# DATA COMPRESSION FOR THE MICROGRAVITY EXPERIMENTS

Khalid Sayood[†‡], Wayne A. Whyte, Jr.[*],
Karen S. Anderson[†], Mary Jo Shalkhauser[*], and Anne M. Summers[†]

[†]Department of Electrical Engineering
and
The Center for Communications and Information Sciences
University of Nebraska–Lincoln
Lincoln, NE

[*]Space Electronics Division
NASA Lewis Research Center
Cleveland, OH

## 1.0 BACKGROUND

NASA has undertaken an advanced technology development program in the area of high resolution high-frame-rate video imaging to support microgravity science and applications experiments, with the goal of removing constraints on the amount of high speed, detailed data that can be recorded and transmitted. Numerous microgravity experiments have been proposed for Space Station Freedom and the Shuttle which require a broad range of imaging capabilities. Figure 1 presents survey results of user requirements; the chart shows frame rate versus image resolution requirements for many of the proposed microgravity experiments. NASA will develop a digital video imaging system that will be capable of fulfilling as many of the requirements as is practicable. (Initial survey requirements from several of the experiments far exceed state-of-the-art video imaging capabilities. Reexamination of those requirements is now taking place.)

A representative experiment, sponsored by scientists at NASA Lewis Research Center, is entitled Nucleate Pool Boiling. The experiment involves heating freon locally by means of passing a large current through a thin gold coating on quartz. At some point the freon begins to boil causing vapor bubbles to form, grow and depart from the surface. Information to be derived from the experiment includes bubble shape, bubble growth, collapse, departure, and motion after departure from the surface. To obtain the desired measurement accuracy from the video image data, a minimum resolution of 500X1000 pixels is required at a desired frame rate of 1000 frames per second. These requirements are typical of the resolutions and frame rates needed in many of the proposed microgravity experiments.

The imaging system development will progress in stages starting with a demonstration breadboard system which can be upgraded as technology advances. The system will consist of a high resolution imaging device (camera/solid state sensor), a high speed video interface, and a mass storage device (dynamic RAM/magnetic tape). The Phase 2 imaging device will be a 1024X1024 pixel-addressable solid-state sensor with an 80 Mpixels per second multichannel scan rate, providing monochrome images with 8 bits gray scale resolution. It will be capable of image subframing in order to trade off

† Supported by the NASA Lewis Research Center under grant NAG 3-806.
‡ Supported by the NASA Goddard Space Flight Center under grant NAG 5-916.

field of view for frame rate. (For a 1024X1024 pixel image, the 80 Mpixels/sec scan rate of the sensor would allow a maximum frame rate of 76.29 frames per second, however, a subframed image of 128X128 pixels could achieve a frame rate in excess of 4800 frames per second.) The high speed video interface will provide synchronization and routing of data to the mass storage devices. Both high capacity magnetic tape and high speed 512 Mbyte dynamic RAM will be available for mass storage of image data.

Video data compression is scheduled to be incorporated into the imaging system to enhance its capabilities for data acquisition, storage and transmission. Researchers at NASA LeRC and the University of Nebraska–Lincoln are working together to develop appropriate compression algorithms. The data compression aspects of the high resolution high-frame-rate video technology (HHVT) project will be the focus of this paper.

## 2.0 DATA COMPRESSION IN THE HHVT SYSTEM

The quantity of image data that will be generated by most, if not all, of the proposed microgravity experiments is so large that data compression (image processing) will be a necessity in the imaging system. Image data compression can be used to enhance the capabilities of the HHVT system in at least two areas. First, compression that is achievable between the imaging device and the mass storage unit directly increases the storage capacity. A two-to-one compression factor would double the amount of storable data, thereby doubling the available experiment time. (The high speed 512 Mbyte dynamic RAM can accommodate just 6.4 sec of data at the full scan rate.) The second area of enhancement is with image data transmission to Earth. Here, data compression can be used to reduce the transmission bandwidth and total time required for transmission. (A third area where it may be possible to apply data compression techniques is in the focal plane, however, this area is not currently under study).

The data compression requirements differ depending on where in the system compression is being applied. Compression prior to mass storage must be kept simple for straightforward implementation due to the high data throughput rate. The techniques used must work in real-time and, typically, should be lossless to maintain complete data integrity. (Lossy schemes may be acceptable for some experimental data requirements, however, lossy schemes are generally more complex and hence more difficult to implement for real-time processing. Additionally, the compression techniques to be incorporated at this stage in the system will be hardware based rather than software. It may not be desirable to have multiple algorithms in hardware due to weight constraints, therefore, a single lossless technique which is universally applicable would be preferred.)

Once the data is in the mass storage device, high speed processing becomes less of a requirement on the data compression system. A much broader range of compression techniques becomes available because implementation can be done in software rather than strictly in hardware. Because processing speed is no longer as critical at this stage of the data handling process, different data compression techniques can be applied to particular experiments in order to take advantage of differing end requirements among the various experiments. For example, the fidelity criterion is experiment dependent. Some experiments may require that quantitative data be measured from the video record, as in the measurement of bubble size. Other experiments may only require qualitative observation of experiment progress to enable control of activity. In the latter case, image resolution may not be as critical as near real-time control. The data compression techniques selected will most likely be experiment dependent and as such will be capable of responding to individual experiment requirements.

