REAL-TIME VERY LOW BIT RATE VIDEO CODING WITH ADAPTIVE MEAN-REMOVED VECTOR QUANTIZATION

Dietmar Saupe, Bernd Butz

Universität Freiburg, Institut für Informatik, Am Flughafen 17, 79110 Freiburg, Germany

ABSTRACT

This is a contribution for very low bit rate video coding in real-time and software only on computers of PC class. It is based on frame replenishment with block coding using mean-removed vector quantization. No motion compensation is employed and the VQ codebook is small. Algorithms are developed to minimize the bit rate and to reduce the search complexity for the vector quantizer. The codebook is adaptive ensuring good overall quality encodings for head-and-shoulder image sequences. Results are provided for some test sequences and compared to some other state-of-the-art techniques.

1. INTRODUCTION

For image sequence encoding only relatively little work has been reported for methods that provide real-time and software-only implementations for video phone applications. We have developed a codec for circuit switched applications with the following goals in mind.

1. The bit rate is scalable within a range from about 10 kb/s up to 100 kb/s and more.
2. The bit rate per frame is constant.
3. The codec is fast, so that a real-time implementation on a PC platform can work in software only.

Based on these constraints we consider in this paper only YUV color image sequences in 4:2:0 QCIF format (176 $\times$ 144 pixels) where U and V chroma channels are subsampled by a factor of 2 horizontally and vertically. We aim at frame rates of about 8 frames per second. This gives a rate of 0.05 to 0.5 bpp (bits per pixel) corresponding to channel bit rates of 10 to 100 kb/s. The method can easily be modified to encompass a broader range of parameters.

There are a number of contributions reporting progress in the direction of our goals, two of them carrying a close relation to the method proposed here. Wang et al [2] propose plain vector quantization (VQ) of 8 $\times$ 8-blocks with an adaptive codebook. Blocks are encoded using motion compensation if appropriate. A constant reproduction quality is targeted and, thus, the bit rate is not constant. Nicholls and Monro (see, e.g., [1]) discuss a method that yields a constant bit rate. It uses fractal and DCT block coding for image sequences. In this work we show that it is advantageous to combine adaptive vector quantization with the constant bit rate encoding strategy. Together with further significant enhancements large gains in reconstructed image quality can be achieved over the transform based approach.

2. CONSTANT RATE ADAPTIVE VQ IMAGE SEQUENCE CODING

Bit rate and time manager. The target bit rate and the frame refresh rate together define the bit budget and the time available for the coding of a single frame. The overall control of the algorithm is executed by the bit rate and time manager. It works much in the same way as the corresponding construction in [1]. A new frame is read in and compared blockwise with the frame buffer, which contains the previously encoded frame (available at the decoder, too). Blocks are of fixed size, e.g., 4 $\times$ 4 pixels. The blocks are sorted with respect to decreasing mean square difference. From this priority list blocks are encoded and transmitted until the allocated bit budget or the available time is used up. This scheme is a type of *frame replenishment,* where only blocks with the largest deviations from the previously encoded frame are updated, while the other blocks remain unchanged.

Mean-removed vector quantization. The block encoding is done using vector quantization with an adaptive codebook of blocks. This is similar to the scheme proposed in [2] but differs from it in some important aspects. Firstly, since CPU time is at a premium no motion compensation is attempted. Secondly, blocks are encoded using mean-removed VQ [3]. This product code technique effectively enlarges the number of possible reconstruction vectors without requiring the increase of computing time that is necessary with plain VQ (used in [2]) in order to arrive at about the same quality. Thus, there is a scalar codebook for uniformly quantized block mean values and a shape codebook for mean-removed blocks. In the following we use the word 'codebook' for the vector codebook of these shape blocks.

Codebook update. For each coded frame a certain target quality tol will be computed (see below). In the case

that a given block cannot be represented sufficiently well by mean-removed VQ (i.e., the corresponding mean square error $MSE > tol$) the quantized mean and block (with quantized mean removed) are transmitted. The shape block is encoded using DPCM of quantized prediction errors and entropy coding. Moreover, this new quantized shape block is included in the codebook while another one is removed. We call this process a codebook update.

