# Low-Complexity Scalable Video Coding through Table Lookup VQ and Index Coding

Marco Cagnazzo, Giovanni Poggi, and Luisa Verdoliva

Università Federico II di Napoli
Dipartimento di Ingegneria Elettronica e delle Telecomunicazioni
Via Claudio, 21 – 80125 Napoli, ITALY
Fax +39 081 768.31.49, Phone +39 081 768.31.51
{cagnazzo, poggi, verdoliv}@unina.it

**Abstract.** The Internet community is very heterogeneous in terms of access bandwidth and terminal capabilities, hence, there is much interest for low-computation, software-only, scalable video coders that guarantee universal access to video communication. Scalability allows users to achieve a fair quality of service in relation to their resources. Low complexity, on the other hand, is necessary in order to ensure that also users with low computing power can be served.
In this work, we propose a multiplication-free video codec, whose complexity is much reduced with respect to standard coders at the price of a limited increase in memory requirements. To this end we resort to very simple coding tools such as table lookup vector quantization (VQ) and conditional replenishment. We start from the simple coder proposed in [1], which already guarantees high scalability and limited computational burden, and improve upon it by further reducing complexity, as well as the encoding rate, with no effect on the encoding quality. The main innovation is the use of ordered VQ codebooks, which allows the encoder to generate *correlated* indexes, unlike in conventional VQ. Index correlation, in turn, allows us to carry out conditional replenishment (the most time-consuming operation in the original coder) by working on indexes rather than on block of pixels, and to reduce drastically its complexity. In addition, we also take advantage of the correlation among indexes to compress them by means of a predictive scheme, which leads to a 15-20% rate reduction in the base layer, without significant increase in complexity. Thanks to these and other minor optimizations we have obtained improved performance and, more important, a 60-70% reduction of the encoding time (on a general purpose machine) with respect to [1].

## 1 Introduction

The last decade has witnessed an exponential growth of the information and communication technology, with the huge diffusion of Internet and mobile communications. Yet, expectations about the advent of widespread broadband access have fallen short, long awaited UMTS networks have yet to be deployed, and most end users keep accessing voice and data networks through narrowband channels. On the other hand, not even UMTS, when available, will provide universal wideband access for free, and it is quite likely that bandwidth shortage will keep being an issue, at least for mobile-service users.

F. Boavida et al. (Eds.): IDMS/PROMS 2002, LNCS 2515, pp. 166–175, 2002.

In such a scenario, the quest for efficient video coding techniques is more urgent than ever, as they will allow for the provision of interactive video services over the existing networks. In particular, given the wide variety of access channels and terminals that can be envisioned, scalability (spatial, temporal, and SNR) and low computational complexity are very important for such algorithms.

The major current video coding standards, like MPEG-1/2 and H.261/263, guarantee a very good performance in a wide range of conditions but only a fair level of scalability, and present a significant encoding complexity that cannot be reduced too much without a complete change of approach. This is all the more true for MPEG-4, due to the need of complex image analysis algorithms to extract video objects. Under this point of view, wavelet-based techniques (e.g., [2,3]) appear more promising, due to the efficient implementation of the wavelet transform and the availability of high-performance fully scalable coding algorithms such as EZW and SPIHT.

However, wavelet-based techniques still require a 3d transform, which can be too demanding, both in terms of complexity and memory requirements, in certain situations. The mobile-user scenario is a good such example, since all resources, and in particular computing power, are severely constrained by sheer terminal size, and hence simpler symmetric encoding and decoding algorithms are required. This motivates our work towards the development of a symmetric scalable codec that does not require any multiplication at all, thus allowing a real-time video communication on low-power terminals.

To this end we must resort to a very simple compression scheme, even though this entails some performance loss in terms of increased rate or impaired reproduction quality. In particular, our work moves from the all-software video coding system proposed by Chaddha and Gupta in [1], based on conditional replenishment (CR) and hierarchical vector quantization (HVQ) [4,5]. Such a coder, referred to as CG coder from now on, is scalable in space/time-resolution as well as reproduction quality, thus adapting to a wide range of bandwidths, and provides an embedded encoded stream to allow for multicast services. In addition, it has a very limited complexity, because the HVQ coder uses only table lookups, and time-consuming motion compensation is not performed. A different video codec based on HVQ was proposed in [6] but it was definitely more complex, including also motion compensation, although more accurate than [1].