Feature and data extraction also offer the possibility for significant reductions in data transmission requirements. If sufficient sophistication can be incorporated into the imaging system to extract the required quantitative data prior to downlinking, only the measurement results may need to be transmitted. For example, rather than transmitting the high resolution image of a bubble to determine its size, only the dimensions would need to be transmitted if that information could somehow be extracted from the data. While feature extraction is more commonly associated with image enhancement rather than data compression, many of the same techniques may be applicable to both areas.

The remainder of this paper shall address several of the algorithms which have been studied or are currently under study for application to data compression in the HHVT system.

## 3.0 DATA COMPRESSION SCHEMES

The different algorithms presented in this section are elements in a possible "toolkit" of schemes which may be available to the user. The compression scheme presented in Section 3.1 is a lossless coding scheme which is very amenable to real-time hardware implementation. This scheme is therefore a candidate for implementation between the imaging device and the mass storage unit. The remaining algorithms are lossy algorithms and could be used (depending on user requirements) after mass storage.

Several of the lossy algorithms were developed with different applications in mind, but can be adapted for use in the HHVT system. A common property of all the lossy systems is their edge preserving capability. This capability is especially important for the types of images generated by the microgravity experiments, as size and location information is usually derived from edges.

It should be noted that the algorithms presented in this paper do not constitute all the algorithms to be investigated for inclusion in the toolkit. This program is in its initial stages and the algorithms presented in this paper are simply some of the algorithms that currently look promising.

### 3.1 A DIFFERENTIAL LOSSLESS CODING SCHEME

A high resolution image can be viewed as an image which has been "oversampled". This view leads directly to the inference that there is a high degree of correlation between pixels. The oversampling point of view also automatically discards such pathological cases as images of snow on a TV screen, which can play havoc with any data compression scheme. If we assume a Natural Binary Coding (NBC) or Folded Binary Coding (FBC) scheme, we can also assert that a high degree of pixel to pixel correlation will result in a high probability of the most significant bits of the neighboring bits being identical. A similar argument can be used, with some modification, for other binary coding schemes. The noiseless coding scheme presented in this section takes advantage of this fact to provide compression. It has been motivated by an encoding scheme for tree structured vector quantization [1]. The algorithm functions by comparing the current pixel (byte) value with a reference pixel to obtain a prefix and suffix value for each pixel in the image. The prefix and suffix values comprise the noiseless code for the pixel. In the following we describe the details of both the suffix and the prefix.

The prefix value is the number of MSB (upper bits) in a byte that are identical to the reference pixel. For example:

| | | |
|---|---|---|
| reference pixel (previous byte) | = | 11010110 |
| current pixel (byte being coded) | = | 11011010 |
| prefix value = 4 | | - - - - (1101) |

Before being sent the prefix value is Huffman encoded. A given prefix value is assigned a predetermined Huffman code. A Huffman code is a tree code with varying lengths. Values with higher probabilities of occurrence are given shorter binary codes than values with lower probabilities of occurrence. The prefix value can range from zero to eight. The prefix values zero to eight are assigned Huffman codes generated by that image. Currently, a unique set of Huffman codes (for the prefix values) is being generated for every image. Some examples of Huffman codes are shown in Table 1. Further investigation may be given to using standard sets (same set) of Huffman codes for every image. Initial investigation indicates that more than one set of codes would be needed in order to not decrease the compression ratio. A set for high, medium, and low correlation would most likely be used.

The suffix is the bits of the current pixel that are not identical to the reference pixel minus the most significant bit (MSB) of the nonidentical bits. The MSB (of the nonidentical bits) is not sent because it is obviously the opposite of the reference pixel (otherwise it would be the same as the reference and be included in the prefix value).

The actual data sent for each pixel is the Huffman code for the prefix value and suffix, sent as is (bit for bit). In the previous example, if the Huffman code for 4 is 10 the code sent for the current pixel given would be 10010. Due to the Huffman code (variable length code) and the fact that the suffix length is directly dependent on the value of the prefix, the compressed code sent is a variable length code.

The next problem is to actually transfer the new code. Data is transferred in bytes (eight bits). To get data compression, the codes must be compacted into full bytes. If a byte is used for each code there would be no compression. Therefore, bits are placed into bytes and transferred as soon as a byte is filled.

The decoding is done by reading the bytes bit by bit. The bit(s) are matched against the Huffman codes to determine the prefix value. The Huffman code is currently being sent with the encoded image. If no match is found, another bit is added to the prefix bits and the new set is matched against the Huffman codes. Once a match is found, that many upper bits of the reference pixel are set in the current pixel being decoded. Then the next bit (bit # = 7-prefix value) value is flipped, from that of the reference pixel. Then, according to the prefix value, the suffix bits are set. If the prefix value is four, then the suffix must contain three bits. For example, reversing the first example:

code sent = 1 0 0 1 0
first bit compared = 1 (no match)
add bit, compare = 1 0 (matches prefix = 4)
if, reference pixel = 1 1 0 1 0 1 1 0
set current pixel = 1 1 0 1
flip next bit = 1
set the next three (7-4) bits, suffix = 0 1 0
current pixel = 1 1 0 1 1 0 1 0

The next bit read from the code would be the start of the next prefix value.

