An Experimental Evaluation of Video Coding Algorithms for ATM-based Networks

Azam Syed
Carey L. Williamson
Department of Computational Science
University of Saskatchewan

February 10, 1993

Abstract

Recent advances in network technology have increased the bandwidth available on high speed networks, and enabled the development of new network applications such as real-time video teleconferencing and medical imaging. In networks such as Broadband-ISDN, the primary goal is to integrate services such as video with existing information services, such as voice and data. Coding techniques for packet video transmission are necessary to effectively develop and integrate these applications.

In this paper, we present an experimental evaluation of several existing coding methods designed for packetized video transmission in an ATM-based network environment such as B-ISDN. Four evaluation criteria are established: compression ratio, picture quality, processing requirements and robustness. Four coding algorithms are then quantitatively evaluated with respect to these criteria, using experimental measurement and simulation.

1 Introduction

The demand and desire to integrate services such as voice, video and data are the major motivating factors behind B-ISDN (Broadband Integrated Services Digital Network). With B-ISDN, a single integrated high speed network will be able to support a wide spectrum of current and future services with widely varying transmission rates and service requirements.

ATM (Asynchronous Transfer Mode) has been proposed as the preferred packet transport technique for B-ISDN [8]. ATM is a high-bandwidth, low-delay packet-switching and multiplexing technique. Packet switched transmission is favoured since it provides higher utilization of the channel capacity and better methods for error recovery. In ATM, bandwidth is divided up into small fixed-size packets called *cells* that are allocated to connections on demand. Each cell contains a 5-byte header, and a 48-byte data payload. ATM provides a unified means by which all types of traffic, both connection-oriented as well as connectionless, can be transported on a common network.

In a fully integrated ATM-based network environment, video traffic is clearly one of the most demanding network applications. Video traffic typically requires several orders of magnitude more bandwidth than voice or data services (e.g., raw full motion video requires up to 100 Mbps). Furthermore, the service requirements for video traffic (with respect to delay and loss sensitivity) are considerably different than the service requirements of conventional data services. Data traffic is usually highly loss sensitive but can tolerate some delay. Video traffic is usually highly delay sensitive but can tolerate small amounts of information loss.

Video coding algorithms are the key to the successful integration of video services into ATM-based networks. Coding techniques help to reduce the bandwidth requirements of video by an order of magnitude or more. With the use of high compression methods and error correction techniques, most of the service requirements for video traffic can be satisfied.

Many different video coding algorithms are proposed in the literature [2, 4, 6, 7, 11]. For ATM-based networks, variable bit rate (VBR) video coding methods are frequently favoured over constant bit rate (CBR) coding methods, since different VBR video sources can be statistically multiplexed, thereby improving the channel efficiency [5]. VBR coding provides variable bit rate video with constant video quality instead of constant bit rate video with variable video quality [5].

Many factors must be considered in the choice of a video coding algorithm. For example, one must consider the compression achieved, the picture quality, the complexity of the video codec, as well as its cost. Tradeoffs exist between many of these factors. Perhaps one of the most important considerations in the choice of a video coding algorithm is its close relationship with the underlying network. That is, one should consider the effect of the video traffic on the network (e.g., bandwidth requirements, and the burstiness of the traffic generated) as well as the effect of the network on the video traffic (e.g., the effect of network-level packet loss or delay on the video quality).

The purpose of this paper, then, is to quantitatively evaluate several currently proposed video coding techniques, with particular emphasis on their suitability for ATM-based networks. We establish four evaluation criteria: compression ratio, processing requirements, picture quality, and robustness. Two of these criteria (processing requirements and picture quality) represent factors important to the end users of video applications, and the other two criteria (compression ratio and robustness) represent factors important to the network provider. Four coding algorithms are then quantitatively evaluated with respect to these criteria, using experimental measurement and simulation. The algorithms evaluated are recursive block coding [4], JPEG [11], MPEG [7], and a pixel scattering technique proposed by Turner and Peterson [10]. The algorithms have been implemented in software on a SparcStation IPC and experimentally evaluated on a test suite of images. Simulation is used to assess performance effects for packetized video transmission in an ATM-based network environment.

The remainder of this paper is organized as follows. Section 2 introduces the service requirements for packet video and discusses the importance of video coding. Section 3 describes the four coding algorithms used in our experiments. Section 4 describes the performance experiments conducted, and the evaluation criteria used to evaluate the video coding algorithms. Section 5 presents our experimental results. Finally, Section 6 presents our conclusions and suggested directions for future research.

2 Packetized Video

2.1 Video Service Requirements

Video traffic can be broadly classified into real-time and non-real-time services. In real-time video applications, video traffic has very strict delivery time constraints. Examples of real-time applications are teleconferencing, interactive medical imaging, and dynamic radiation therapy planning. In non-real-time applications, such as image visualization or image transfer applications, the images are transferred on user request and have no hard real-time constraints. Examples of non-real-time applications are videotext services, photography editing, and X-ray imaging applications. The traffic generated by these applications is usually highly sensitive to loss, but can tolerate some level of delay.

The service requirements necessary to support video applications (particularly real-time video) are significantly different from those required for data and voice communication. In particular, video traffic has stringent requirements for bandwidth, latency, jitter, and synchronization.

- **Bandwidth**: Video applications require very high bandwidths, several orders of magnitude higher than that of voice traffic. For example, full motion uncompressed video transmission requires bandwidths up to 100 Mbps. High Definition Television (HDTV) requires bandwidths up to 1.2 Gbps.

