LOW BANDWIDTH VIDEO COMPRESSION WITH VARIABLE DIMENSION VECTOR QUANTIZATION

R B Lambert, R J Fryer, W P Cockshott and D R McGregor

University of Strathclyde
Department of Computer Science
Glasgow G1 1XH
Scotland, UK
E-mail robert@cs.strath.ac.uk

ABSTRACT

This paper presents a new video compression algorithm targeting low bandwidth conversational applications. Based on variable dimension vector quantization, video frames are encoded with variable sized image blocks in a manner which gives precise control over the number of bits assigned to each frame and hence over the transmission delay. This algorithm is compared with an implementation of H.263 for conversational services. The results clearly demonstrate its superiority in minimizing transmission delay while maintaining an image quality comparable to H.263.

1. INTRODUCTION

Advances in communications technology have given us access to ever greater quantities of digital information. However remotely accessing this data increasingly involves long data transfer periods as demand continues to outstrip available bandwidth. While conversational services such as video conferencing and video phone exist, there is in practice insufficient bandwidth available to make their use common-place or fully satisfactory.

Two key factors determine the usability of a video codec for conversational services: the frame capture to frame display time and the perceptual quality of the received sequences. To maintain a natural conversation it is necessary for the participants to see and hear each other's reactions to vocal and visual signals in as short a time as possible. The frame capture to frame display time disrupts this interchange. Note that this interval includes frame encoding, transmission and decoding times. Perceptual quality of the display affects the usability, both aesthetically and in terms of being able to interpret subtle "face and body language".

The objective of this paper is to present a new video codec targeted specifically at low bandwidth conversational services and hence explicitly structured to minimize transmission delay whilst maintaining high perceptual quality.

2. VARIABLE RATE IMAGE COMPRESSION

A typical video sequence, even one limited to a "talking head" will have periods with high detail and/or motion, separated by periods of little motion or detail. In such circumstances it would seem, at first sight, reasonable to focus available bandwidth on subsequences with high detail and/or motion at the expense of the remaining frames.

In such a case the bit budget available to compress a single video frame is determined by the instantaneous frame rate and currently available bandwidth. As both these factors are in principle variable, the compression ratio for each frame must be controllable.

Compression of still images represents a simplification of video compression. The problem of variable compression ratio compression is therefore best illustrated by consideration of this case.

2.1. Variable Dimension Vector Quantization

Vector Quantization (VQ) has been found to be an efficient technique for data compression [1, 2]. Following Shannon's rate-distortion theory, a better compression is achievable by coding vectors rather than scalars. VQ represents a mapping from a k-dimensional space R^k to a finite subset Y of R^k. This finite set Y is called a *codebook*. To code a data sequence, an encoder assigns to each data vector $\mathbf{x} \in R^k$ an index corresponding to a vector in Y, that in turn is mapped to a codeword $\mathbf{y} \in Y$ by a decoder.

In its simplest implementation for image compression, VQ requires that an image be divided into fixed sized non-overlapping sub-image blocks, each represented by a data vector $\mathbf{x} \in R^k$. The pixels in the sub-image block correspond to the elements in the data vector $\mathbf{x}$. Each image data vector is compared with the entries of a suitable codebook Y and the index i of the codebook entry $\mathbf{y}_i$ most similar to the source data vector is transmitted to a receiver. At the receiver, the index accesses the corresponding entry from an identical codebook, allowing reconstruction of an approximation to the original image. This process is illustrated in figure 1. Compression is obtained if fewer bits are required to transmit the codebook vector index than would be required to transmit the raw pixel data.