The very first pixel of every image is always sent as is and is always the first reference pixel. The first line always sets the reference pixel to be the previous pixel, to the left. For the first pixel on each

line the reference pixel is always the pixel directly above the current pixel. These reference pixels are always true no matter how the rest of the image is referenced. To determine the reference pixel for the rest of the image, three different algorithms have been investigated. The first algorithm, REFLEFT, sets the reference pixel, except for the first pixel on each line, to be the previous pixel, the pixel to the left. The second algorithm, REFUP, sets the reference pixel, except for the first line, to be the pixel directly above the current pixel. The third algorithm, THRESH, combined the first two algorithms. The third algorithm flips the reference pixel between above and to the left depending on the threshold value. The threshold value is set at the beginning of the program. If a prefix value drops below the threshold value, the reference pixel is switched (from above to left or vice versa). For example, if the reference pixel currently being used is to the left and the threshold value is three and the current prefix value is two, then for the next pixel, the reference pixel used will be above.

In Table 1 the compression obtained, using several images, for the three different algorithms is presented. For the third algorithm, THRESH, the data is presented using threshold values of three, four, and five. The results obtained by using these algorithms were compared against the commercially available compression program PKARC. PKARC compresses files by optimizing between Huffman encoding, a static Lempel-Ziv-Welch coding scheme, and a dynamic Lempel-Ziv-Welch coding scheme. Thus PKARC provides a good benchmark against which to test this algorithm. Note that as the current approach consists of a single algorithm, it is much simpler to implement than PKARC. The results are also shown in Table 1.

As one can see, the new algorithms provide consistently better compression. There is also a direct relationship between the validity of the oversampling assumption and the compression obtained. The compression obtained for the 384X512 images is in general substantially higher than the compression obtained for the 256X256 images. Among the 384X512 images the IBMAD image has the lowest compression because of the presence of granular noise in the image. This is evident from the IBMAD picture. The granular noise because of its "white" nature violates the oversampling assumption. The oversampling assumption is also violated in a more direct manner in Images 13 through 15, and therefore there is a corresponding drop in compression. Obviously this scheme will perform best for the application for which it has been developed, namely, high resolution images.

As mentioned previously, all our tests have been conducted on relatively low resolution images. We expect substantial increases in performance when we code high resolution images. Noting that going from a 256X256 image to a 384X512 image approximately doubles the compression efficiency, we expect the same kind of performance improvement when going from 384X512 to 1024X1024 images.

## 3.2 ENHANCED DPCM ALGORITHM

An algorithm has been developed which is based on differential pulse code modulation (DPCM) for simplicity of implementation, but incorporates performance enhancements which result in reconstructed images that are subjectively indistinguishable from the original image at an average rate of 1.8 bits per pixel (bpp). A hardware implementation of the algorithm has been developed and is presently undergoing testing. The algorithm was developed for use with standard NTSC (National Television Systems Committee) video images, and will therefore need to be modified for application to the HHVT imaging system. However, the required modifications should not be major, nor should they affect the performance of the algorithm. In addition to the DPCM, the algorithm incorporates a non-adaptive predictor value, non-uniform quantization and multilevel Huffman coding to significantly improve upon the performance achievable using a standard DPCM approach.

A two-dimensional pixel average is used to generate the predicted value, PV, for determining difference values in the DPCM process, as shown in the block diagram in Figure 2. For the NTSC signal, sampling is done at four times the color subcarrier frequency (14.32 MHz). Neighboring pixels having the same subcarrier phasing relationship as the current pixel are used for the prediction. The difference value, DIF, is calculated by subtracting both the predicted value, PV, and a non-adaptive predictor value, NAP, from the current pixel value, PIX, ($DIF = PIX - PV - NAP$). The function of the NAP is to improve the prediction of the current pixel. The non-adaptive predictor estimates the difference value that would be obtained if just the predicted value were subtracted from the current pixel value ($PIX - PV$). The subtraction of the NAP value from $PIX - PV$ causes the resulting difference value (DIF) to be close to zero. The smaller the DIF, the more efficiently the quantized pixel information can be transmitted due to the use of Huffman coding prior to transmission over the channel. (Huffman coding assigns variable length codewords based upon probability of occurrence.) To reconstruct the pixel, the decoder uses a lookup table to add back in the appropriate NAP value based upon knowledge of the quantization level from the previously decoded pixel.

The development of the non-adaptive predictor was predicted on the likelihood that the difference values of adjacent pixels are similar. The prestored NAP values were generated from statistics of numerous television images covering a wide range of picture content. The NAP values represent the average difference values ($PIX - PV$) calculated within the boundaries of the difference value ranges of each quantization level for the sample images. The use of the NAP results in faster convergence at transition points in the image, thereby improving edge detection performance. The rapid convergence also reduces the total data requirements by increasing the percentage of pixels in the middle quantization levels, where the shortest length codewords are assigned by the Huffman coding process.

The quantizer shown in Figure 2 has thirteen (13) levels. Each level has a quantization value associated with a non-uniform range of difference values. The quantizer provides more levels for small magnitude differences which would result from subtle changes in picture content. The human eye is sensitive to small variations in smooth regions of an image and can tolerate larger variations near transition boundaries where large difference values are more likely to occur. The non-adaptive predictor discussed previously, acts to reduce the difference values thus improving image quality by reducing the quantization error. This is because the non-uniform quantizer results in lower quantization error for small magnitude differences than for large magnitude differences.