Codebook adaptation. The codebook is maintained in the form of a priority queue with the goal to collect the more frequently used blocks at the front. This serves two purposes:

1. Codebook block indices referenced by the encoder are transmitted as an entropy coded bit stream. Thus, keeping the frequently used blocks up front reduces the entropy and improves the rate.
2. For each block that requires an update in a given frame the encoder must search the codebook for an acceptable matching block. This search can take advantage of the sorting of the codebook, speeding up the algorithm (see below).

In practice, the codebook can be initialized either as empty or as a codebook obtained from a training sequence. The blocks in the codebook are sorted according to a biased frequency count, which is incremented by one each time the corresponding vector is referenced by the encoder. After such an incrementation the sorting is reestablished by moving the block forward by one position, if necessary. When a new codebook vector is inserted into an already full codebook, the vector with the smallest count is deleted. The new block might be used in several future frames. Thus, it is not advisable to append the new block at the end of the list, initializing its frequency counter with the value one. On the other hand, the move-to-front rule adopted in [2] causes a lot of unfavorable fluctuation at the beginning of the codebook and widens the histogram of used indices. Thus, guided by experience, we propose a compromise by initializing the frequency count to $k_{min} + (k_{max} - k_{min})/4$ (rounded to an integer) where k_{max} and k_{min} denote the maximal and minimal counts found at the beginning and the end of the codebook.

Adaptation of target quality. The bit budget for a frame will be used up for

1. Bits for addresses of blocks to be updated.
2. For each block to be updated:
   a. Bits for the block mean,
   b. 1 bit for the VQ/codebook update alternative,
   c. either bits for an index to a VQ codebook entry, or bits for all quantized block pixels.

The alternative VQ versus codebook update is decided by the tolerance tol mentioned above which may be interpreted as a target quality for the mean square error. A high value of tol will lead to more blocks being encoded using VQ while a small number forces a larger number of more costly codebook updates. Since the bit budget per frame is constant the setting of tol thus affects the total number of blocks sent. When due to a strict setting of tol only a small number of blocks are sent, then other blocks that would require an update cannot be dealt with due to the limited bit budget. In this case better overall distortion for the current frame would have been achieved by using a larger value for the tolerance tol. Therefore we propose an adaptive tolerance which takes into account the overall need for updates, which can be estimated by the mean square difference (ms-diff) between the current frame and the previous decoded frame. We set $tol =$ max (30, min (ms-diff, 150)).

Color. The U and V color components are encoded in much the same way as the luminance blocks. We use the same size blocks and the current luminance VQ shape codebook. We find that it is not necessary to carry out codebook updates for color. In fact, the encodings presented here use only the mean of U,V-blocks with satisfying results. For each frame the encoding of the luminance has priority. This means that it occurs first and in the case that a large number of codebook updates or a large amount of CPU time is spent for luminance encoding some or all of the color blocks cannot be transmitted due to exhausted bit rate or computing time.

Entropy coding. In our coder we employ entropy coding using variable length codewords. There are different Huffman tables used for the VQ codebook indices, the quantized block mean values, and for the quantized prediction errors for the components of the blocks which are inserted in the codebook. The block locations of the transmitted blocks are run length encoded with Huffman entropy coding of the runs. The choice for Huffman entropy coding was motivated by the real-time constraint.

Fast codebook search. For the codebook search we combine two known techniques and additionally take advantage of the sortedness of the codebook. For the codebook search we employ partial distortion elimination (PDE, [3]) enhanced by sorting the components of query vectors according to decreasing absolute value. Moreover, we remark, that it is not necessary to retrieve the absolutely best codebook block. It suffices to identify a block that meets the tolerance criterion $MSE < tol$, if such a block exists in the codebook. Due to the sortedness of the codebook such a block can often be found near the start of the codebook. This brings about a speedup (by a factor of about 1.7 w.r.t. PDE in our experiments) at the expense of a slightly worsened distortion. In practice, we define a uniform partitioning of the codebook and test the acceptance criterion for the currently optimal codebook block each time after scanning of a segment has been completed.