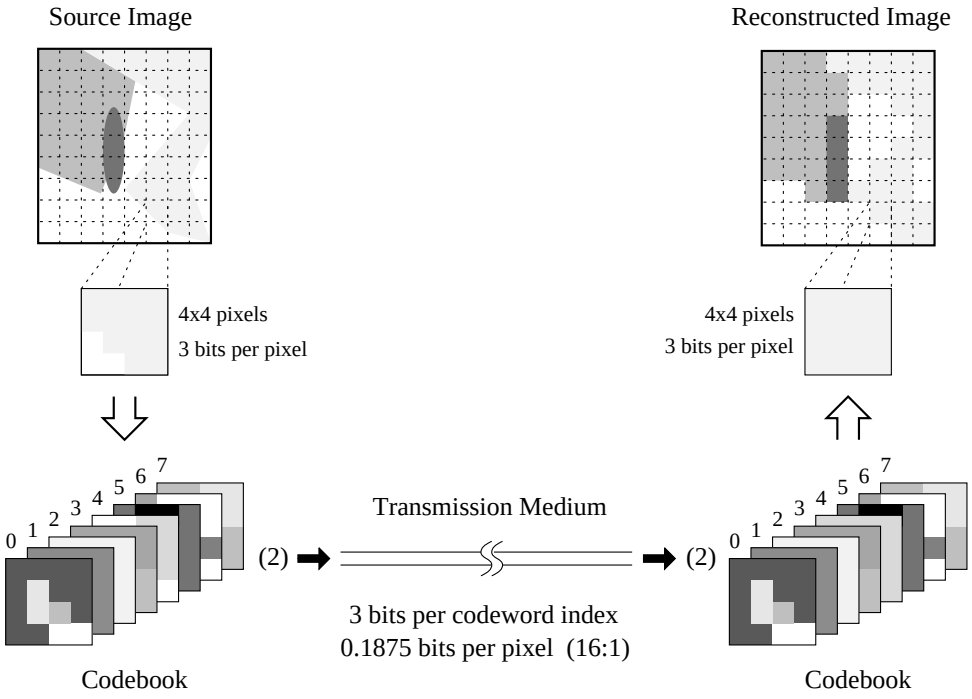

Figure 1: Overview of compression by vector quantization. In this illustration the source image is divided into 4×4 sub-image blocks where each pixel has 8 grey levels giving 48 bits per block. Each block is matched to one of 8 codebook images requiring 3 bits to encode the codebook index giving a compression ratio of 16:1.

While simple, this implementation can only compress at a single compression ratio determined by the size of the block used to partition the image and the number of entries in the codebook. Changing either the block size or the codebook size will allow the compression ratio to vary, but involves the training and storage of many codebooks to take account of all block and codebook sizes used if the compression ratio is to be a dynamic variable.

An alternative is to use several distinct block sizes to encode an image with a codebook assigned to each block size. Variable dimension vector quantization operates on the principle that large blocks should be used to encode areas of low detail, while small blocks should encode areas of high detail [3]. It represents a technique whereby an image can be compressed to a pre-determined compression ratio by defining, for instance, the number of blocks overall that may be used in the encoding of the image.

2.2. Image Segmentation

The established implementation of image compression by variable dimension VQ is to divide an image using a quad-tree format [4] as shown in figure 2. This process, hereafter termed *total block replacement*, divides an image into a grid of $p \times p$ sub-image blocks. Each of these blocks may be replaced by four $\frac{p}{2} \times \frac{p}{2}$ blocks, which in turn may each be replaced by four $\frac{p}{4} \times \frac{p}{4}$ blocks, etc. according to image detail and the required compression ratio. The number of blocks used to encode the image is determined by the compression ratio which defines the total number of bits available (the bit budget).