The final major aspect of the encoding algorithm is the multilevel Huffman coding process. Huffman coding of the quantized data allows shorter codewords to be assigned to quantized pixels having the highest probability of occurrence. A separate set of Huffman codes has been generated for each of the thirteen quantization levels. The matrix of code sets is used to reduce the number of data bits required to transmit a given pixel. The particular Huffman code set used for a given quantized pixel is determined by the quantization level of the previous pixel. As with the NAP, the Huffman code trees were generated by compiling statistical data from numerous images covering a broad range of picture content. Probability of occurrence data was compiled for each of the thirteen quantization levels as a function of the quantization level of the previous pixel. A separate Huffman code set was then generated based on the probability data of "current" pixels falling into each of the thirteen quantization levels of the "previous" pixels. There is a tendency for neighboring pixels to fall into the same or close to the same quantization level. By recognizing and taking advantage of this fact, the use of the multilevel Huffman code sets provides significant reductions in bits per pixel over a single Huffman code tree because they allow a greater percentage of pixels to be represented by short length codewords.

Due to the predictive nature of DPCM-based schemes, bit-errors on the channel can effect the quality of the prediction of future pixels on a line. This has the subjective effect of producing a visible streak across the reconstructed image from the point of the error to the end of the line. To minimize the propagation of such errors, the algorithm employs line and field resynchronization. In addition, the University of Nebraska has developed an error detection/correction scheme which is directly applicable to this algorithm and offers significant error immunity for minimal data overhead.

## 3.3 EDGE PRESERVING DPCM

Adaptive Differential Pulse Code Modulation (ADPCM) is a very popular compression technique because it is easy to implement, has low processing overhead, and relatively good fidelity. However, ADPCM image compression is far from ideal. The most obvious drawback is poor edge performance. ADPCM cannot track sudden changes in the image statistics, and this causes substantial edge distortion in the reconstructed image. Some changes in the basic approach are required to reduce edge degradation, while retaining simple, high speed image compression.

We have developed a modified ADPCM scheme which uses a very simple algorithm to prevent edge degradation [2]. The structure of the proposed system is based on the embedded DPCM scheme of Goodman and Sundberg [3]. The new system detects edges and sends extra bits containing edge information. We have shown that substantial improvements in both the subjective and objective edge performance can be obtained using this method [2].

Figure 3 shows the general block diagram of a DPCM system. It works much like Delta Modulation. In fact the basic concept is the same; only the information that cannot be predicted at the receiver is sent. $P$ denotes the predictor and $Q$ the quantizer; $s$ is the value of the $k$ input pixel and $p$ is the predicted value. The difference,

$$e = s - p \tag{1}$$

is the prediction error. This value is quantized, and the quantized value $eq$ is sent to the receiver. The quantizer error, $q$, can be viewed as an additive noise process.

$$e_q = e + q \tag{2}$$

The quantized error, $eq$, is fed back to the predictor, added to the current predictor value,

$$\hat{s} = e_q + p \tag{3}$$

and used as input to for the next prediction.

$$p = f(\hat{s}, e_q) \tag{4}$$

The predictor function $f(\hat{s}, e_q)$ is discussed in the following section. A corrupted version of $e_q$ arrives at the receiver.

$$\tilde{e}_q = e_q + c \tag{5}$$

where $c$ is the channel noise. This is added to the receiver's predicted value, and if the predictors at the receiver and transmitter are the same, and the channel noise is negligible $\hat{s}$ and $\tilde{s}$ will be the same. Therefore the reconstructed signal, $\tilde{s}$, and the true signal, $s$, will differ only by the quantizer noise.

$$\tilde{s} = s + q \tag{6}$$

This basic fact has led to many designs that attempt to minimize quantizer noise. Most of them are application specific, and for the most part they are successful, especially when applied to speech signals. However, the results are not as impressive when applied to image data [4] [5]. The best results have been achieved using adaptive quantizers and/or adaptive predictors. Such systems are usually referred to as Adaptive DPCM, or ADPCM [4].

The predictor function, $f(\hat{s}, eq)$, is chosen so as to minimize the variance of eq. There are many well-known adaptive filter algorithms that can be used to adapt the predictor. We have found that the simple Least Mean Square (LMS) gradient search algorithm is an effective algorithm for adapting the predictor. We have previously shown that edge performance is improved if a pole zero or ARMA predictor is used instead of an all pole or AR predictor. Therefore the adaptive predictor used in the ADPCM system is an ARMA predictor. Both the AR and MA coefficients are adapted using an LMS algorithm. Because of the non-stationary nature of image data, optimization of the gain parameter in the LMS algorithm is not possible. The gain should be relatively small to insure stability. The presence of adaptive zeros also makes the system less susceptible to channel noise.

The quantizer is a two-bit Robust Jayant quantizer [6] [7]. It is a uniform quantizer whose stepsize, $(k)$, is adapted based on the previous sample. The stepsize is expanded if the input falls in the outer quantization levels while it is contracted if the input falls in the inner quantization regions. This algorithm is simple to implement and requires very little computational overhead. Since DPCM is most often used in systems where speed is premium, this method is understandably quite popular. It decreases the quantizer noise; however, it doesn't adapt well enough to solve the edge distortion problem. Simulation results in [2] clearly show the poor edge performance of the ADPCM system. A plot of the quantization noise when encoding a simulated edge shows that the magnitude of $q$ is large near the edge and slowly dies away as the system adapts. The error images obtained in this study clearly show that the quantizer distortion is mainly an edge phenomena.