- **Latency**: Real-time voice and video applications have very strict latency requirements (on the order of tens of milliseconds). The packets have to arrive before their play time.

Late packets are equivalent to lost packets. Retransmission of lost packets is out of question because of the end-to-end delay incurred.

- **Jitter**: Variations in packet arrival times are referred to as packet delay jitter. Delays in interarrival time complicate image reconstruction, as well as buffer management. Since long delays can impede information exchange, such delays must be avoided.

- **Synchronization**: Voice and video signals have to be synchronized at the receiver end to provide lip synchronization. Synchronization of multiple information channels should be provided in order to maintain synchronized delivery of information in applications that involve multi-site and multi-point communication.

To help meet these service requirements, different methods of video coding and compression are used.

2.2 Importance of Video Coding

The basic objective of video coding is to reduce the number of bits that need to be transmitted over the network. Video coding is important because it reduces the requirements for bandwidth, processing, and storage.

- **Bandwidth**: Compression of digital data reduces the bandwidth requirements significantly. Compressed full motion video transmission requires about 2 Mbps instead of 100 Mbps [1]. Compressed HDTV requires 100 Mbps instead of 1.2 Gbps.

- **Processing**: Compression reduces the number of bits that need to be processed by the receiver station. The fewer bits to be processed, the lower the overhead for packet processing (i.e., packetization, depacketization, checksumming, segmentation and reassembly).

- **Storage**: Compression considerably reduces the storage requirements for video data. One uncompressed NTSC-standard colour image (640x480 pixels with 24 bits of colour information per pixel) requires 0.9 Megabytes of space. At the broadcast standard rate of 30 frames/sec, a database of a one minute video clip (1800 640x480 pixel images) requires over 1.6 Gigabytes of storage space. Compression is essential to make storage of video images practical.

2.3 Video Coding Methods

Various coding schemes have been developed and proposed for packetized video. The basic coding scheme considered for real-time video applications is *source coding*. In source coding, the sender encodes the source material for transmission using motion prediction or compensation techniques. The receiver decodes and reconstructs the image. The basic objective of source coding is to remove the redundant spatial and temporal information [12].

There are two categories of source coding: *intraframe* coding and *interframe* coding. Intraframe coding is used for single (still) images. Interframe coding is used for motion video.

In intraframe coding, each video frame is encoded based on the information available within that frame. The most widely used approaches in intraframe coding are transform coding, recursive block coding and predictive coding. Intraframe coding typically produces compression ratios of 2:1 to 30:1.

In interframe coding, the current video frame is encoded based on the information available from the previous frame. Since video is composed of a sequence of images taken at short intervals, similar information is often present in successive frames. Interframe coding exploits this property of video [7], offering high compression ratios (e.g., 50:1 to 200:1). The interframe coding technique is used in the MPEG and H.261 standards [3].

Compression algorithms can also be classified as *lossless* or *lossy*. In lossless algorithms, the image reconstructed from the compressed image exactly matches the original image. In lossy algorithms, the reconstructed image is only an approximation of the original image. Lossy coding algorithms typically produce better compression than lossless algorithms.

3 Video Coding Algorithms

Video coding schemes are used to reduce the service requirements for packetized video, making the job of the network much simpler in handling congestion control, packetization delays, jitter, synchronization, and real-time scheduling. Four examples of video coding algorithms are described in this section.

3.1 Recursive Block Coding

In Recursive Block Coding (RBC) [4], an image is initially segmented into blocks (subimages) of a fixed size, and then these blocks are recursively subdivided into subblocks of uniform colour or intensity. The RBC algorithm uses a data structure called a *quad tree*, where the image (or subimage) forms the root of the tree and each node of the tree represents one block (or subblock) of the image (see Figure 1).

The decision whether to subdivide a block or not is based on a uniformity criterion that measures the difference among the pixels in a block. The uniformity measure computed for a block is compared against a threshold parameter that determines the allowable error in the encoded image. If the uniformity criterion is not met, then the block is subdivided repeatedly using the above process until all subblocks satisfy the uniformity criterion. Examples of uniformity criteria discussed in the literature are maximum absolute value and the mean

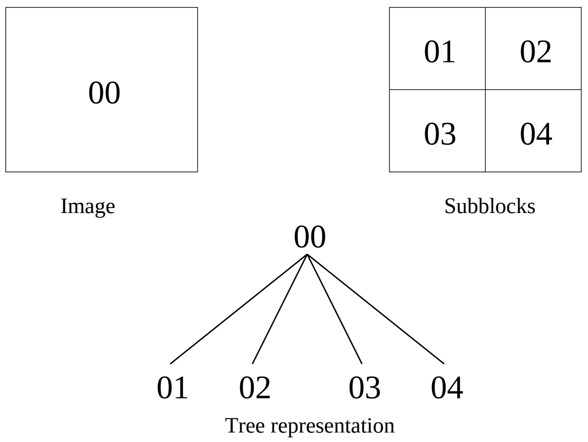

Figure 1: Recursive block coding

residual energy of the block [4]. RBC is a lossy coding algorithm.

Compression is achieved in the RBC algorithm since colour information need only be transmitted once for each block, rather than once for each pixel in a block. For every uniform block, the coordinates of the block are transmitted, along with a pixel value to represent the block's colour. Decoding is a simple process, as the coordinates and the associated pixel value for each block are used to reconstruct the original image.