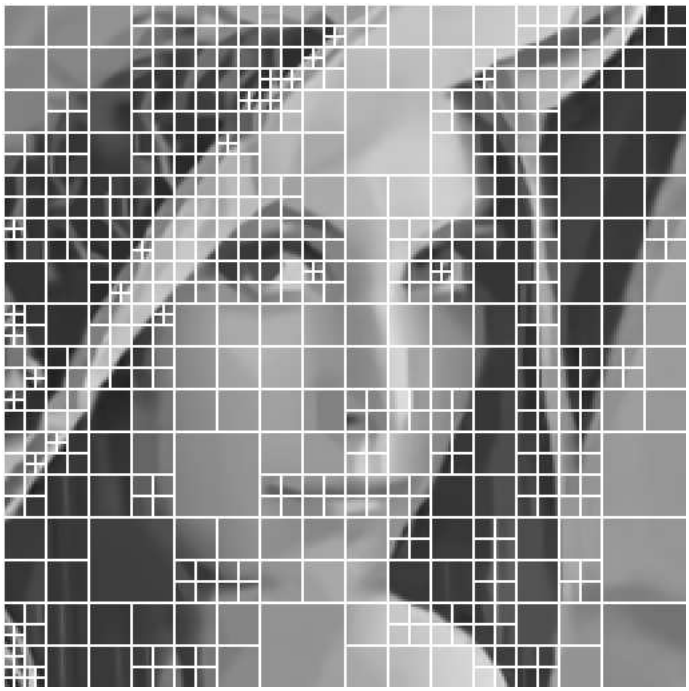

Figure 2: Image divided using a quad-tree structure with block size based on image detail. This image is an actual compression at 0.2 bits per pixel utilizing total block replacement and illustrates the actual encoded blocks used to represent the original 8-bit monochrome image.

While total block replacement does take account of the detail in a source image and can deliver a precise compression ratio, the encoding can include blocks that give little or no improvement [5]. For example, a single block may be replaced by four quarter sized blocks in order to better represent detail contained in just one quadrant of the original block. In this case only one of the quarter sized blocks was required to improve compression quality, the other three being redundant.

Consider an alternative sub-division of this illustrated image, shown in figure 3. Termed *block occlusion*, any size of block at any valid position may be used in the encoding of an image, provided it improves the match between the original and compressed representation. Block occlusion typically gives a better compression than total block replacement [5] as it targets blocks according to where they will do the most good.

The process of block occlusion operates by generating a number of source image block approximations, hereafter termed patches. The compression ratio is defined by the number of patches used in the encoding of an image. The process starts by generating a patch for each of the largest blocks and adding these patches to the encoding of the image. This provides an encoding of the picture at the highest compression ratio possible.

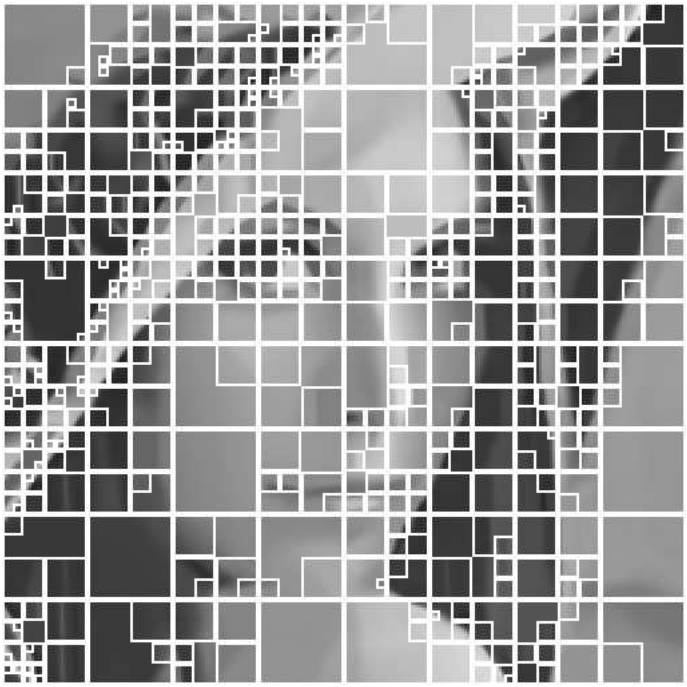

Figure 3: Image divided using block occlusion with block size based on image detail. This image is an actual compression at 0.2 bits per pixel utilizing block occlusion and illustrates the actual encodings of the blocks used to represent the original 8-bit monochrome image.

While the encoding of the image is greater than the specified compression ratio, the block (of any valid size at any valid position) that has not been approximated by a patch and gives the greatest error between the current compressed representation and source image (typically the block with highest squared error) is approximated and the patch added to the encoding. Any patch that is now occluded by the addition is removed from the encoding if it no longer improves quality (e.g. total occlusion).