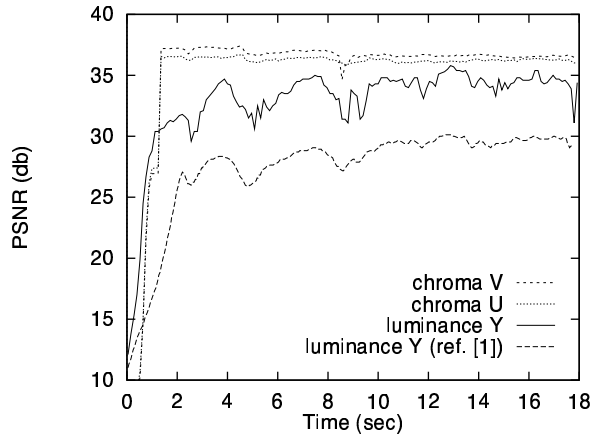

Figure 1: Time course of PSNR for YUV components of the Salesman sequence at 50 kb/s, compared to [1].

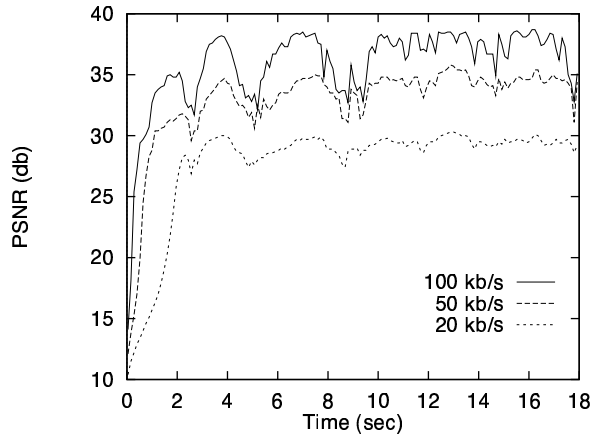

Figure 2: Time course of PSNR for the salesman sequence.

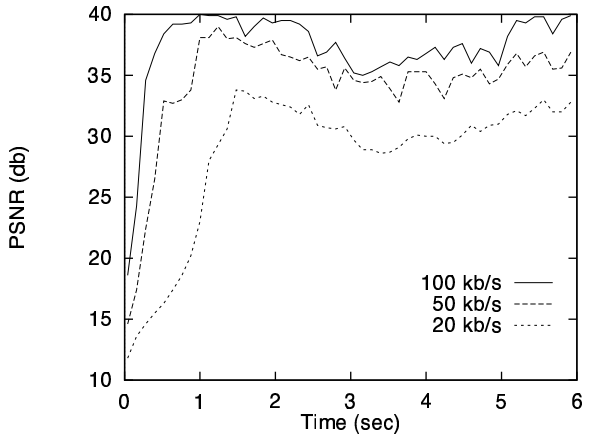

Figure 3: Time course of PSNR for the Miss America sequence.

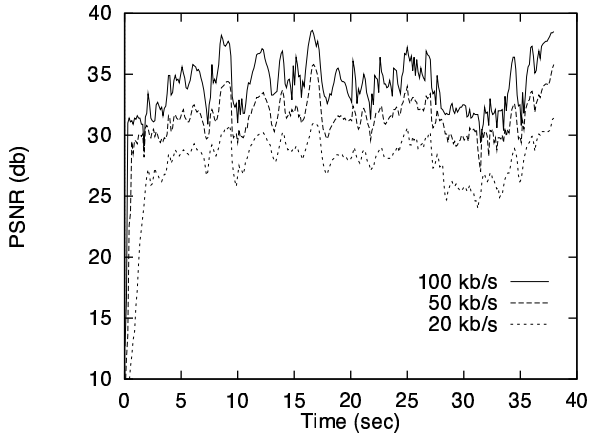

Figure 4: Time course of PSNR for the Mother-and-Daughter sequence.

3. RESULTS AND COMPARISON

We have developed an implementation for 4:2:0 QCIF format color image sequences. The program runs on an Intel Pentium 133 MHz processor. Encoding parameters were:

- block size: 4×4 pixels,
- number of quantization levels for mean values: 64,
- number of quantization levels for DPCM prediction error for block components: 64,
- VQ codebook size: 512,
- frame frequency: 8.3 Hz,
- color coding: means only.

