

Simulation results and conclusions: The proposed VLC coding method was compared with the VLC coding method in the TML-1 codec. To demonstrate the validity of the proposed algorithm, the algorithm was applied to two syntax elements, the macro block (MB) type and coded block pattern (CBP). The test sequences were composed of 300 frames of 'Foreman', 'Hall', 'News' and 'Container ship' sequences in the quarter common intermediate format (QCIF) (176 × 144 pixels) at 10 frame/s. A fixed quantisation step size, i.e. $Q_{step} = 4, 5, 7, 10, 15, 25$, was used for all the sequences according to the common test conditions of the H.26L standard [4]. Table 1 shows the VLC table for the MB type in the TML-1 codec. Note that the VLC code for the INTRA MB type was 0000011, i.e. 7 bits and a decimal value of three. In Table 2, the probabilities for each MB type are presented to show the statistical variations according to sequence and coding parameter, i.e. Q_{step} . To measure the performance of the VLC tables, we computed the average number of bits per symbol (bit/symbol):

$$\bar{B} = \sum_{i=0}^{L-1} s(i)p(i) \quad (5)$$

For comparison, we also performed computations using the entropy and mapping-based method by replacing $s(i)$ with $-\log_2 p(i)$ and $s(m(i))$, respectively. Table 2 shows that the symbol probabilities are drastically changed with respect to the sequence and quantisation step size. Also, the coding performances measured in terms of the average number of bits per symbol show that the TML-1 coder gives a relatively poor performance compared with the entropy, which is the theoretically optimal performance value and that there is a high probability that the proposed mapping method can be used to improve the coding efficiency. Note that this mapping-based result is obtained when the symbol probability is known.

Table 3: Average bit rate and percentage reduction in number of bits for MB type

Q	Container		Foreman		Hall		News	
	TML	Prop	TML	Prop	TML	Prop	TML	Prop
4	460	315	555	279	448	270	385	261
5	457	320	557	279	441	261	364	282
7	360	276	511	314	350	161	319	279
10	324	236	488	327	333	150	296	275
15	264	226	455	324	259	212	264	260
25	168	170	373	275	145	142	198	194
[%]	24.10		38.82		39.47		15.06	

Table 4: Average bit rate and percentage reduction in number of bits for CBP information

Q	Container		Foreman		Hall		News	
	TML	Prop	TML	Prop	TML	Prop	TML	Prop
4	761	407	675	275	761	237	564	404
5	734	413	676	301	795	304	515	370
7	683	387	668	349	828	429	455	317
10	634	444	661	433	781	581	419	317
15	458	395	604	500	408	367	347	300
25	141	132	363	343	87	80	181	170
[%]	36.15		39.65		45.41		24.31	

The results in Tables 3 and 4 are obtained by applying the proposed algorithm presented in the preceding Section, using the probabilities computed from the previously encoded data. The parameters (L, W) for the MB type and CBP are (9, 1) and (48, 5), respectively. Table 3 shows the average bit rate comparison for the MB type between TML-1 and proposed method. Each value in Table 3 represents the average number of bits per frame and the columns and rows represent the Q_{step} size and coding method used in the simulation, respectively. From these results, the proposed method gives a ~30% reduction in the overall bit rate over the TML-1 method. In the Table, the improvement is reduced as the quantisation step size is increased, implying that TML-1

method is designed for low bit rate applications, though H.26L covers high bit rate applications [4, 5]. A similar trend is also observed for CBP information, as shown in Table 4.

From these simulation results, we conclude that the symbol probability is drastically changed by the sequence and coding parameters and that the proposed method gives a remarkable improvement over the conventional VLC coding method.