2.3. The Best First Rule

Vector quantization rarely matches a codebook vector to a data vector without error. Rather the codebook vector represents an approximation to the data vector. The difference between a set of codebook vectors and the data vectors they represent determines the quality of the compression.

For image compression utilizing sub-image blocks of variable size, many possible encodings of an image, for a given compression ratio, are possible. Each of these encodings gives a different figure for compression quality. The traditional way to select a good encoding of an image is to target those areas of the image to be compressed that have the greatest error between the current compressed representation and the original image.

However this approach does not take account of how well the codebook vectors chosen to approximate an image block actually match the original image. The *best first* rule, rather than targeting the image blocks most in error, targets the image blocks that will give the biggest improvement, where improvement for a block is defined as the difference between the compressed representation and original image prior to and following the approximation of the block. For best first block occlusion, the block encoded is the block, regardless of size, that when encoded and added to the compressed representation gives the greatest improvement between the compressed representation and the original image.

3. VIDEO COMPRESSION

The block occlusion and best first processes together provide a means of obtaining variable rate compression over a wide range of compression ratios. With these processes, the compression of video

is a simple extension of still image compression.

3.1. The Strathclyde Compression Transform

Our video codec, hereafter known as the Strathclyde Compression Transform (SCT), assumes that each frame may be encoded by block approximation. These blocks may be of any size and at any position, though in practice a limited number of distinct sizes are permitted at a limited number of positions to simplify block addressing and codebook construction.

The SCT assumes that the last encoded frame represents the starting state (initial compressed representation) for the compression of a new frame. While the encoding for the frame is within a defined bit-budget, the block that does not have a patch in the frame encoding that if approximated and added to the encoding gives the greatest reduction in error between the current compressed representation and source frame, is approximated with a patch and the patch added to the frame encoding. Note the addition of a patch to the compressed representation results in replacement of data over its own area within the encoding of the frame.

The SCT allows for a block to be approximated by many different encoding schemes. However for simplicity the current implementation assumes that a block may be approximated by a VQ codebook, or it may be taken from an encoding of an earlier frame.

3.2. VQ Codebook

Ideally a single constant size universal codebook, available to both compressor and decompressor, should be used. To maintain quality the size of a codebook must be increased as the block size is increased. This limits the practicality of codebook for large block sizes given the memory required for storage of the codewords and the time required to identify the optimal match.

Memory requirements and search time can be traded for compression complexity and compression quality by adopting a coding scheme that takes advantage of correlations in the image data. For example, the size of the codebook can be reduced by setting the mean intensity of each coding block to zero before it is vector-quantized [6]. A further reduction in codebook size can be achieved by normalizing block variance (or contrast) in addition to standardizing the mean intensity [7].

The SCT codec described in this paper uses four distinct codebooks, one for each block size. The codebooks were created using a proprietary self-organizing neural network [8, 9] trained on a set of images of natural and man made scenes. Note that none of the test images and video sequences used to evaluate the SCT were included in this training set. The sizes are 256 of size 32×32, 256 of size 16×16, 256 of size 8×8 and 64 of size 4×4.

To keep the size, search time and memory required by the codebooks low, the codebook vectors were normalized with respect to contrast and mean intensity. A block approximation by a codebook is thus a triple consisting of the codebook index, a contrast adjustment and a brightness offset. The quality of the compression is determined by the quantization of the contrast and mean intensity values, the size and content of the codebook and the overhead of encoding block position.

3.3. Motion Compensation

For most video sequences there is a high correlation of information between frames. To take advantage of this correlation and improve quality most video codecs include motion compensation. Typically motion between frames is calculated and encoded as part of the frame encoding such that a frame is built from detail in the last frame translated in position according to the motion followed by an encoding of the difference between the motion compensated representation and the original frame.

To allow control of bit allocation, the SCT does not separate the encoding of motion and new detail. Rather it treats the last encoded frame or frames as extensions to the VQ codebook. If a block is better approximated by copying a region of equivalent area from a past frame rather than the standard VQ codebook, it will be added to the compressed representation as a block encoded by block translation from a previous frame. In practice only the preceding frame is used with a finite search area of +/- 15 pixels. Note the current implementation of the SCT uses single pixel translation.