In our implementation, the entire image is initially subdivided into blocks of 16x16 pixels. The uniformity criterion used is *mean residual energy*, defined as the mean squared deviation of each pixel in an NxN block from the mean of the four corner points of the block. Mathematically,

$$MRE = \frac{\sum_{i,j=0}^{N} y(i,j)^2}{(N+1)^2 - 4}$$

where $y(i,j)$ is the *residual* (i.e., the difference) between the pixel value at location (i,j) in the block, and the mean of the four corner points of the block. If the mean energy of the block is less than or equal to the threshold, then the block is considered uniform. Otherwise, the block is recursively subdivided into four square subblocks of equal size. The smallest subblock allowed in the algorithm is 2x2 pixels. Each subblock requires 8 bytes of information: 2 bytes for x-coordinate, 2 bytes for y-coordinate, 1 byte for block size, and 3 bytes for RGB colour information. Six subblocks fit in each ATM cell.

3.2 JPEG

The Joint Photograph Experts Group (JPEG) has established standards and algorithms for the compression of still images [11]. The JPEG method provides high compression ratios and high picture quality.

A JPEG encoder works as follows. First, the input analog signal is digitized and the source image is divided into non-overlapping blocks of 8x8 pixels. Each block undergoes a transformation using discrete cosine transform (DCT) that results in an 8x8 transformed matrix of the original image block. The resulting transform matrix (64 DCT coefficients) represents the relative signal strength of the input sample. The DCT coefficients are then quantized (discretized), reducing the number of bits required to represent each coefficient. The quantization step makes JPEG a lossy algorithm. The quantized coefficients are then run length encoded using either a zigzag encoder or a difference encoder, resulting in a one dimensional array. Finally, Huffman coding is performed on the resulting coefficients from the zigzag encoder. Huffman coding reduces the number of bits needed to represent the data without losing any further information. JPEG decoding is the reverse of the encoding process.

In our experiments, the Independent JPEG Group [11] implementation of JPEG is used for evaluating the algorithm. The JPEG algorithm is implemented in software and supports the sequential compression process. The implementation is as described above.

3.3 Pixel Scattering Method

A pixel scattering algorithm has been proposed by Turner and Peterson [10] for transferring images in satellite imagery applications. Their algorithm is designed to be robust to non-negligible levels of packet loss (e.g., 1-10%) in the network, without requiring end-to-end retransmission of lost packets. Their approach exploits the spatial redundancy present in most images by (approximately) reconstructing lost pixel information from neighbouring pixels.

The basic idea in the pixel scattering algorithm is to make sure that adjacent pixels in the original image are placed in separate network packets. For example, rather than choosing pixels horizontally or vertically for placement into network packets, pixels are chosen in a diagonal fashion (see Figure 2), and successive packets carry non-adjacent diagonals. In this way, the loss of a single network packet, or a small burst of network packets, has minimal effect on picture quality, since adjacent pixels are rarely lost. The algorithm can also be extended to video applications by treating the temporal axis (i.e., time) as a third dimension in the diagonalization process.

In our implementation of the pixel scattering algorithm, we adapt the algorithm to an ATM-based network environment. The image is divided into equal blocks of 16x16 pixels. The pixels for each packet are selected from the block using two parameters: $Packet\_offset$ and $Byte\_offset$. The packet offset specifies the starting pixel location for the next packet, and the byte offset specifies the next pixel to be picked for the same packet. In our implementation, the packet offset is half the row length, and the byte offset is the row length (16) plus one. The selected 16 pixels are encoded and transmitted in one ATM cell. Each cell stores 48 bytes (i.e., 16 pixels with a resolution of 24 bits per pixel) of (uncompressed) information. Individual packets could also be compressed in this scheme, though the compression achieved along diagonals is slightly lower than that achievable in the horizontal or

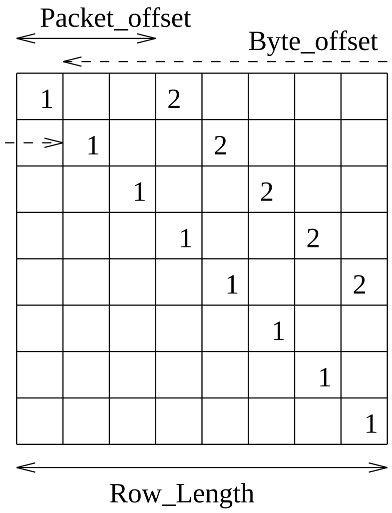

Figure 2: Pixel Scattering Algorithm

vertical dimensions [10].

3.4 MPEG

The MPEG (Motion Picture Experts Group) video compression algorithm is designed to provide high compression ratios while maintaining high video quality [7]. The MPEG standard exploits the advantages of both intraframe and interframe coding methods. The MPEG algorithm is depicted in Figure 3.

There are three types of video frames in MPEG: Intraframe pictures (*I-frames*), Predicted frames (*P-frames*) and interpolated frames (*B-frames*, for Bidirectional prediction). Only the I-frames and the P-frames are actually transmitted over the network; B-frames are reconstructed locally at the receiver. I-frames are encoded (with moderate compression) using an algorithm like JPEG, and provide a reference for encoding and decoding P-frames and B-frames. P-frames are coded with reference to either an I-frame or a P-frame using an interframe method, providing high compression. B-frames are obtained using *motion-compensated interpolation* or *bidirectional prediction*, providing very high compression ratios. The use of past and future reference frames makes the reconstructed frame a close approximation to the actual frame and minimizes error propagation to future predictions [7]. In order to minimize error propagation effects, the MPEG standard calls for a new I-frame after a fixed number of B and P frames (12 in the example in Figure 3).