The first step to improving edges is detecting edges. Once this is done, steps can be taken to alleviate the excess noise. Ideally the edge detection would be simultaneously performed at both the transmitter and the receiver thus eliminating the need for transmitting the edge location. Fortunately the Jayant quantizer is well-suited to this task. The Jayant quantizer is designed to track the variance of the quantizer input by changing its stepsize $(k)$. Since edges are regions where the statistics change rapidly, it follows that the stepsize will expand repeatedly when it encounters an edge. This fact is made use of in the following rule to detect edges:

An edge is detected when the stepsize of the Jayant quantizer expands more than $P$ times in succession, $P > 1$. $P$ should be small to reduce the detection delay; a value of two seems to work well. The output of the edge detector is one when edges are present (that is, the Jayant quantizer stepsize expands more than two times in succession) and zero everywhere else. This detector algorithm, with $P = 2$, was added to the ADPCM simulation and tested using a step input. The results showed that both the receiver and the transmitter simultaneously detect the same edges. As such, no extra information is required to synchronize the detectors.

Now that an effective mechanism for detecting edges at both the transmitter and receiver has been

obtained, this information can be used to improve the edge performance of the ADPCM system. The structure used in the current approach is the embedded DPCM structure proposed by Goodman and Sundberg [3]. The embedded DPCM scheme employs an additional or "embedded" quantizer to transmit the quantized quantization error of the DPCM structure to develop a strategy for transmission over noisy channels. In the current approach the embedded quantizer is switched on by the edge detector and remains active for as long as the edge detector declares the edge to be active. During this period the embedded quantizer transmits a quantized version of the ADPCM quantizer error $q$ over a "side channel". This is removed from the ADPCM receiver output $\tilde{s}$. Thus during the period that the edge detector declares an edge to be active the reproduction error is $(q - q)$ instead of $q$. This has the effect of reducing the large quantization error at the edges and preventing edge degradation. As the edge is detected simultaneously at both the transmitter and receiver, the receiver knows when to expect transmission over the side channel and the transmitted quantization error values are synchronized with the reconstructed values at the output of the ADPCM receiver. The issue of exactly how to configure the side channel is not addressed in this work. However, the ability to easily achieve synchronization seems to suggest that configuring the side channel should not be a very difficult task.

The proposed system was simulated using the USC GIRL and USC COUPLE image as the source images. A two-bit robust Jayant quantizer and a pole-zero (ARMA) adaptive predictor of the type described before was used. There was considerable improvement in the edge performance. This was reflected in both objective (SNR) and subjective (perceptual) improvement. The overhead due to the side information was less than half a bit per pixel.

While the use of the Jayant quantizer for edge detection is efficient from the point of view of savings on side information, the current definition of an edge is rather ad hoc. Because of this, the savings in side information during the edge detection process may be offset by the extra side information needed for the edge preserving process. In fact an overhead of around 0.5 bits/pixel for a coding scheme with nominal rate 2 bits per pixel seems rather high. We are currently examining this technique from several points of view. The first is to get a more exact definition of an edge in terms of the Jayant quantizer than the one used in the above study. The second is to examine more conventional edge detection systems including the IDS system proposed by Cornsweet [8] and Huck [9]. These methods would be used to find and extract the edges from the image. The edges could then be coded separately, while the image sans edges could be very efficiently coded using a low rate DPCM system. Finally we are examining the possibility of developing multiquantizer ADPCM schemes where the switching between quantizers with different rates would be performed based on the behavior of the Jayant quantizer.

### 3.4 A MODIFIED RUN-LENGTH CODING SCHEME

The final algorithm presented here is also a variation of the popular DPCM scheme. Again, one of the objectives is to reduce the excessive edge degradation present in standard DPCM systems. Another objective is to have a system that can operate under situations where a common communication channel is being used by a number of users and thus the available channel capacity may vary over the period of a single transmission. Under such situations the system would be able to reduce the rate in return for accepting a certain amount of distortion. However, the edge fidelity which is the primary objective would still be protected.

The system block diagram is essentially similar to the DPCM diagram of Figure 3 with one important modification. The DPCM encoder output forms the input to a modified run-length

encoder. Of course, the inverse operation precedes the DPCM decoder. This system is a variation of the system presented in [10]. The various elements of the system are presented below.

The predictor is a one tap "integer" predictor. The output of the predictor is given by

$$p_n = \lfloor a s_{n-1} \rfloor \tag{7}$$

where $\lfloor . \rfloor$ denotes the "floor" function. The floor function is used so as to force the predictor output to be an integer. This was done to allow the system to be used for lossless encoding.

The quantizer is a uniform quantizer with stepsize $\Delta$ which effectively contains an infinite number of levels. This means that the only type of quantization noise present is granular noise. There will be no overload noise at the output of the quantizer. If $\Delta = 1$, the quantizer becomes an identity mapping. An infinite number of quantization levels would generally imply an infinite rate; something we definitely want to avoid. This is done by the use of the modified run-length encoder.

The modified run-length encoder puts out n bit, fixed length, codewords corresponding to $2^n$ output levels of the quantizer. The lowest output level represented is denoted by the symbol LOW while the maximum valued output level represented is denoted HIGH. Note that

$$HIGH = LOW + (2^n - 1)\Delta \tag{8}$$

If the quantizer puts out a value corresponding to the levels between HIGH and LOW, the corresponding n-bit codeword is transmitted by the modified run-length encoder. If the output value $X$ is greater than or equal to HIGH, then the codeword for HIGH is transmitted and $X$ is replaced by $X$ – HIGH. If the new value of $X$ is less than HIGH then the corresponding codeword is transmitted, or else the codeword for HIGH is transmitted and $X$ is again decremented by HIGH. This procedure is repeated until the value of $X$ falls below HIGH. The modified run-length decoder treats HIGH as a "concatenation symbol". Whenever the codeword corresponding to HIGH is received the decoder begins accumulating the values until a codeword corresponding to a value less than HIGH is received. A similar procedure is used when the quantized value is less than equal to LOW.