All our encodings were carried out in real-time, often with an ample supply of CPU time left over for each frame. The peak-signal-to-noise ratios (PSNR) are generally computed using only the luminance channel.

Although there are some commonly used test image sequences, it is hard to compare experiments such as ours to others found in the literature because parameter choices are numerous and diverse. The encodings of the Salesman sequence in [1] are best corresponding to ours in terms of parameter settings. The only difference is the slightly lower spatial resolution in [1] (160×128 pixels). Figure 1 compares the performance for the sequence of 18 seconds. Our results are superior to those based on fractal and DCT block encodings provided in [1] (lower curve in figure 1).

Bit rate	Average PSNR (dB)	
(kb/s)	Method [1]	Our method
20	24.8	29.2
50	28.7	33.8
100	30.6	36.6

The table above shows the average PSNR from different encodings. The average is computed ignoring a transient initial phase of two seconds.[1]

[1]The numbers for [1] and the lower graph in figure 1 were obtained

Figure 5: A frame of the original salesman sequence.

Figure 6: Same frame coded at 50 kb/s.

We also provide encoding results for different bit rates and image sequences in figures 2 to 4.

Wang et al [2] encoded the Suzie sequence at 8 Hz and obtained an average PSNR of about 35 dB at an average rate of about 830 kb/s. Our method yields the same average PSNR at constant rate of 168 kb/s and 8.3 Hz. However, we remark, that this comparison is not fully conclusive, since their encodings used higher resolution data (480 $\times$ 720 pixels), which, however, on the other hand introduces an increase of redundancy and makes the encoding easier. Moreover, it is not clear that their encodings can indeed be carried out in real-time on a machine of PC type.

Our coder may well compete with some others that do not impose the real-time constraint. For example, Zhang et al [4] employ a sophisticated motion compensation scheme in combination with mean-removed residual vector quantization. They also encode the Salesman sequence (CIF resolution, 15 Hz) and report an average performance of about 34 dB PSNR with a variable bit rate ranging roughly between 90 kb/s and 270 kb/s. Our real-time method yields the same average PSNR at 12.5 Hz and the constant rate of only 63 kb/s. However, for the comparison it must be taken into account that our encoding was done using the QCIF format.

4. CONCLUSION

We have presented a new technique for very low constant bit rate encodings of image sequences in real-time and software only. The method consists of frame replenishment with mean-removed VQ encodings of blocks with large activity estimates. Special adaptive codebooks provide good visual quality and raise the efficiency of the entropy encoding of the indices. An implementation demonstrates that the method outperforms some comparable current state-of-the-art systems. The improvements of the proposed method over comparable previous ones are due to:

- use of adaptive product code VQ in place of ordinary VQ or nonadaptive block transform coding,
- more efficient adaptivity of the VQ codebook,
- fast codebook search techniques,
- image quality adapted to global frame activity,
- incorporation of entropy coding of VQ indices, codebook block updates, and block positions.

Future work will include postprocessing at the decoder and an expansion of the product code technique by using rotated and reflected blocks thereby further increasing the reconstruction quality without requiring excessive extra encoding and decoding time. With enlarged CPU power of future PC generations it will become possible to improve the encoding by including motion compensation and/or recursive subband transformation. Furthermore, the quantizer for the codebook block update should be made adaptive to the global scene activity.

5. REFERENCES

[1] Nicholls, J., Monro, D., *Scalable video by software*, in: *Proceedings IEEE Intern. Conf. Acoustics, Speech, Signal Proc.* (ICASSP), Atlanta, 1996.

[2] Wang, X., Shende, S., Sayood, K., *Online compression of video sequences using adaptive VQ codebooks*, in: *Proceedings DCC'94 Data Compression Conference, 1994, Snowbird, Utah,* J. A. Storer, M. Cohn (eds.), IEEE Computer Society Press, 1994.

[3] Gersho, A., Gray, R.M., *Vector quantization and signal compression,* Kluwer, Boston, 1992.

[4] Zhang, K., Bober, M., Kittler, J., *Robust motion estimation and multistage vector quantization for sequence compression*, in: *Proceedings IEEE Intern. Conf. Image Proc.* (ICIP), Austin, 1994.

using the plot provided in [1].