The MPEG standard is intended for hardware implementation. In our experiments, we use a software implementation of an MPEG decoder obtained from [9]. The decoder takes a compressed video stream, uncompresses it and displays the video sequence on the X window system using either 8-bit, 24-bit, or 32-bit colour depth.

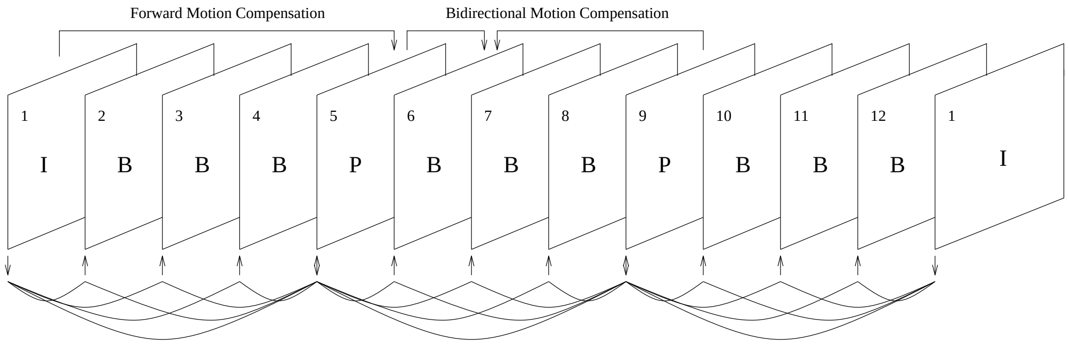

Figure 3: MPEG Frames Combination

4 Description of Experiments

This section describes the experiments conducted to evaluate the video coding techniques. The algorithms are implemented in software on a SparcStation IPC and experimentally evaluated on a test suite of still images. Simulation is used to assess the performance effects for packet video transmission in an ATM-based network environment.

4.1 Evaluation Criteria

Four basic criteria are used to evaluate the performance of the video coding algorithms: compression ratio, picture quality, processing requirements and robustness.

- **Compression ratio**: The compression ratio is the ratio of the original image size (in bytes) to the size of the compressed image. The higher the compression ratio, the better.

- **Processing Requirements**: A certain amount of processing is necessary in order to encode and decode images. The computational complexity of the coding algorithm should be minimal in order to encode and decode the image in real time.

  The processing requirements vary depending upon the picture quality required and the coding scheme used. Some coding algorithms are *symmetric*; the decoding process is a mirror process of the encoding process, and the time required to decode the image is the same as that required to encode the image. Some coding algorithms are *asymmetric*; the decoding process and coding process are different, resulting in different decoding and encoding times. We measure the encoding and decoding times (in seconds) of each algorithm on fixed size images, using the Unix *times()* function. This function measures the CPU seconds (to the nearest 1/60th of a second) spent executing user-space instructions for the encoding (and decoding) process.

- **Picture Quality**: Good compression often comes at the expense of picture quality. Two types of measures for image quality are *subjective* and *objective* measures. The subjective method depends upon the perception of the human observer (i.e., how the human observer assesses the quality of the picture). Objective methods quantitatively measure the error between the original image and the reconstructed image after decoding. The most commonly used objective measures are *absolute error* and *mean square error*. We use both of these objective measures in our experiments.

  The mean square error (MSE) performance metric measures the mean squared deviation of reconstructed pixels from the original image, as follows:

  $$MSE = \frac{\sum_{i,j=1}^{N} (p(i,j) - \hat{p}(i,j))^2}{N^2}$$

  where $p(i,j)$ is the pixel value at position (i,j) in the original NxN image, and $\hat{p}(i,j)$ is the pixel value at the same position in the reconstructed image. The MSE value is computed separately for each colour component (red, green, and blue) in an RGB image, and then summed to get the total MSE.

  The absolute correctness metric simply expresses the percentage of pixels in the reconstructed image that exactly match the original image. For a lossless compression algorithm, this measure would be 100%. For lossy algorithms, the absolute correctness is lower.

- **Robustness**: Coding algorithms should be able to tolerate and recover from packet loss in the network. In many coding schemes, lost packets prevent the reconstruction of the image. In others, image reconstruction can proceed with incomplete information, though image quality suffers.

  Robustness is measured by simulating image transfers in an ATM-based network model, with various levels of (simulated) cell loss. The quality of the image after decoding with incomplete information is then assessed.

4.2 Experimental Method

Experiments were conducted on a test suite of still images for the first three algorithms (RBC, JPEG, Scatter), and on three short video sequences for the MPEG algorithm. Three different types of images were used: a portrait corresponding to the type of image seen in teleconferencing applications, a nature scene with lots of colour and fine detail, and a hot-air balloon image with high-contrast and distinct colours. All images were 704x704 pixels (slightly larger than the NTSC standard) with 24 bits of RGB colour information per pixel.

For each image, our experiments were conducted as follows. First, the image was encoded using each coding algorithm. The compression ratio and the processing requirements were determined while varying the specific parameters for each algorithm. Second, the packetized video transmission of the coded image was simulated using our simulation model with various cell loss probabilities. Finally, the image was decoded after the simulated network transmission, and the decoded image was used to assess the picture quality and robustness of the coding algorithm.

5 Results

This section focuses on the performance results from the experiments that were conducted on the four different video coding algorithms.