The effect of this approach is to raise the instantaneous rate whenever the prediction error is high, which usually occurs at edges. However because there is no overload noise there is none of the edge degradation usually associated with DPCM systems. Also by adaptively changing $\Delta$, the output rate of the coder can be made to match the available channel capacity.

The USC GIRL image was encoded using this scheme. Noiseless coding was achieved at the rate of about 6 bits per pixel. At bit rates above 2.5 bits per pixel there was no perceptual difference between the original and reconstructed images. Below two bits per pixel granular distortion was noticeable in the quasi-constant regions. However there was no noticeable edge degradation.

## 4.0 SUMMARY AND CONCLUSIONS

In this paper we have attempted to present the environment and conditions under which data compression is to be performed for the microgravity experiment. We have also presented some coding techniques that would be useful for coding in this environment. It should be emphasised that we are currently at the beginning of this program and the "toolkit" mentioned is far from complete.

## REFERENCES


1 D.L. Neuhoff and N. Moayeri, "Tree Searched Vector Quantization with Interblock Noiseless Coding," Proc. 1988 CISS, Princeton, NJ, pp. 781–783, 1988.

2 S.M. Schekall and K. Sayood, "An Edge Preserving DPCM Scheme for Image Coding," Proc. 31 Midwest Symp. Circ. Syst., St. Louis, MO, 1988.

3 D.J. Goodman and C.E. Sundberg, "Combined Source and Channel Coding for Variable-Bit-Rate Speech Transmission," Bell Syst. Tech. J., Vol. 62, pp. 2017–2036, Sept. 1983.

4 N.S. Jayant and Peter Noll, Digital Coding of Waveforms, Prentice-Hall, New Jersey, 1984.

5 K. Sayood and S. Schekall, "Adaptive Prediction Algorithms in Differential Encoding of Images," Proc. 29 Midwest Symp. on Circuits and Systems, Lincoln, NE, 1987, pp. 415–418.

6 N.S. Jayant, "Adaptive Quantization with a One-Word Memory," Bell System Tech. J., pp. 1119–1144, Sept. 1973.

7 D.J. Goodman and R.M. Wilkinson, "A Robust Adaptive Quantizer," IEEE Trans. on Communications, pp. 1362–1365, Nov. 1975.

8 T.N. Cornsweet and J.I. Yellot, Jr., "Intensity-dependent Spatial Summation," J. Opt. Soc. Am., pp. 1769–1786, 1975.

9 F.O. Huck, "Local Intensity Adaptive Image Coding," Proc. NASA Science Data Compression Workshop, Snowbird, UT, pp. 301–309, May 1988.

10 K. Sayood and M.C. Rost, "A Robust Compression System for Low Bit Rate Telemetry – Test Results with Lunar Data," Proc. of the Scientific Data Compression Workshop, CP-3025, Snowbird, UT, 1988, pp. 237–250.

TABLE 1

| IMAGE | PKARC | REFLEFT | REFUP | THRESHOLD 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| (384 x 512) | | | | | | |
| IBMAD | 13% | 23.3% | 26.9% | 26.7% | 26.8% | 26.5% |
| DERIN | 33% | 45.2% | 50.6% | 50.8% | 50.9% | 50.9% |
| EWEEK | 41% | 49.3% | 53.6% | 54.9% | 55.1% | 55.2% |
| PATTY | 35% | 46.1% | 46.7% | 47.8% | 48.1% | 48.1% |
| KARANNE | 26% | 39.7% | 48.5% | 47.5% | 47.4% | 47.2% |
| MARILYN | 30% | 41.3% | 38.2% | 39.8% | 40.0% | 40.5% |
| (256 x 256) | | | | | | |
| HAT | 7% | 21.8% | 25.0% | 23.8% | 23.6% | 23.4% |
| IMAGE01 | 24% | 28.3% | 28.1% | | 28.5% | |
| IMAGE02 | 27% | 33.6% | 35.0% | | 36.3% | |
| IMAGE03 | 13% | 21.1% | 23.7% | | 22.5% | |
| IMAGE04 | 31% | 16.0% | 16.8% | | 16.7% | |
| IMAGE05 | 7% | 15.3% | 13.2% | | 14.8% | |
| IMAGE06 | 42% | 42.8% | 43.1% | | 43.8% | |
| IMAGE13 | 7% | 16.3% | 14.6% | | 15.8% | |