© IEE 1999

10 September 1999

Electronics Letters Online No: 19991452

DOI: 10.1049/el:19991452

Kook-yeol Yoo (Information Media Laboratory, Corporate R&D Center, Samsung Electronics Co., Ltd., Korea)

E-mail: kyoo@mnmrnd.sec.samsung.co.kr

Byung Sun Choi (Department of Electrical Engineering, Korean Advanced Institute of Science and Technology (KAIST), Korea)

References

- JAIN, A.K.: 'Fundamentals of digital image processing' (Prentice Hall, Inc., 1989)
- BJONTEGAARD, G.: 'Enhancement of the telenor proposal for H.26L'. ITU-T Q.15/16, February 1999, Monterey, Paper Q15-G25
- BJONTEGAARD, G.: 'Enhancement of the telenor proposal for H.26L'. ITU-T Q.15/16, July 1999, Berlin, Germany, Paper Q15-H10
- SULLIVAN, G.: 'Draft meeting report of the eighth meeting (meeting H) of the ITU-T Q.15/16'. ITU-T Q.15/16, July 1999, Berlin, Germany, Paper Q15-H37d1
- SULLIVAN, G., and HIBI, K.: 'Draft call for proposals for H.26L video coding'. ITU-T Q.15/16, Geneva, January 1998

Adaptive type II hybrid ARQ scheme using zigzag code

K.S. Chan, Li Ping and S. Chan

The authors propose a new hybrid type II ARQ scheme based on the simple multidimensional concatenated zigzag code. Owing to the fact that the error-correcting capability of zigzag codes is far better than that of conventional convolutional codes, the throughput of our scheme is better than existing hybrid ARQ schemes.

Introduction: Automatic repeat-request (ARQ) schemes are commonly used in data communication systems in order to provide reliability of communication. Many recent research projects have focused on type II hybrid ARQ schemes [1, 2]. In communication systems where the channel state temporarily varies, it is preferable to change the error-correcting capability in the hybrid ARQ scheme according to the channel state. In [1], the author proposed a new hybrid ARQ scheme with adaptive forward error correction called the adaptive incremental redundancy (AIR) scheme. This scheme is based on the punctured and repetition convolutional code. The basic idea is that the coding rate for error correction is varied according to the system parameters. It has been shown that the performance of this scheme is better than that of other ARQ schemes [1]. However, there are still two issues for this scheme. The first is the error-correcting capability of the convolutional code. The convolutional code is not the best forward error correcting code. The coding gain of the turbo code may be much higher. Since the throughput of the ARQ scheme is related to the coding gain of the code used, a better choice of code may lead to further improvement in the throughput. Another issue is related to the decoding complexity of the punctured code. The decoding complexity of the punctured code is almost the same as the non-punctured code. This means the decoding complexities are the same for both the good and bad channels. However, intuitively, we would expect a lower decoding complexity for a good channel since a simpler code can be used in this case.

In this Letter, we propose a new adaptive hybrid ARQ scheme to address the above two issues. We will adopt the multidimensional zigzag code proposed in [3]. The simulation results in [3] show the error-correcting capability of this code is comparable to

that of two-dimensional turbo codes, while the decoding is much simpler. The decoding complexity is also lower for a higher coding rate. The simulation results show the improvement in the throughput is substantial compared to the AIR scheme. As the decoding of the multidimensional zigzag code is much simpler than that of the convolutional code, our scheme is also easy to implement.

