Finite State Vector Quantization With Multi-Path Tree Search Strategy for Image/Video Coding*

Shih-Chou Juan Yen-Jean Chao Chen-Yi Lee
Dept. of Electronics Engineering, National Chiao Tung University,
1001, University Road, Hsinchu 300, Taiwan, ROC
Tel: 886-35-712121 ext.3286; Fax: 886-35-724361; Email: cylee@cc.nctu.edu.tw

ABSTRACT

This paper presents a new vector quantization (VQ) algorithm exploiting the features of tree-search as well as finite state VQs for image/video coding. In the tree-search VQ, multiple candidates are identified for on-going search to optimally determine an index of the minimum distortion. In addition, the desired codebook has been reorganized hierarchically to meet the concept of multi-path search of neighboring trees so that picture quality can be improved by 4 dB on the average. In the finite state VQ, adaptation to the state codebooks is added to enhance the hit-ratio of the index produced by the tree-search VQ and hence to further reduce compressed bits. An identifier code is then included to indicate to which output indices belong. Our proposed algorithm not only reaches a higher compression ratio but also achieves better quality compared to conventional finite-state and tree-search VQs.

1. INTRODUCTION

Vector quantization or VQ is a technique for data compression. The key concept inherent in this technique is to replace input vector by an index whose codevector has the minimum distortion compared to other codevectors of a designed codebook. To find such an index, several search schemes have been proposed in the literature [1], such as full search, tree-search, ... etc. In general, these schemes are evaluated *in terms of computation complexity, compression ratio, and picture quality.* Some minor modifications for storage space reduction on these schemes can also be found in the literature. For example, in [2,3], the authors exploit transforms and then select key components for comparison. Thus both computational complexity and storage space can be reduced. However, the picture quality becomes degraded due to the selection of fixed transform components. To provide a high-quality picture service, a large codebook is often demanded. However, the drawback lies in the low compression ratio which can be defined by $\frac{W\times N}{\log_2 M}$, where M, N, and W stand for codebook size, input vector size, and input word length respectively. It can be found that the compression ratio is very related to the codebook size M.

One way to improve the compression ratio while maintain high picture quality is to exploit finite state machines for state tracing. These finite state vector quantization or FSVQ methods have been found to be an efficient technique for image compression [4,5,6]. Figure 1 shows the block diagram of typical FSVQs. Both encoder and decoder of an FSVQ have a finite state machine, which uses previously encoded vectors to decide current state and then selects one corresponding state codebook, which is a subset of master codebook, to quantize input vector. Since the state codebook is smaller than the master codebook, FSVQ can achieve both higher compression ratio and lower computational complexity than the full search VQ. And if the finite state machine can be designed well, the quality of this coding method will be better than that obtained by full search VQ. For example, if the state codebooks contain 16 codevectors and the master codebook has 512 codevectors, then the computational complexity and the bit rate of FSVQ are 1/32 and 4/9 respectively of those obtained by full search VQ.

However, the state machine of FSVQ often does not produce the correct current state. In other words, the selected state codebook often does not contain the closest codevector and the encoder can only find suboptimal codevectors to encode the input vector. This causes larger distortion during encoding process and since next state is selected according to the previously decoded vector, the wrong selection of current state will influence the selection of following states. Therefore error becomes propagated and hence conventional FSVQs only get 1~2dB better coding quality than that obtained by the full search VQ at the same bit rate. In addition, since the current state codebook highly relies on previously decoded vectors, we cannot start to quantize the current input vector and select the state codebook from the state machines until the previous input vectors have been vector quantized. This feed-back structure or iterative bound of FSVQ makes it difficult to develop real-time cost-effective hardwares for practical applications.

