overview of the codec design. A subband decomposition stage consisting of two stages of 2-D QMF filter banks accepts eight-bit grayscale video as input [4]. It produces frequency coefficients in the form of twelve distinct frequency bands as represented in Figure 8, where the four bands belonging to the highest-frequency quadrant are discarded because they don't contribute significantly to image quality.

These frequency coefficients are then scanned into individual blocks. Specifically, we may represent each original grayscale image as 8 x 8 blocks. After passing through the subband decomposition stage, there are a set of twelve subband images that have height and width a quarter of the respective dimensions of the original image (due to the down sampling performed twice along each dimension). These subband images would be considered on the basis of 2 x 2 blocks of frequency coefficients that spatially map to their original 8 x 8 grayscale blocks.

Motion estimation is performed by calculating the mean-square difference of the lowest-frequency subband (hereafter called LFS), numbered "1" in Figure 8, with that of the previous frame. As explained earlier, motion is quantized to one of three levels. For transform-based coding, motion estimation should be done in the transform domain and not the spatial domain (in this case grayscale data). In a general implementation that could draw upon many existing compression techniques prior to the block scan stage in Figure 7, we would need to define a metric that characterizes levels of variation in transform domain data. Significant changes in transform domain data would correspond to significant motion from a subjective point of view after it is processed by the decoder and inverse transform stage. Although there may be a strong correlation between detected motion levels in the spatial and transformed domains, this relationship is unreliable. For example, in our experiments, we often note instances where blocks judged to have low motion in the grayscale domain actually result in high motion levels in the frequency domain after subband filtering.

The coder accepts the transform images for the twelve subbands and maintains a set of FSMs obeying the semantics of Figure 4. From a given block region's current state and its current level of motion, the coder determines the resolution at which that block region will be transmitted. Figure 8 shows how each level of resolution or substream is mapped to a set of subbands for transmission.

The network transport services the video blocks according to the negotiated QOS substream contracts established at setup time. Figure 6 generalizes this in the form of the transport accepting as input the three video data substreams. From here, the blocks arrive asynchronously at the decoder block.

Section 4.1 discussed how a block's semantics could be fragmented among possibly multiple messages. In our implementation, we assume that a block's data is fragmented so the each message transports data from one of the subbands comprising the block's substream assignment.

The decoder maintains a state subband image for each of the twelve subbands. For each block region, there is also a notion of *current* frame sequence number. When triggered to produce an output frame, the decoder observes on a per-block basis the current frame number received on the LFS. For each block, it yields its current state data for the LFS as well as the current data it may have for the other subbands, as long as their frame number matches that of the LFS data. The rationale is that the LFS provides a low-resolution approximation to the block, and only data from other subbands originating from the same frame as the LFS data should be used to enhance the resolution of that block region.

These considerations imply a natural set of rules for processing incoming video blocks. First, we consider blocks from subbands besides the LFS. An input block for a certain subband is compared with the frame numbers for the matching subband and also the LFS. If its frame number is less than that of the matching subband, it is stale and is discarded. If its frame number is greater than that of the matching subband and less than or equal to the LFS frame number, it is incorporated into the state data for that subband immediately, erasing the old information for that block. The current frame number for the region is updated. If its frame number is greater than that of the matching subband and greater than the LFS frame number, it is buffered for future use without affecting any state or frame sequence information.

For input blocks of the LFS, the procedure is simpler. If its frame number is greater than that of the current LFS state data for that block, it is incorporated; otherwise, it is discarded. If data from the LFS is accepted, the decoder observes if there are any blocks from other subbands that were buffered due to their frame number being greater than that of the LFS at the time they arrived. Quite possibly at this point, a buffered block's frame number will be less than or equal to the newly arrived LFS block, and thus, it can be written into the state information for its subband.

The decoder achieves its purpose of asynchronously processing video in a surprisingly straightforward fashion. It can use the frame identifier of input video blocks to sensibly order and process data.