5.1 Recursive Block Coding

Figure 4 summarizes the results of our performance experiments for the recursive block coding (RBC) algorithm [4]. The top row shows results for compression ratio and processing requirements, the middle row shows results for picture quality, and the bottom row shows results for robustness. All graphs for the RBC algorithm are shown as a function of T, a threshold parameter that determines whether a block is considered uniform (enough) or needs to be further subdivided.

The RBC algorithm offers low to moderate levels of compression, depending on the value of T used. When T is small, many 2x2 blocks (the minimum block size) appear in the coded image, providing excellent picture quality (high resolution), but low compression ratios (2:1). When T is large ($T = 2048$), larger blocks result, and the compression ratio improves (8:1 to 16:1), but at the expense of picture quality (lower resolution). The compression ratio achieved also varies slightly for different image types.

The processing requirements for the RBC algorithm are fairly significant. Encoding requires 15-30 seconds per image, depending on the threshold parameter T. The larger T is, the fewer blocks there are to encode, transmit, and decode, and the lower the processing requirements. Decoding time is significantly less than encoding time, as was indicated in Section 3.1. In either case, however, the RBC algorithm is not suitable for real-time video coding.

Our objective measures of image quality are presented in the middle row of Figure 4. One performance metric (MSE) measures the mean squared deviation of reconstructed pixels from the original image; the other (Absolute Correctness) measures the number of pixels in the reconstructed image that exactly match the original image. Both measures indicate that image quality decreases with increasing T. In general, MSE values less than 200 represent good image quality. A subjective assessment of image quality would suggest that a threshold value of $256 \leq T \leq 1024$ is a reasonable tradeoff between compression ratio and picture quality.

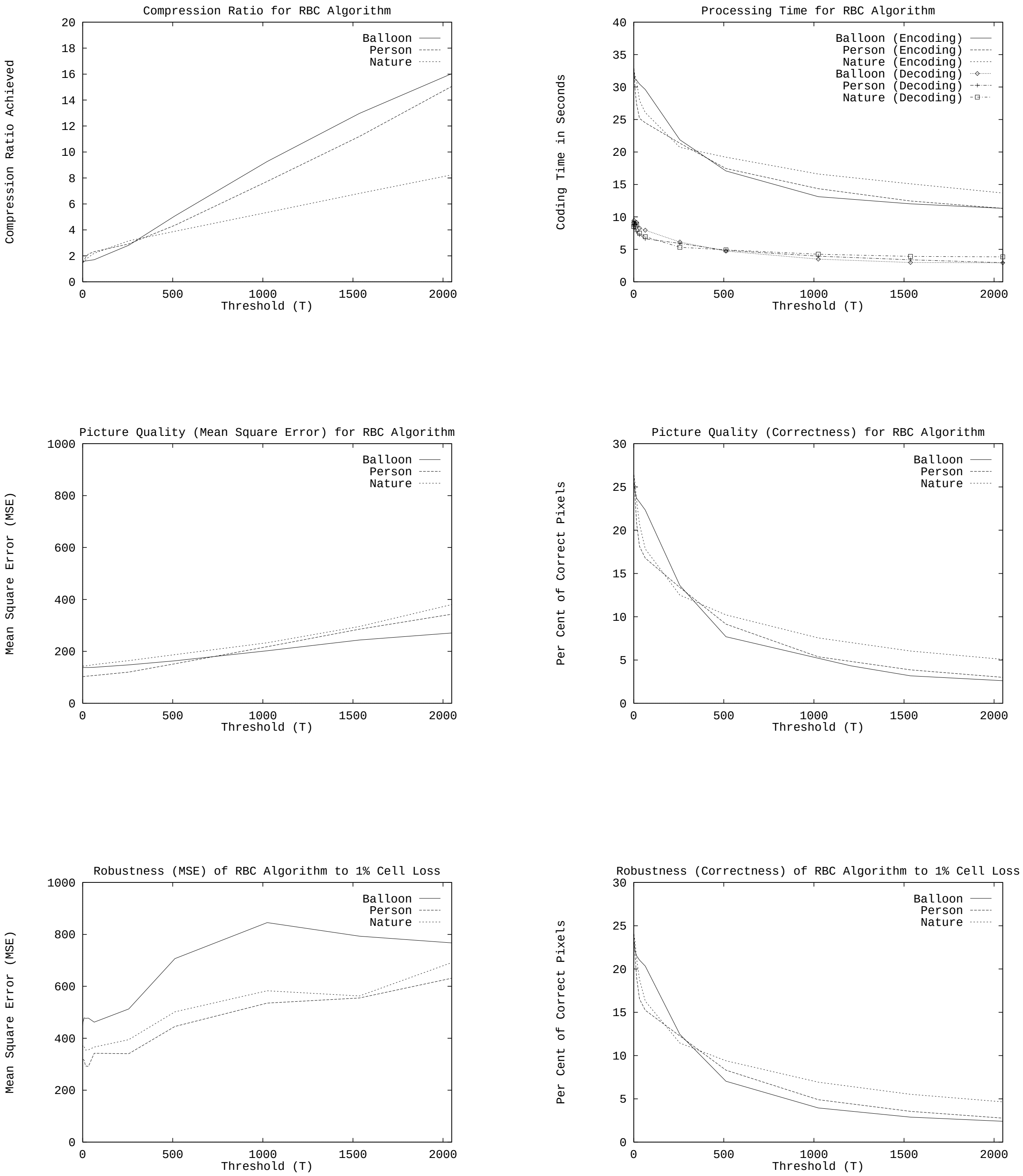

Figure 4: Experimental Results for the RBC Algorithm

One redeeming feature of the RBC algorithm is its ability to tolerate cell loss. Since each block can be encoded and transmitted separately, the loss of one block has no effect on other blocks. The bottom row of graphs in Figure 4 shows the effect of a (simulated) 1% level of network cell loss on image transfer. Because of the cell loss, the error in the reconstructed image increases, but the algorithm still produces reasonable images. To maintain reasonable picture quality with 1% cell loss, the threshold value should be set between $128 \leq T \leq 512$.