4. EVALUATION AND RESULTS

The proposed H.263 standard[1] is generally regarded as the state of the art for low bit rate video compression. To demonstrate performance in terms of delay and quality, the SCT has been compared with an implementation of H.263 developed by Telenor R&D[2] employing the TMN5 rate control for conversational services. Comparisons are made against H.263 without PB frame encoding as frame look-ahead introduces an additional delay. The Unrestricted Motion Vector, Advanced Prediction and Syntax-based Arithmetic Coding modes are all used.

To illustrate delay and quality, a 50 second sequence showing city life in Glasgow was chosen. This included several scene changes, panning shots and zooms. The source sequence is of size 192×192 pixels, 24 bits/pixel. The specified compression rate is 28.8k bits per second at 10.0 frames per second.

The graph in figure 4 illustrates the delay incurred by frame transmission, as defined by the channel bandwidth and bits allocated per frame, for the SCT and H.263 compression of the Glasgow sequence. The delay for the SCT is approximately constant at 0.1 seconds per frame as determined by the bit allocation per frame. However the delays incurred by H.263 are far from constant with the bit usage per frame varying according to the degree of change between consecutive frames.

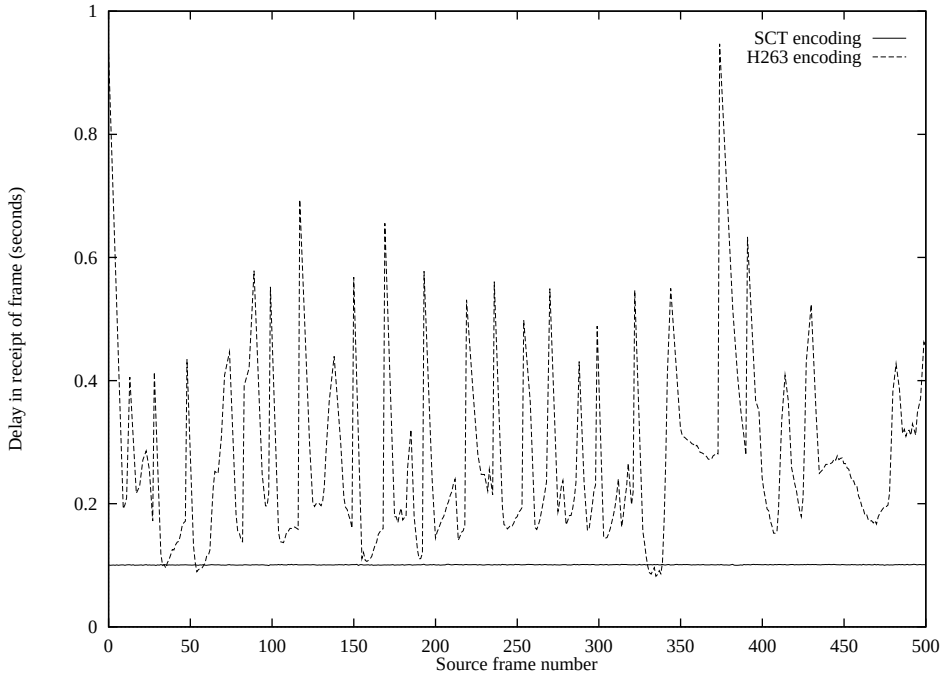

Figure 4: Graph of delay incurred by transmission against frame number for the SCT, and H.263 codecs compressing the Glasgow sequence at 28.8k bits/second.

As can be seen from figure 4, the consequence of applying a variable compression ratio process to video is to incur long delays in frame update during scene changes and periods of rapid or complex motion. Following the encoding of such frames, it is necessary for the H.263 encoder to skip frames in order to keep to the the constraints of the channel bandwidth. For the Glasgow sequence at 28.8k bits/second, only 353 from the initial 500 are compressed by the H.263 encoder.