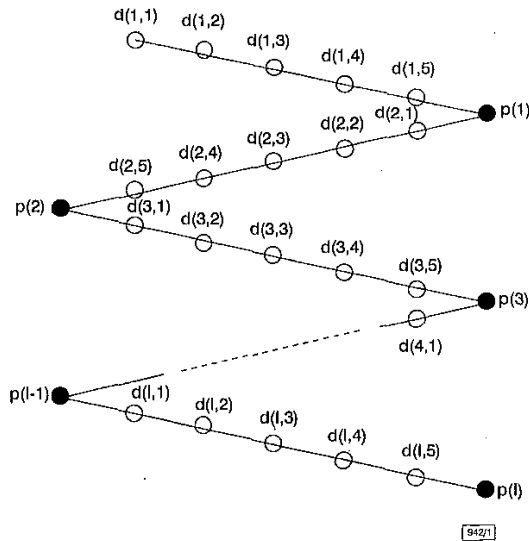


Fig. 1 Graph representation of zigzag code

Scheme description: The zigzag code is illustrated in Fig. 1. Here white nodes represent information bits: $\{d(i, j)\}$, $i = 1, 2, \dots, I$, $j = 1, 2, \dots, J$. Black nodes represent parity bits: $\{p(i)\}$, $i = 1, 2, \dots, I$. We refer to $[p(i-1), d(i, 1), d(i, 2), \dots, d(i, J), p(i)]$ as a segment. The parity bits are chosen such that each segment on the graph contains an even number of ones. The multidimensional concatenated zigzag code is shown in Fig. 2. In this Figure, D is an $I \times J$ array of information bits, and $D_n = \pi_n(D)$ is an $I \times J$ array obtained from D using interleaving π_n , $n = 1, 2, \dots, N$. The zigzag coding scheme is applied to produce the parity check vectors p_n for each D_n . The overall encoding and decoding algorithms of the concatenated zigzag code can be found in [3].

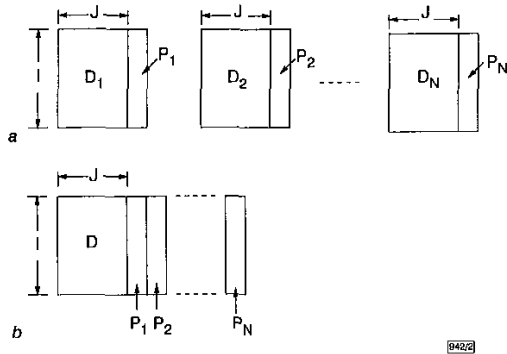


Fig. 2 Multidimensional zigzag code

a Encoding process
b Overall codeword

We now describe our type II ARQ scheme. As in [1], an (n, k) block code is used for error detection. Error correction is achieved using the multidimensional zigzag code shown in Fig. 2. It is assumed that a noiseless feedback link is available so that the receiver can reliably inform the transmitter of the packets that can be successfully decoded. Thus when a k -bit information packet needs to be sent, it is first encoded as an (n, k) block code β . This code is then used as the information packet to be encoded in the form of an N -dimensional concatenated zigzag code. If the transmission of the packet includes k levels, the N parity vectors P_1, P_2, \dots, P_N are divided into k groups: G_1 includes P_1 to P_{k_1} ; G_2 includes P_{k_1+1} to P_{k_2} ; ...; G_k includes $P_{k_{k-1}+1}$ to P_N . The operation of the

transmitter at different levels encountered during the transmission of a packet β is described below:

Level 1: Initially, β and the parity vectors included in group G_1 are transmitted in the channel. At the receiver, the received packet is a k_1 -dimensional concatenated zigzag code. After the decoding process, the receiver decodes the (n, k) block code to see if it is error free. If it is, the transmission of β is complete. Otherwise, β moves up to level 2.

Level i , $2 \leq i \leq k$: At this level, the transmitter sends all parity vectors included in group G_i . At the receiver, the received sequence is combined with the previously received sequences for the same data packet for decoding. If the decoding process is successful, transmission of β is complete. Otherwise, β moves up to level $i+1$ for $i < k$ or level 1 for $i = k$.

The above hybrid ARQ scheme can work both for the stop-and-wait and go-back- N ARQ protocols. For the stop-and-wait protocol, the transmitter will not begin transmitting another packet unless the current packet is received correctly. In the go-back- N ARQ protocol, the transmitter will send new packets when the transmitter is waiting for acknowledgement. However, the receiver will not begin decoding new packets before the current decoding packet has been declared successful.