5.2 JPEG

Figure 5 summarizes the performance results for the JPEG image compression algorithm. In JPEG, the operation of the algorithm is controlled by a parameter Q specifying the number of quantization levels used. Quantization is a process where the pixel value (consisting of the red, green and blue signal strengths) is compared with a set of discrete levels. Pixel strengths falling between two levels within the quantization band are rounded off to a fixed level. The quantization number Q sets the number of discrete levels used.

The JPEG algorithm offers moderate to high levels of compression, depending on the quantization parameter Q. A low quantization number leads to very few binary levels (codes) that are available to represent pixels, and results in high compression (up to 160:1). Higher quantization levels result in lower compression ratios (20:1), but better picture quality.

Processing requirements for the JPEG algorithm are fairly intensive (about 20 seconds per image), but slightly lower than those for RBC. In JPEG, the coding times are almost identical for each image type, and vary only slightly with Q, the number of quantization levels. Decoding times are slightly lower than encoding times in our experiments, despite the fact that JPEG is supposedly a symmetric algorithm.

Our objective picture quality measures are presented in the second row of Figure 5. Picture quality (according to the MSE measure) is excellent as long as a reasonable number of quantization levels are used (here, approximately 10). A suggested value for Q is $15 \leq Q \leq 50$. Interestingly enough, the absolute correctness measure of picture quality shows that the JPEG algorithm preserves less than 1% of the original pixel values in the reconstructed image. This result is a natural consequence of the quantization process, and should not be interpreted to say that JPEG is a poor algorithm. Rather, this result can be interpreted to mean that our absolute measure of pixel correctness is not a good metric to use (at least in this case). Despite its lossy nature, JPEG offers excellent compression, and excellent image quality.

The JPEG algorithm, however, is not very robust. The loss of even a single cell in the encoded transmission makes the reconstruction of the image impossible (because of the Huffman coding, etc). For this reason, no robustness graphs are shown for this algorithm.