In this work we improve upon the basic CG coder, further reducing both its computational complexity and its encoding rate. The key idea is to exploit the correlation among VQ indexes that appears when an ordered codebook is used [7]. This will allow us to simplify the conditional replenishment check (a relatively complex step in this codec) and to efficiently entropy encode the VQ indexes themselves, thus reducing the encoding rate. Furthermore, even the filtering and interpolation steps, necessary in the pyramidal encoding scheme to guarantee spatial scalability, will be carried out using only table-lookups, all but eliminating computation, at a cost of a negligible performance degradation.

In Section 2 the CG coder is briefly revised, Section 3 illustrates the proposed improvements and, finally, Section 4 shows a few sample experimental results.

## 2 The Chaddha-Gupta Coder

The CG coder uses conditional replenishment to exploit temporal redundancy and vector quantization to take advantage of spatial redundancy. The choice of CR instead of motion compensation is dictated by the need to reduce complexity, as the accurate estimation of motion vectors is usually quite expensive. Of course, rate-distortion performance suffers from this choice but, if videotelephony and videoconference are the intended applications, where the typical scene has a large fixed background, the performance gap can be quite small. Using VQ in a low-complexity coder, instead, might look paradoxical, as is well-known that VQ's major weakness is just its exceedingly high computational burden.

In vector quantization a set of template blocks (or codewords) called codebook is designed off-line, and for each input block the encoder must single out in the codebook the minimum distance codeword. Once the best matching codeword is found, only its index (a single scalar) is sent and used by the decoder to access a local copy of the codebook and approximate the input block.

Unfortunately, looking for the best matching codeword requires computing a number of vector distances which is quite expensive and hardly affordable in a real-time system. In hierarchical VQ, however, all computation is made off-line and for each possible input block the appropriate codeword is selected at run time only by means of table lookups. Of course, a table with an entry for each possible input block would be prohibitively large, which is why encoding is performed by means of repeated accesses to a hierarchy of small tables. As an example, to encode an 8-pixel block at 0.5 bit/pixel with HVQ only 7 memory accesses (to three 64-kbyte tables) are typically required (see Fig. 1) and no mathematical operation, as compared to the 256 multiplications and additions required in full-search VQ. A thorough description of the table design procedure can be found in [5]. The price to be paid for such a smooth encoding is a limited performance impairment (usually less than 1 dB) with respect to unconstrained VQ.

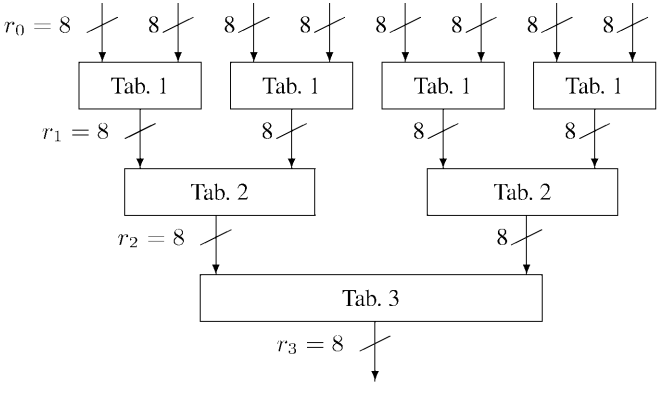


**Fig. 1.** A 3-stage HVQ encoder

The basic encoder can therefore be described as follows. The input frame $X$ is divided in macroblocks (MBs) and each MB is compared with the homologous MB in the reference encoded/decoded frame $\widehat{X}_R$. If a suitable distance between them (e.g., the euclidean distance) is below a given threshold, the current MB is declared "fixed" and not coded, but reproduced as the reference MB. Otherwise, the MB is further divided in smaller blocks each of which is coded by HVQ and represented by an index in the VQ codebook.