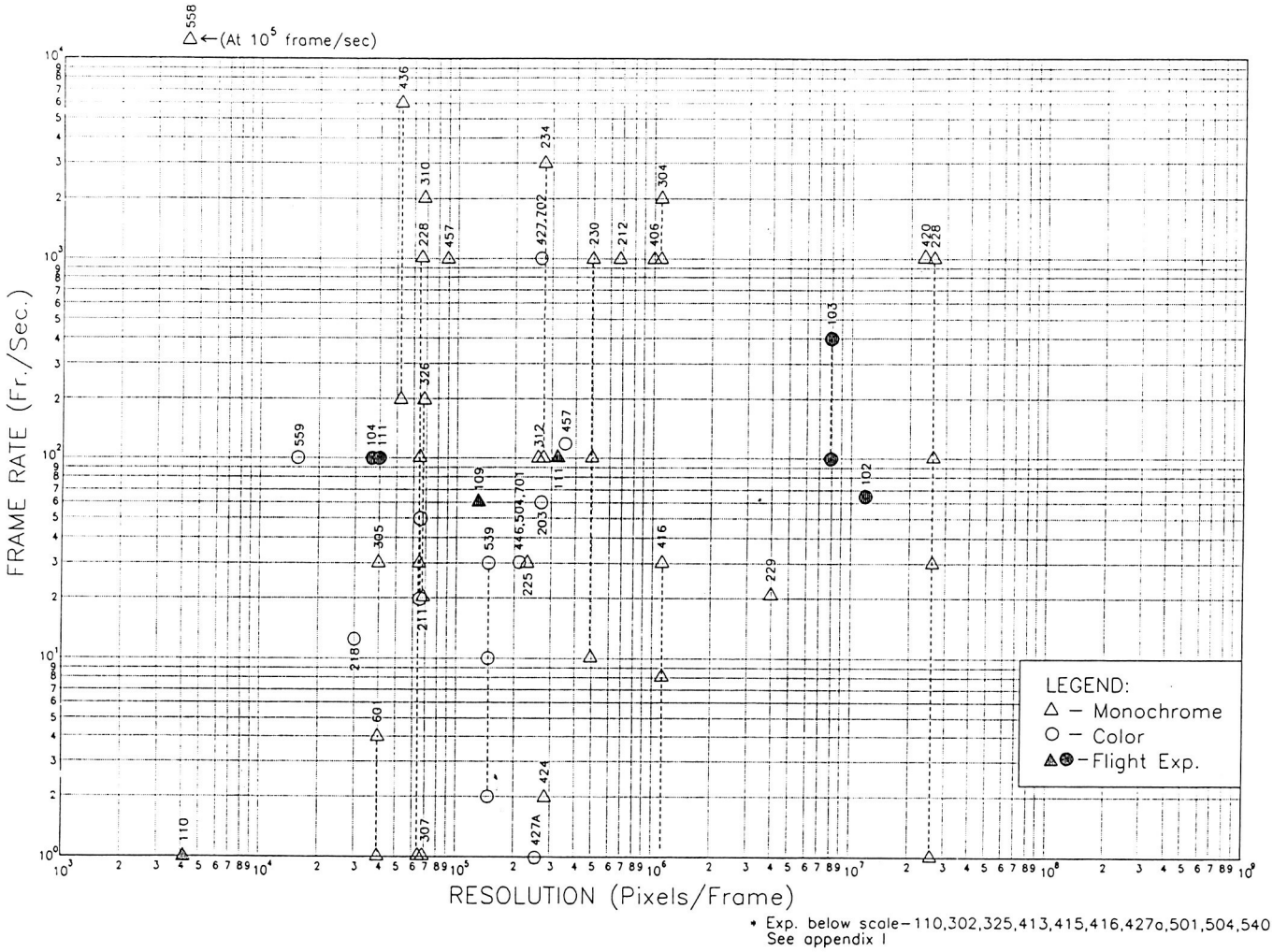



Figure 1. User Requirements Survey Results

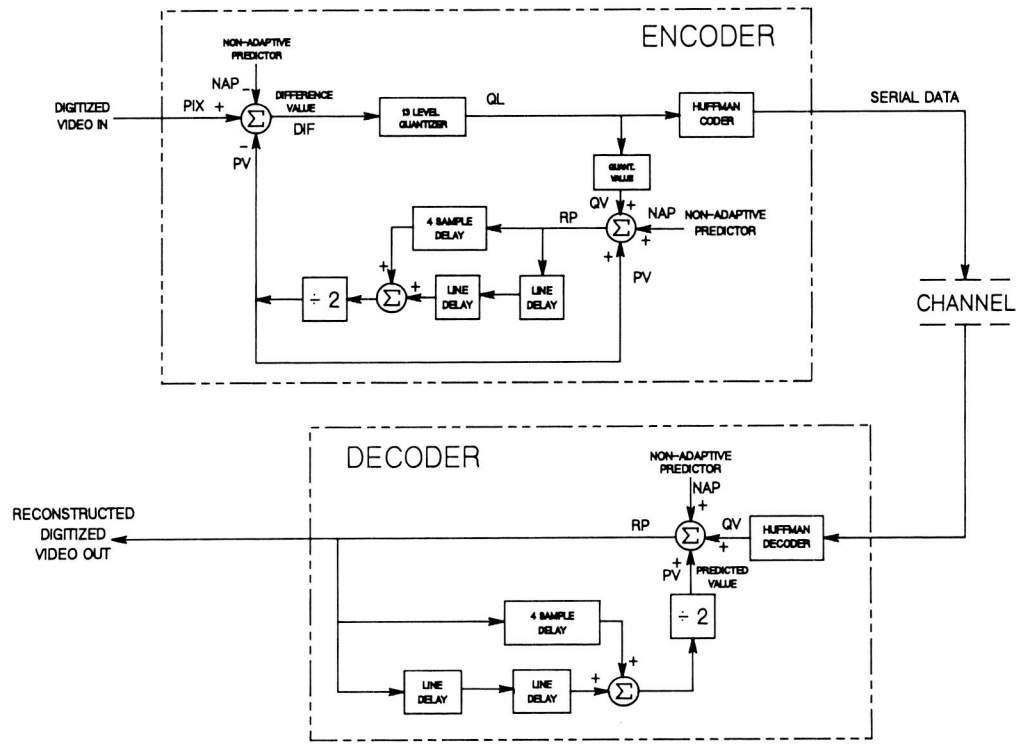