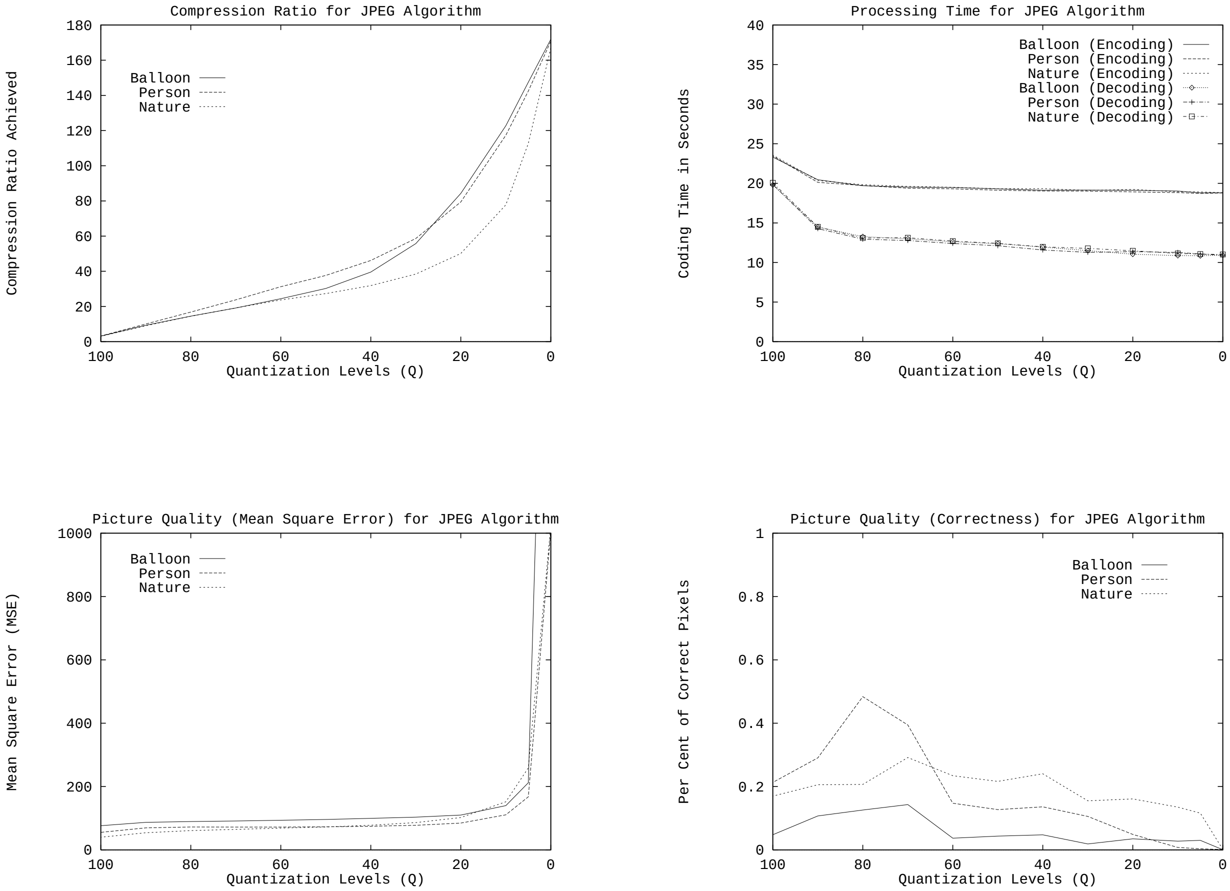

Figure 5: Experimental Results for JPEG

5.3 Pixel Scattering

Figure 6 summarizes the performance results for the pixel scattering algorithm [10]. In our simple implementation of this algorithm, there is no compression at all: all that changes is the order of mapping pixels into network packets (cells). However, the strength of the algorithm is in its robustness. For this reason, all performance results for this algorithm are plotted with respect to the network cell loss probability p.

The compression ratio and the processing requirements of the algorithm (the top row of Figure 6) are clearly independent of the cell loss probability. Encoding time and decoding time depend only on the image size, and not on how many cells are lost. However, the time spent reconstructing the image does depend on the cell loss probability. The second row in Figure 6 presents a measure of the time spent in image correction (reconstructing lost pixels from neighbouring pixels) at the receiver. This image correction time increases linearly with p.

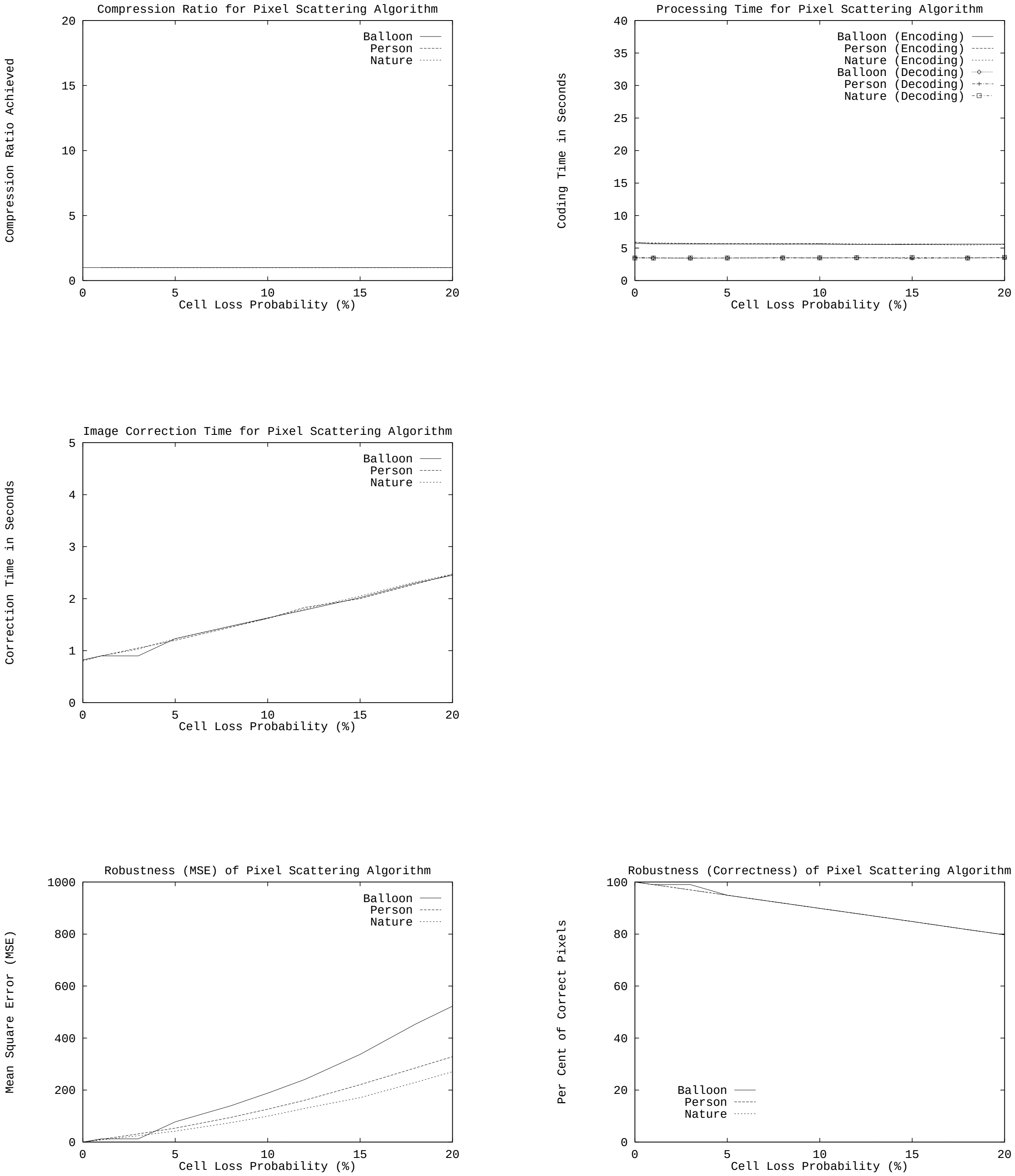

Figure 6: Experimental Results for the Pixel Scattering Algorithm

The bottom row of graphs measure the robustness of the algorithm to cell loss. Our two picture quality measures (MSE and Absolute Correctness) are each plotted. The pixel scattering algorithm provides excellent image quality, even when the network cell loss probability is absurdly high (up to 10%). With perfect network transmission (0% cell loss), pixel scattering is a lossless algorithm.