To further reduce the bit-rate, and also to adapt to limited-resolution terminals, the CG coder provides for three types of scalability, briefly sketched here (see [1] for more detail). Spatial scalability is ensured by resorting to a Laplacian pyramid decomposition: low bandwidth/resolution users receive only the base layer of the pyramid, and only when more resources are available an enhancement layer at double resolution is added. A third level of resolution is obtained only by interpolation. Likewise, temporal scalability is obtained by using several embedded layers: low bandwidth users get only every eighth frame, and add intermediate frames (every fourth, every second, etc.) as more bandwidth is available. Finally, SNR scalability is obtained by using tree-structured (hierarchical) VQ and sending only a certain number of bits for each VQ index.

The coder outputs an embedded scalable bit-stream, that is exploited to provide efficiently a multicast video service by means of the multiple multicast groups (MMG) concept, just as is done in [8].

## 3 Proposed Improvements

A CPU-time analysis of the CG coder (see Table 3 later on), conducted on a general purpose machine, shows that almost 50% of the encoding time is devoted to carry out the conditional replenishment, and almost all the rest is spent on filtering and interpolation required by pyramidal coding. By contrast, HVQ complexity is quite negligible. Our first goal, therefore, is to cut CR complexity, and this is achieved by resorting to ordered-codebook VQ.

### 3.1 Ordered Codebooks

Usually, when we have to design a VQ codebook, the only goal is to choose a set of codewords $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ that guarantee the smallest possible average encoding distortion. We impose here an additional constraint, that codewords with close indexes are similar and vice versa, namely

$$|i-j| \text{ small} \quad \Leftrightarrow \quad \| \mathbf{y}_i - \mathbf{y}_j \|^2 \text{ small}$$

Such a statement is necessarily vague, as it amounts to requiring some kind of continuity in the codeword-to-index mapping, which is clearly impossible. Nonetheless, the design of ordered VQ codebooks is a well-known and well-understood topic [7], and can be easily accomplished by rearranging a generic codebook or, better yet, by designing an ordered codebook from scratch by the Kohonen algorithm [9]. The algorithm starts with an arbitrary initial codebook; then, a large number of training vectors, $\mathbf{x}(k), \ k = 1, 2, \ldots,$ are examined sequentially and, for each of them, all codewords[1] are gradually

[1] Not just the best matching as happens with the popular K-means algorithm.

updated according to the rule

$$\mathbf{y}(i) = \mathbf{y}(i) + \gamma(k, d)\ [\mathbf{x} - \mathbf{y}(i)]$$

until convergence is reached. Here, $\gamma(k, d)$ regulates the speed of adaptation and the ordering of the codebook and decreases both with time $k$, to ensure convergence, and with the index distance from the best matching codeword $d = |i - i_{\mathrm{BM}}|$, to ensure the desired codebook ordering. With a careful tuning of parameters, The Kohonen algorithm has proven [7] to guarantee both low encoding distortion and a satisfactory codeword ordering. An example is shown in Fig.2, where a 256-codeword ordered Kohonen codebooks is shown.