The graph shown in figure 5 illustrates the quality of the the compression processes in terms of the PSNR between the source and reconstructed frames, where PSNR is defined as;

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{MSE}} \tag{1}$$

For most of the sequence the traces for the SCT and H.263 follow each other closely. However at a scene change or during rapid motion H.263 allocates a higher number of bits to maintain quality at

[1] Documentation for the H.263 standard is available from ftp://ftp.std.com/vendors/PictureTel/h324

[2] Version 1.7 of the Telenor H.263 codec from http://www.fou.telenor.no/brukere/DVC/

the expense of delay and frame rate. The SCT by keeping to a fixed frame rate with a fixed bit budget per frame cannot achieve the high quality initial frames produced by H.263, giving large differences in PSNR. Note however that it is not necessary to keep the frame rate constant. If so desired, the SCT can also skip frames to give a higher quality initial frame, though the the impact on perceptual quality by the delay incurred should be taken into consideration.

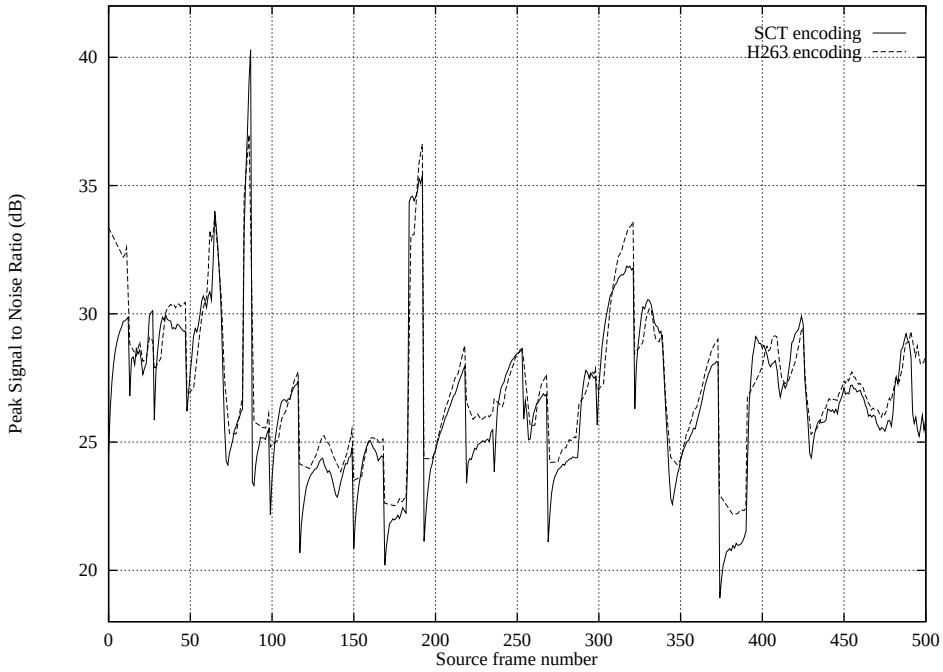

Figure 5: Graph of PSNR between the source and reconstructed frames for the SCT and H.263 codecs compressing the Glasgow sequence at 28.8k bits/second.

The relationship between frame delay, PSNR and the nature of the frames compressed can best be illustrated by a temporal trace of the difference in delay against the difference in PSNR for the Glasgow sequence. Shown in figure 6, the trace represents the change in delay and PSNR differences frame by frame. Note that the greater the delay figure (y-axis), the greater the bit allocation for that frame by H.263.

The H.263 codec tries to maintain image quality at the expense of frame rate and transmission delay. Figure 6 illustrates the consequence of this in comparison with the SCT. The H.263 codec assigns sufficient bits to scene changes and complex motion to maintain quality. This corresponds to the sudden changes in delay and PSNR difference between frames. Following a scene change the SCT catches up and in some cases exceeds the quality of H.263, while maintaining a constant and significantly lower transmission delay.

5. DISCUSSION AND CONCLUSIONS

The usability of conversational services such as video phone or video conferencing is determined by the delay incurred in the transmission of each frame, as defined by the bandwidth and bits used to encode the frame, and by the perceptual quality of the reconstructed frames. In this paper we have presented a new video codec specifically designed for low bandwidth conversational services. Delivering a constant frame rate and transmission delay, the SCT is able to compress to a quality comparable to the H.263 codec.