5.4 MPEG

Our final set of experimental results are for a software version of an MPEG decoder [9]. The video sequences used were produced by XING Technology. The XING MPEG video data contains only I-frames [9]; there are no P-frames or B-frames (i.e., frames with motion compensation or bidirectional prediction). We experimented with three short video sequences varying in length from 60 to 700 frames. In each video sequence, the frame size is always 160x120 pixels (one-quarter the NTSC standard size), with a colour depth of 8 bits per pixel. Thus each uncompressed frame contains 19200 bytes.

The results for these experiments are shown in Table 1. Each line of the table shows a brief description of the video sequence used, the average compression per frame in the sequence, the average frame rate at which the sequence was displayed, as well as a qualitative assessment of picture quality and robustness.

Since only I-frames are used in these video sequences, the compression ratio and the picture quality are fairly low. However, the picture quality and compression ratio would be improved significantly if the decoder were to use P-frames and B-frames.

The processing requirements for MPEG decoding make MPEG (almost) feasible for real-time video (i.e., even with software decoding, frame rates as high as 7 frames/sec are achievable). No experimental results are available for the encoding process.

Finally, it is observed that MPEG is not very robust to cell loss. MPEG cannot tolerate cell loss greater than 2%, and in some cases not more than 1%. The picture quality shown in the table is a purely subjective measure, as the original uncompressed movies were not available for objective measurement.

Table 1: MPEG Performance Results

Movie Type	Compression	Frames/sec	Picture Quality	Robustness
Animation (20 frames)	6.44	2.24	Low	1% Cell Loss
Animation (120 frames)	7.92	5.25	Low	2% Cell Loss
Video Clip (760 frames)	9.53	7.18	Low	1% Cell Loss

Table 2: Qualitative Summary of the Results

Algorithm	Compression	Processing	Quality	Robustness
RBC	Medium	High	Low	Medium
Scattered	Low	Low	Good	High
JPEG	High	High	Very Good	Very Low
MPEG	Medium	Low	Low	Low

5.5 Summary

Table 2 presents a summary of our experimental results. RBC provides a compression ratio of up to 16:1 and JPEG up to 150:1. The scattered encoding algorithm does not provide any compression, and MPEG (with I-frames only) offers compression up to 10:1. JPEG and scattered encoding outperform RBC with respect to picture quality. Scattered encoding outperforms RBC and JPEG with regards to processing requirements, but the MPEG decoder is still the most likely candidate for real-time decoding. For robustness, JPEG is highly intolerant to packet loss, and RBC is quite tolerant to packet loss. The scattered encoding algorithm is superior for robustness.

6 Conclusions

Several conclusions can be derived from this work. First, good compression is often achieved at the expense of picture quality and processing requirements. JPEG is a complex algorithm that provides high compression ratios, but with a lossy image. The pixel scattering algorithm has much lower processing complexities, and a lossless image, but provides no compression.

Second, for an ATM-based network a robust method of video coding is highly desirable. In an ATM-based network that statistically multiplexes variable-bit-rate video sources without explicit bandwidth guarantees, the cell loss probability may be non-negligible (e.g., up to 1%). In video applications, the compression ratio can be traded off with robust algorithms that are capable of recovering from cell loss, without the delay of end-to-end retransmission.

Finally, tradeoffs exist between the end user requirements (picture quality and processing requirements) and the requirements of the network provider (compression ratio and robustness). Robust methods with low processing requirements that provide reasonably good picture quality are highly favourable for video applications on an ATM-based network.

Further work is required to explore methods such as scattered encoding for real-time video. Work continues to evaluate MPEG more accurately. The objective methods of picture quality presented in this paper provide only an approximate quality measure. More work is needed to determine an accurate method to assess image and video quality.

References

[1] P. Ang, P. Ruetz and D. Auld, "Video compression makes big gains", *IEEE Spectrum*, October 1991, pp. 16-19.

[2] Y. Chen, K. Sayood and D. Nelson, "A robust coding scheme for packet video", *IEEE Transactions on Communications*, Vol. 40, No. 9, September 1992, pp. 1491-1502.

[3] CCITT, Recommendation H.261, Study Group XV, December 1990.

[4] P. Farrelle, *Recursive Block Coding for Image Data Compression*, Springer-Verlag 1990.

[5] I. Habib and T. Saadawi, "Multimedia traffic characteristics in broadband networks", *IEEE Communications*, Vol. 30, No. 7, July 1992, pp. 48-54.

[6] G. Karlsson and M. Vetterli, "Three dimensional sub-band coding of video", *Proceedings of IEEE ICASSP*, April 1988, pp. 1100-1103.

[7] D. LeGall, "MPEG: A video compression standard for multimedia applications", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 46-58.

[8] S. Minzer, "Broadband ISDN and Asynchronous Transfer Mode (ATM)", *IEEE Communications*, Vol. 27, No. 9, September 1989, pp. 17-24.

[9] L. Rowe, K. Patel and B. Smith. "MPEG video software decoder", Computer Science Division-EECS, University of California at Berkeley, November 1992.

[10] C. Turner and L. Peterson, "Image transfer: An end-to-end design", *Proceedings of ACM SIGCOMM '92*, August 1992, pp. 258-268.

[11] G. Wallace, "The JPEG still picture compression standard", *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 31-44.

[12] W. Zou, "Digital HDTV compression Techniques", *IEEE Transactions on Broadcasting*, Vol. 37, No. 4, December 1991, pp. 131-133.