From here, subband data produced by the decoder is

synthesized into a displayed representation of the original image with two banks of inverse QMF filters.

5 Empirical Results

QMF filters for subband decomposition were implemented with eight-tap FIR filters [4]. Subband coefficients were represented as unsigned integers with a certain allocated number of bits and offsets to make possible representation of negative values. Bit errors were assumed to occur uniformly, corrupting each bit in the unsigned integer fields with equal probability.

For our experiments, we chose to independently investigate the effects of perceptual delay, delay-bound requirements, and channel errors on the subjective quality of video.

5.1 Perceptual-delay experiments

To measure the perceptual delay, we used a 15 frames/sec *talking-heads* sequence, which we refer to as SEQ. This sequence has significant high-motion content (i.e., head and lip movements), low-motion content (i.e., a slowly panning overlay), and stationary content (i.e., a motionless background).

We simulated ASV coding of SEQ with different delay requirements for each substream. For this experiment, we chose the delay requirement for the coarse substream to always be less than the 66 ms frame time for SEQ; that is, we require that all high-motion, coarse-resolution blocks from the same frame arrive at the receiver within 66 ms. For the other substreams, we assigned delay requirements greater than 66 ms. We then played back the processed data with its corresponding audio to observe *lip sync*.

From our observations, we could not differentiate the lip sync of the processed video from the original, and the only noticeable degradation in image quality was the reduced image resolution in the high-motion areas. Therefore, we concluded that the perceptual delay for SEQ was indeed dominated by the coarse substream (i.e., the substream that transports the high-motion areas). We were able to verify this by keeping the delay of the coarse substream constant (and always greater than 66 ms) and, then, varying the delay requirements for the medium and fine substreams. Under these constraints, lip sync was always disrupted regardless of the delay requirements assigned to the medium and fine substreams.

In summary, this empirical evidence suggests that we can indeed relax the delay requirements on the medium and fine substreams without introducing objectionable artifacts, but there are bounds on how much we can relax these requirements. The next section presents some empirical results to quantify these bounds.

5.2 Delay-bound experiments

The effects of relaxed delay bounds on substreams besides the coarse substream offer the transport considerable flexibility in meeting the needs of these substreams in terms of error protection, etc. However, they may result in artifacts due to the frame displacement of coarse, medium, and fine information within a block, and also frame displacement of different blocks, relative to their original relationship. The question is whether subjective quality degradation can be held in check while the transport delay bounds for certain substreams varies as much as multiple frame intervals.

To measure how relaxed the delay bounds can be on the medium and fine substreams, we used a 24 frames/sec action sequence, which we refer to as MOT. This sequence also had significant high-motion, low-motion, and stationary content.

For this experiment, we kept the delay requirement for the coarse substream below the 42 ms frame time for MOT and varied the delay requirements for the other substreams until the video quality became objectionable. From our observations, we concluded that the medium substream could tolerate a delay up to one frame time and the fine substream could tolerate a delay up to three frame times. In other words, low-motion, medium-resolution data would not arrive in time for the *next* output frame but for the one after and stationary, fine-resolution (or full-resolution) data could be delayed by two additional frames.

Due to the frame displacement of blocks transmitted along the medium and fine substreams, certain areas of the screen are expected to display out-of-date information, manifested as “shadows” or stale regions left behind by moving objects. Even with a high-motion sequence such as the MOT sequence, though, these effects are minimized by prudent selection of motion thresholds in the coder. For regions of the image with appreciable motion, it was required to set motion thresholds so that they would be transmitted on the coarse substream with a low delay guarantee to avoid shadow effects. Conversely, the coder was also able to effectively determine which image regions had low activity and to transmit these on the medium and fine substream channels without incurring annoying artifacts in the output.