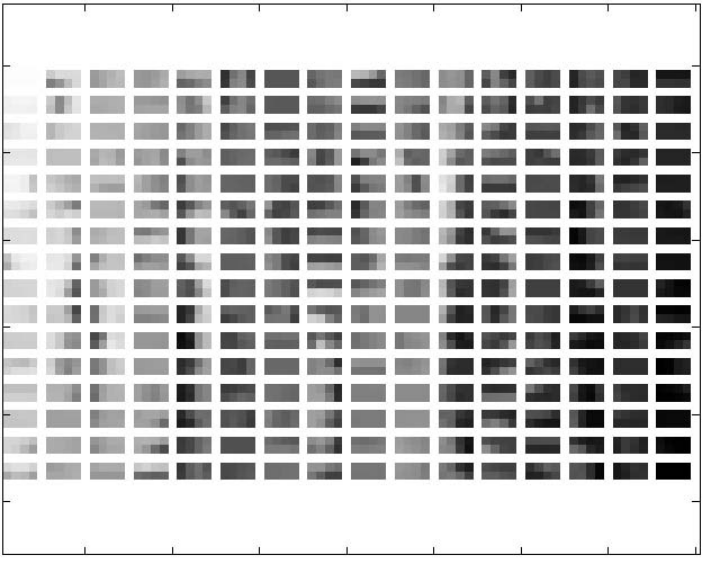

**Fig. 2.** Kohonen codebook

### 3.2 Index-Based Conditional Replenishment

Now it is easy to reduce CR complexity. To take a fixed/moving decision for a given block MB we should first evaluate the sum of euclidean (or other) distances of all component VQ blocks from their counterparts in the reference frame,

$$\sum_{k \in \mathrm{MB}} \| \mathbf{x}(k) - \mathbf{x}_R(k) \|^2$$

and then compare it with a threshold.

However if an ordered codebook is available, where the pseudo-continuity holds, than the index distance is a faithful indicator of vector distance, and we can take the decision based only on the few VQ indexes of a MB rather than on all individual pixels. More precisely, for each MB, all component VQ blocks are quantized (only table lookups), their indexes are compared with the corresponding indexes of the reference MB, and a distance measure is computed and compared with a threshold

$$\sum_{k \in \mathrm{MB}} |i(k) - i_R(k)| \leq T$$

Only when this test fails VQ indexes are actually sent, otherwise the reference MB is copied.

Note that, if we did not use an ordered codebook, we could spare transmitting only VQ indexes that remain *exactly unchanged* between successive frames, as proposed in [10] and again in [11]. With our CR technique this would happen for a CR threshold $T = 0$.

### 3.3 Index-Predictive Vector Quantization

By resorting to an ordered VQ codebook we can also obtain a bit-rate reduction in the spatial domain without any quality loss. In fact, spatially neighboring blocks are strongly correlated and, therefore, their corresponding VQ indexes will be correlated as well if an ordered codebook is used. One can exploit such a correlation through a simple predictive encoding scheme [7]. The index of the current codeword is predicted from some neighboring indexes already known to the receiver. More precisely, the prediction is equal to the index above or beside the current one, based on which one is actually available at the receiver and, if both are present, which one is expected to provide the best prediction. The prediction error is then entropy encoded (another table lookup) by means of Huffman coding.

### 3.4 Table Lookup Filtering and Interpolation

A further reduction of complexity can be obtained by performing antialias filtering and interpolation via table look-up. In our current implementation, these operations, needed in order to implement pyramidal coding, require only the evaluation of image sample means. For example, we have to compute the mean value of four high-resolution layer samples in order to obtain a base layer sample of the Laplace pyramid, and also our simple bilinear interpolation requires some sample mean evaluation (see figure 3). This can be carried out by using a table in which the means between every possible couple of input values are stored. As the input values are bytes, and the mean is also stored as a byte, this table requires 64KB of memory. Table lookup filtering introduces a small error since we use only 8 bits instead of 9 bits to store the mean value between 2 bytes. However, such error is totally negligible with respect to the error introduced by CR and VQ, as also confirmed by experimental results. Note that table look-up implementation is also amenable for longer filters and some performance improvement can be expected, although more memory will be required.

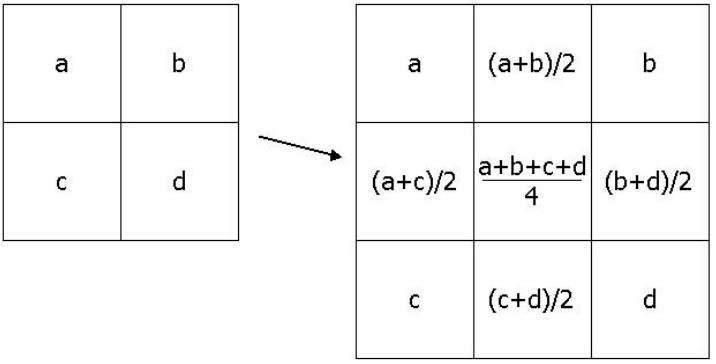


**Fig. 3.** Interpolation scheme

### 3.5 Computational Complexity of the Proposed Scheme

As already noted, the proposed improvements make our algorithm multiplication free, but also cause a more intensive use of memory. Therefore, this approach does improve encoding speed only if memory accesses are significatively faster than multiplications. As a matter of fact, even though last years have been characterized by a fast increase in CPU speed and a slower improvement in memory performance, memory access operations are still much faster than multiplications. In addition, one could even develop terminals whose hardware fully exploits the table lookup nature of the proposed algorithm.