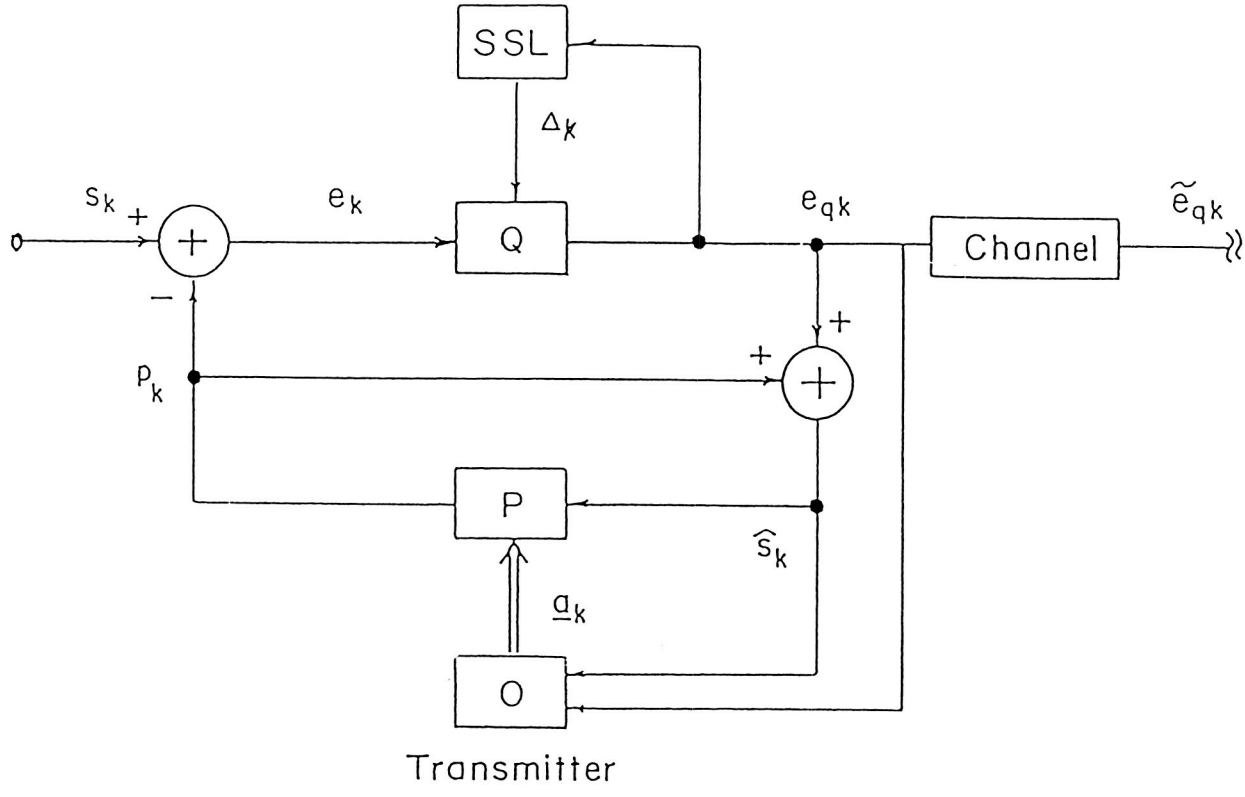



Figure 2. Enhanced DPCM Algorithm Block Diagram



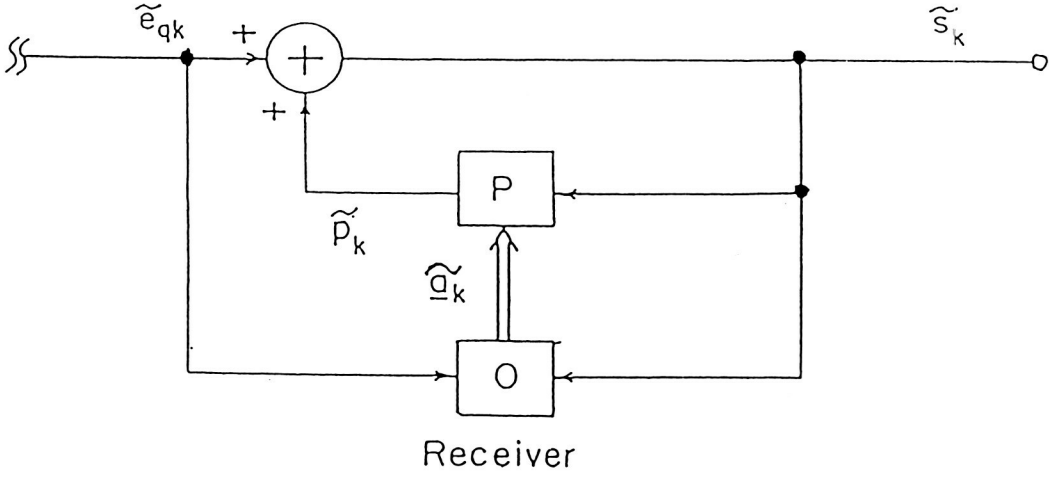




Figure 3. ADPCM Block Diagram