5.3 Error-resiliency and bit rate experiments

It was also of interest to test the resilience of the video quality to error impairments. As discussed before, coarse-resolution substream is more tolerant of errors than the other substreams of finer resolution due to its high motion content. We found that for the MOT sequence, the high-motion, coarse-resolution substream could endure an error

rate approximately three or four orders of magnitude higher than the stationary, fine-resolution substream for roughly equivalent subjective impairment. In the future, it will be of interest to test algorithms for error detection/correction instead of always making use of corrupted data. With the absence of any form of error control, video quality breaks down under error rates on the coarse substream in the range of 1×10^{-4}, a fairly reasonable error probability to sustain on wireless links.

Figure 9 provides an example of the bandwidth gains resulting from our approach. For the MOT sequence displaying 320 x 240 frames of eight-bit grayscale data at 24 frames/sec, the average bit rate produced by the coder is about 5 Mb/s while it is offered a raw bit rate of 14.7 Mb/s. ASV coding typically yields gains in the range of 2:1 to 3:1 for various test sequences. It should be noted that the work we have discussed has not focused at all on attempts to merge ASV coding with popular compression techniques, but doing so should result in significant reductions in bit rate. Nevertheless, our primary goal has been to maximize subjective quality of received video subject to channel limitations. This goal is not the same as the minimization of bit rate, in view of the severe impairments of the wireless channel.

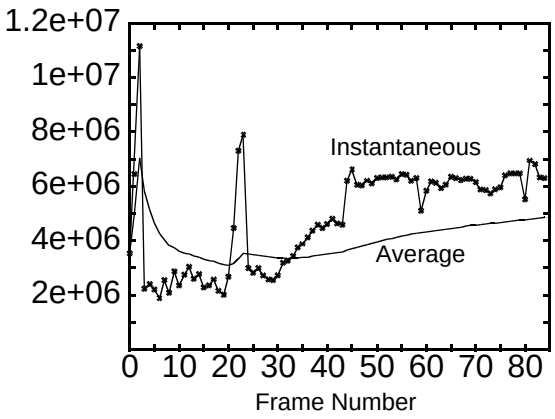

Figure 9: Bit Rate for MOT Sequence

6 Conclusions and Future Work

We consider our work on actual ASV coding and decoding to be very preliminary. Since we could see no path from existing video compression standards like MPEG directly to an ASV version, we instead started from first principles. Thus, our preliminary ASV codec might be considered a form of simple conditional replenishment codec, as it achieves compression only through discarding fine-resolution information in areas of high motion, and transporting low-motion information less often (or only once, where there is no motion at all). The novelty of our codec is in the temporal dimension, where we are moving low-motion and stationary, high-resolution information to later frames, as long as the resulting artifacts are not subjectively objectionable, and thereby gaining traffic capacity advantages.

In future work, we hope to demonstrate more sophisticated versions of ASV codecs, in particular, incorporating other known video compression techniques such as vector quantization and motion compensation. Additional work is needed in verifying ASV video over more accurate transport models. Our efforts to date have focused on demonstrating the perceptual delay advantages of ASV, and the error models used are simplistic. Achieving the best combination of subjective quality and traffic capacity on wireless links will require considerable additional work.

We are currently designing a real-time prototype of ASV transport by wireless CDMA in the context of the Infopad wireless multimedia computing project at Berkeley.

References

[1] P. Haskell and D. G. Messerschmitt, “A Signal Processing Perspective on Networking for Continuous-Media Services”, submitted to *IEEE/ACM Transactions on Networking*, November 1994.

[2] L. Kleinrock, *Queueing Systems. Part I*., Wiley, 1976.

[3] J. T. Kim, H. J. Lee, and J. S. Choi, “Subband Coding Using Human Visual Characteristics for Image Signals,” *IEEE J. Select. Areas Commun*., vol. 11, pp. 59-64, Jan. 1993.

[4] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Inc., 1983.

[5] L. C. Yun and D. G. Messerschmitt, “Power Control for Variable QOS on a CDMA Channel”, *Proceedings IEEE MILCOM*, vol. 1, pp. 178-182, Oct. 2-5, 1994.