In any case, it is interesting to analyze the complexity of the proposed algorithm and of the original CG-encoder, so as to foretell their behavior once the hardware characteristics are known. In table 2 theoretical computational complexity is evaluated for each encoder in terms of how many and which operations are needed for every base level pixel in order to encode both resolution levels. The meaning of all symbol is reported in table 1 ($\overline{c} = 1 - c$). However, note that in the CG coder, CR requires floating point multiplications, while filtering and interpolation need integer multiplications and therefor their cost is quite different.

Although a direct relationship between theoretical complexity and execution time cannot be established, the total elimination of multiplications, and heavy reduction of sums and tests in favor of memory accesses will likely entail a much faster encoding on most hardware platforms.

## 4 Experimental Results

We have implemented the original Chaddha-Gupta coder as described in [1] and then introduced the variations described in Section 3, based on the use of ordered codebooks designed by means of the Kohonen algorithm. Here we report some experimental results obtained on two 180-frame videoconference-like monochrome test sequence (see Fig.4 and 5).

**Table 1.** Meaning of symbols in Table 2

| Symbol | Meaning |
|---|---|
| $c_b$ | CR success fraction in base level |
| $c_e$ | CR success fraction in enhancement level |
| $R$ | pixel per vector |
| $N$ | pixel per MB |
| $\sigma$ | sum |
| $\pi$ | product |
| $\mu$ | memory access |
| $\lambda$ | logic operation |

**Table 2.** Operations required for each base level pixel

| technique | Filtering | HVQ | CR | Interp. |
|---|---|---|---|---|
| CG | $15\sigma + 5\pi$ | $(\overline{c_b} + 4\overline{c_e})\frac{R-1}{R}\mu$ | $5\sigma + 5\pi + \frac{5}{N}\lambda$ | $5\sigma + 3\pi$ |
| Proposed | $15\mu$ | $\frac{5R-4}{R}\mu + \frac{3}{R}\sigma + \frac{1}{R}\lambda$ | $\frac{10}{R}\sigma + \frac{5}{N}\lambda$ | $5\mu$ |

**Table 3.** Computation time comparison (ms)

| technique | I/O | Filtering | HVQ | CR | Interp. | total |
|---|---|---|---|---|---|---|
| CG | 1.7 | 8.6 | 1.0 | 17.0 | 6.3 | 34.6 |
| proposed | 1.7 | 3.4 | 1.9 | 1.5 | 2.8 | 11.3 |

In Table 3 we report the time spent on each encoding step for a single CIF frame (input/output, filtering and decimation, HVQ, CR, upsampling and interpolation) when the original and modified encoder are used[2]. Index-based CR drastically reduces time spent on CR, and only slightly increases time devoted to VQ (because all blocks are now quantized). Table lookup implementation allows for a significative time reduction in filtering and interpolation with an overall time saving above 67%. Extensive experiments (not reported here) show that this computation-time gap remains pretty much the same for a wide range of CR thresholds. As said before, the relationship between theoretical complexity and execution time is strongly implementation[3] dependent, but these example results are nonetheless encouraging. We also compared our algorithm performance with H.261 standard [12], and found that the proposed encoder is almost an order of magnitude faster than the standard one, but the PSNR decreases significatively, up to 5 dB at comparable rates. Although our main focus is on complexity, this performance gap must be reduced to more reasonable values which can be obtained, in our opinion, by suitably tuning the encoding parameters.