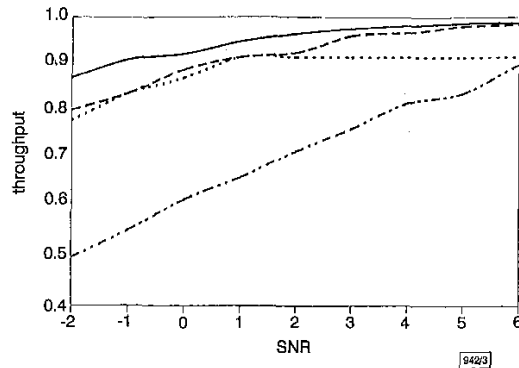


Fig. 3 Throughput against channel SNR for stop-and-wait ARQ scheme

— Sastry with MCD
- - - SW type 2
- · - AIR scheme
— proposed scheme

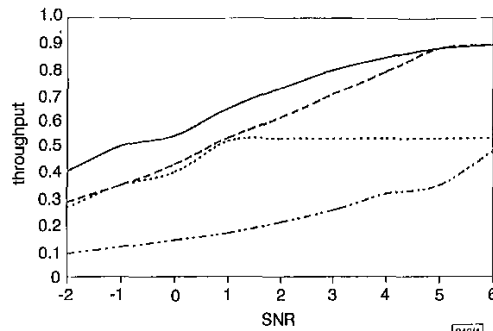


Fig. 4 Throughput against channel SNR for go-back- N ARQ scheme

— Sastry with MCD
- - - GBN type 2
- · - AIR scheme
— proposed scheme

To evaluate the performance of our scheme, we have performed some simulations and compared the results of our scheme with other ARQ schemes. In our simulation, we assume it is a white Gaussian channel, and packets are transmitted in the channel along with γ_b overhead bits required for synchronisation, addressing, etc. The number of channel bits that can be sent during a round-trip delay is denoted by S_b . We assume $n = 500$, $S_b = 5,000$ and $\gamma_b = 50$. We have carried out simulations for stop-and-wait and go-back- N ARQ schemes. Fig. 3 shows the results for the stop-and-wait scheme, while Fig. 4 shows the results for the go-back- N scheme. Both sets of results are measured against the

channel signal-to-noise ratio. The normalised throughput of our scheme, the adaptive incremental redundancy (AIR) scheme proposed in [1], the type II hybrid ARQ scheme with code combining in [2], and the Sastry scheme with multiple copy decoding (MCD) are plotted in these Figures. It can be seen that for a channel SNR of < 5dB, our scheme performs much better than other ARQ schemes. The lower the SNR, the better the improvement.

Conclusions: In this Letter, we have proposed a new type II hybrid ARQ scheme. The new scheme is based on the simple multi-dimensional concatenated zigzag code. The scheme can be easily implemented. The scheme can also automatically adapt to varying channel quality. The simulation results show our scheme is better than other hybrid type II ARQ schemes.

© IEE 1999

14 October 1999

Electronics Letters Online No: 19991429
DOI: 10.1049/el:19991429

K.S. Chan, Li Ping and S. Chan (Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong)

E-mail: kschan@ee.cityu.edu.hk

References

- 1 KALLEL, S.: 'Efficient hybrid ARQ protocols with adaptive forward error correction', *IEEE Trans. Commun.*, 1994, **42**, pp. 281-289
- 2 KALLEL, S.: 'Analysis of a type II hybrid ARQ scheme with code combining', *IEEE Trans. Commun.*, 1990, **38**, pp. 1133-1137
- 3 LI PING, and NAM PHAMDO: 'Zigzag codes and concatenated zigzag codes'. IEEE Information Theory Workshop, Greece, June 1999

Fast nearest neighbour searching algorithm using L_1 norm

Seongjoon Baek, SangKi Kang and Koeng-Mo Sung

Simple but effective fast codebook searching algorithms for vector quantisation are presented. The Euclidean distance (L_2 norm) calculation requires a number of multiplications. The proposed algorithms use the L_1 norm between the input vector and codeword to discard many unlikely codewords. Since the proposed algorithms significantly reduce the number of multiplications, a considerable reduction in encoding time is achieved. Simulation results confirm the effectiveness of the proposed algorithms.

Introduction: Computational complexity is a critical problem of vector quantisation (VQ). The number of multiplications is often used as a measure of the computational complexity of minimum distortion encoding, since it is usually the dominant computation [1]. The average number of multiplications per sample, N , required for full search VQ minimum mean squared error encoding is given by

$$N = 2^{kR}$$

where k is the dimension of the vector and R is the rate in bit/sample. This exponential growth in dimension and rate severely restricts the size of implementable codebooks. In this Letter, we propose a fast encoding algorithm which significantly reduces the number of multiplications.

Proposed algorithms: We will first give a lemma which plays a key role in the proposed algorithm.