The SCT codec described in this paper is an early prototype. Several potential quality enhancements are being explored including improved codebook design with codebook sizes optimized for low bandwidth compression, 1/2 and possible 1/3 pel search and translation for motion compensation and a more efficient encoding of the patches. Any or all of these improvements should raise the quality to

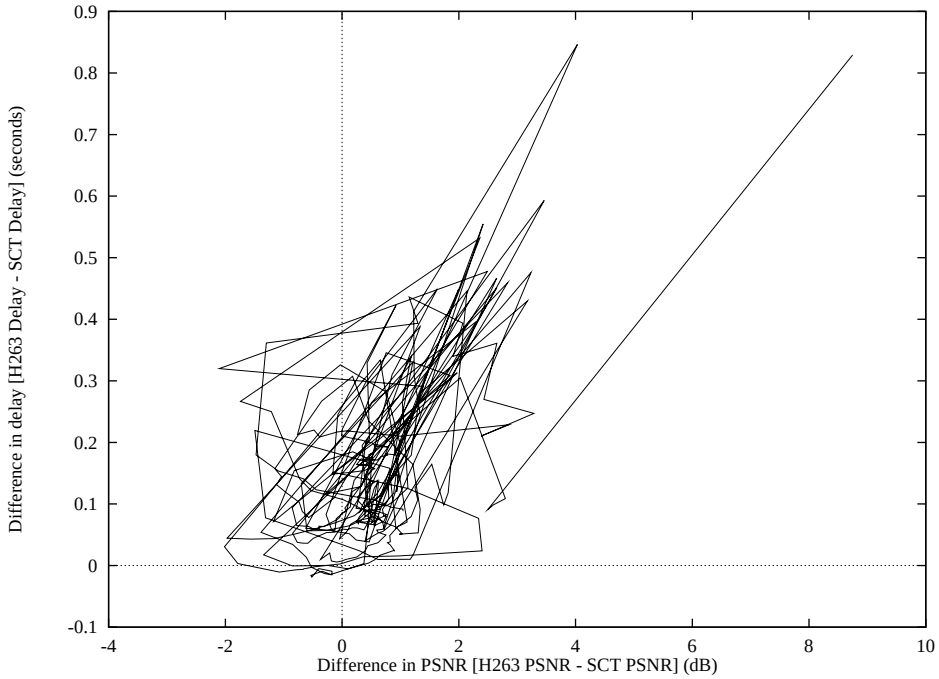

Figure 6: Trace of the difference in delay against the difference in PSNR for the SCT and H.263 codecs as the sequence evolves. Plot is for the Glasgow sequence at 28.8k bits/second. The sequence commences at the top right with a high quality, high bit budget H.263 initial frame.

a level equal or higher than H.263, while providing significantly lower frame capture to frame display delay.

6. REFERENCES

[1] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[2] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36:957–971, 1988.

[3] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1992.

[4] H. Samet. The quad tree and related hierarchical data structures. *ACM Computer Surveys*, 16:188–260, June 1984.

[5] R B Lambert, R J Fryer, W P Cockshott, and D R McGregor. A comparison of variable dimension vector quantization techniques for image compression. In *European Conference on Multimedia Applications, Services and Techniques*, pages 655–670, Louvain-la-Neuve, May 1996.

[6] R. L. Baker and R. M. Gray. Differential vector quantization of achromatic imagery. In *Proceedings of the International Picture Coding Symposium*, March 1983.

[7] T. Murakami, K. Asai, and E Yamazaki. A design of vector quantizer for video signals. In *Proceedings of the International Picture Coding Symposium*, March 1983.

[8] R B Lambert, W P Cockshott, and R J Fryer. A scalable neural architecture combining unsupervised and suggestive learning. In *International Conference on Neural Networks and Genetic Algorithms*, pages 167–176, Innsbruck, April 1993.

[9] R B Lambert, R J Fryer, and W P Cockshott. Recognition system, April 1994. Patent IPN WO 94/24636.