Turning to bit-rate reduction, we have evaluated the entropy of the index prediction error and, for a wide range of operative conditions, a reduction of about 20% with respect to the original 8 bits has been observed. Thanks to this improvement, the rate-distortion

[2] These results have been obtained using a machine equipped with a 2 GHz *Pentium IV* CPU and Linux operating system.

[3] In terms of both hardware and software.

**Fig. 4.** Sequence NEWS: original and encoded frames at 35.8 kbps

**Fig. 5.** Sequence CLAIRE: original and encoded frames at 29.3 kbps

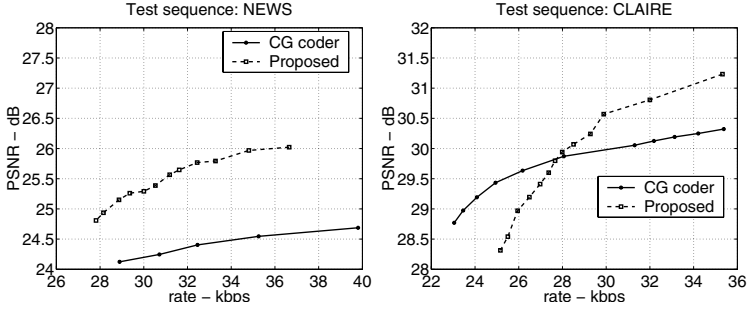


**Fig. 6.** Rate-distortion performance of CG and proposed coder

performance of the modified coder turns out to be superior to that of the CG coder (see Fig.6) in most operative conditions, despite the loss due to the simplified CR. Note that we apply index prediction only to the lower spatial-resolution layer of the coder where a significant index correlation exists, also because this is exactly the layer received by narrowband users, where rate reduction is especially needed.

In conclusion, the use of HVQ for spatial coding, and the extension of the table lookup approach to all remaining processing steps allow for the implementation of a multiplication-free video coder whose encoding quality, although inferior to that of cur-

rent standards, is certainly acceptable for users with very low computation power. In future work we will work on fine-tuning the coder to improve its rate-distortion performance and will study its implementation on current low-power hardware platforms.

## References


1. N.Chaddha, A.Gupta, "A framework for live multicast of video streams over the Internet," Proc. International Conference on Image Processing, pp.1-4, 1996.
2. B.J.Kim, Z.Xiong, W.A.Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)" IEEE Transactions on Circuits and Systems for Video Technology, Dec.2000, pp.1374-1387.
3. J.W.Woods, G.Lilienfield, "A resolution and frame-rate scalable subband/wavelet video coder," IEEE Transactions on Circuits and Systems for Video Technology, Sept.2001, pp.1035-1044.
4. P.C.Chang, J.May, R.M.Gray, "Hierarchical vector quantization with table-lookup encoders," Proc. International Conference on Communications, pp.1452-1455, Chicago (IL), June 1985.
5. N.Chaddha, M.Vishwanath, P.A.Chou, "Hierarchical vector quantization of perceptually weighted block transforms," Proc. Data Compression Conference, pp.3-12, Snowbird (UT), March 1996.
6. K.Mukherjee, A.Mukherjee, "Joint optical flow motion compensation and video compression using hybrid vector quantization," Proc. Data Compression Conference, pp.541, 1999.
7. G.Poggi, "Applications of the Kohonen algorithm in vector quantization", European Transactions on Telecommunications, March/april 1995, pp.191-202.
8. S.McCanne, M.Vetterli, V.Jacobson, "Low-complexity video coding for receiver-driven layered layered multicast", IEEE Journal on Selected Areas in Communications, Aug.1997, pp.993-1000.
9. T.Kohonen, "Self-Organization and associative memory," 2nd Ed., Springer-Verlag, New York, 1988.
10. M.Goldberg, H.Sun, "Image sequence coding using vector quantization", IEEE Transactions on Communications, pp.703-710, 1986.
11. N.B.Karayiannis, Y.Li, "A replenishment technique for low bit-rate video compression based on wavelets and vector quantization", IEEE Transactions on Circuits and Systems for Video Technology, pp.658-663, May 2001.
12. Anonymous ftp ftp://havefun.stanford.edu:pub/p64/P64v1.2.tar.Z