(i) **Lemma 1:** Let $\mathbf{x} = (x_1, x_2, \dots, x_k)$ be an input vector and $\mathbf{y} = (y_1, y_2, \dots, y_k)$ be a codeword. If the distortion is the Euclidean distance, then

$$d(\mathbf{x}, \mathbf{y})^2 \geq \left(\sum_{i=1}^k |x_i - y_i| \right)^2 / k$$

(ii) **Proof of lemma 1:** The above equation, $\sum_{i=1}^k |x_i - y_i|$, can be rewritten as $\sum_{i=1}^k p_i |x_i - y_i|$, where p_i is defined as

$$p_i = \begin{cases} +1 & x_i \geq y_i \\ -1 & x_i < y_i \end{cases}$$

We introduce two variables for the proof:

$$\alpha_i = p_i x_i - p_i y_i \quad \beta = \sum_{i=1}^k (p_i x_i - p_i y_i) / k$$

We start with the following inequality:

$$\sum_{i=1}^k (\alpha_i - \beta)^2 = \sum_{i=1}^k \alpha_i^2 - 2\beta \sum_{i=1}^k \alpha_i + k\beta^2 \geq 0$$

Since $\sum_{i=1}^k \alpha_i = k\beta$, the above inequality is reduced to $\sum_{i=1}^k \alpha_i^2 \geq k\beta^2$. Substituting the following two equations into this inequality gives

$$\sum_{i=1}^k \alpha_i^2 = \sum_{i=1}^k p_i^2 (x_i - y_i)^2 = \sum_{i=1}^k (x_i - y_i)^2 = d^2(\mathbf{x}, \mathbf{y})$$

$$k\beta^2 = k \left(\sum_{i=1}^k (p_i x_i - p_i y_i) / k \right)^2 = \left(\sum_{i=1}^k (p_i x_i - p_i y_i) \right)^2 / k$$

and we now have the inequality we want to prove. \square

With lemma 1 available, we now describe the proposed algorithm. The algorithm consists of two steps. The first is a checking step which determines whether the distance calculation is required or not and the second step is the distance calculation step.

In the first step, for a given vector \mathbf{x} , the algorithm checks if

$$\left(\sum_{i=1}^k |x_i - y_i| \right) / k \geq d_{min}^2$$

where d_{min} is the minimum distance so far. If the answer is yes, the codeword is rejected without calculating the Euclidean distance. Otherwise, the distance is calculated.

We now investigate the number of operations required for the proposed algorithm in detail. We assume that the size of the codebook is N and the number of distance calculations needed is M . The full search algorithm requires $N(2K - 1)$ additions (including subtractions) and NK multiplications. The proposed algorithm requires $2N + MK$ multiplications, $N(2K - 1) + M(K - 1)$ additions, and NK comparisons. When we compare the two algorithms, the proposed algorithm saves $N(K - 2) - MK$ multiplications and loses $M(K - 1)$ additions and NK comparisons.

Consequently, the proposed algorithm reduces the number of multiplications but increases the number of additions and comparisons. Since multiplications require much more time to execute than additions and comparisons, the proposed algorithm reduces the overall encoding time.

It should be noted that the algorithm can be effectively combined with existing fast algorithms. When the proposed algorithm is combined with a fast algorithm such as the ENNS algorithm [2, 3], the required number of distance calculations M becomes very small. As a consequence, the encoding time of the proposed algorithms would be reduced further.

Experimental results: Experiments have been performed using four USC images (Lena, Boat, Man, Baboon) to evaluate the efficiency of the proposed algorithm. The images are 512×512 monochrome with 256 grey levels. The 'Lena' image was used as the training set to generate the codebook with dimension $16(4 \times 4)$.

In Table 1, the average number of distance calculations is compared and shown for various codebook sizes. It is apparent that the average number of distance calculations of the proposed algorithm is only a small portion of the full search algorithm. Even when compared to the ENNS algorithm, the proposed algorithm requires the smallest distance calculations.

As an example, when the codebook size is 256 and Lena is the input image, the average number of distance calculations is 5.76. In this case, the proposed algorithm saves $3491.84 (= 256 \times (15 - 2) - 5.76 \times 16)$ multiplications and loses $86.4 (= 5.76 \times (16 - 1))$ additions and $4096 (= 256 \times 16)$ comparisons. In addition, since the execution time of multiplications and additions depends on the type of the processor, it should be noted that exact savings of encoding time would be different according to the processor type.