To solve the above-mentioned problems, we propose a multi-path tree search FSVQ algorithm and its VLSI architecture. In this novel approach, instead of waiting for the quantization of previous vectors and then selecting current state codebook to quantize input vector, we first use Tree Search Vector Quantization or TSVQ method to find the indices, I_{TSVQ}, of the two closest codevectors from the codebook, and then check whether the state codebook contains any of these two codevectors. If yes, the index, I_{state}, corresponding to one of the two closest codevectors in the state codebook will be transmitted, otherwise an identifier ID together with the index I_{TSVQ} of the closest codevector in the master codebook will be sent out. This method overcomes the problem that the state codebook, in some cases, does not contain the proper codevector to represent the input vector. In section 2 we will describe this algorithm and discuss both FSVQ and TSVQ schemes in detail. Section 3 gives some simulation results to highlight distinguished features of this algorithm. Also some design issues

*Work supported by the National Science Council of Taiwan, ROC, under Grant NSC-83-0404-E-009-058.

of our approach and comparisons with current VQ approaches for image and video coding will be provided.

2. THE MULTI-PATH TREE SEARCH FSVQ ALGORITHM

The functional diagram of the proposed algorithm is shown in Figure 2. It contains two VQ phases: one is the TSVQ, and the other is the FSVQ. In the first phase of encoder design, we first exploit a multi-path tree search VQ to find the nearly closest codevector. Since the computational complexity of TSVQ is almost as low as that of FSVQ, the advantage of lower computational complexity inherent in FSVQ is still reserved in this proposed algorithm. In the second phase, we then exploit a finite state machine to predict the possible index produced by the TSVQ. That is, the state machine selects a state codebook which may or may not contain the index I_{TSVQ} produced by the TSVQ. If the prediction is correct and the selected codebook really contains the I_{TSVQ}, the information needed to be transmitted to the decoder is the position I_{state} which specifies the position of I_{TSVQ} in the selected state codebook; otherwise an identifier code ID together with the I_{TSVQ} are transmitted to the decoder. This solves the problem that sometimes the state codebook does not contain the proper codevector to represent the input.

In the following, we will show how the TSVQ is designed to achieve better quality of decoded images and how the finite state machine is designed to reach lower bit rate.

2.1. Design of Multi-Path TSVQ

The first part of the proposed VQ algorithm is a TSVQ which is exploited to find the nearly closest codevector to represent the input vector. The tree search VQ (TSVQ) or known as tree-structure VQ [7] is an alternative to full search VQ. In principle, the TSVQ performs a sequence of binary (or larger) search instead of one large search as in conventional full search VQs. As a result, *the encoding complexity increases as* $\log_2 N$ *instead of as N, where N is the number of leaves of the searching tree.* The performance of TSVQ will in general suffer some degradation over the performance of full search VQ at the same number of codewords. In addition, the memory requirement in the decoder of TSVQ becomes nearly doubled. However, the great advantage of reducing search complexity over some increase in distortion makes the TSVQ more attractive. The available TSVQ's can be categorized as balanced and unbalanced TSVQs.

Although the performance of unbalanced TSVQ is in general better than the balanced TSVQ, the uncertain searching depth makes it difficult for hardware implementation, which also degrades the advantage of lower computational complexity inherent in TSVQ. In addition, our experiment results show that sometimes the performance of the unbalanced TSVQ is not good enough, especially for images outside the training sequences. The reason of this phenomenon is that unbalanced TSVQ uses more codevectors to encode commonly used clusters while training the codebook, and for the less used clusters, fewer codevectors are used to represent them. Therefore for the outside training images, if most input vectors belong to the less used clusters, the performance of this encoder becomes very bad. Due to this consideration, the balanced TSVQ is exploited here in the first phase to reach a reasonable computational complexity and make the algorithm more suitable for general images.

2.2. Strategies to Improve TSVQ

Sometimes, the TSVQ cannot find the closest codevector because the "father" of the closest codevector is not always closer than the other nodes in the "ancestor" level. This phenomenon becomes more serious when the tree size is larger and the codevectors become much closer to each other. Then the "ancestors" of the closest codevector are not selected as the new search node, because sometimes the brothers of these "ancestors" are much closer to the input vector. If one of the "ancestors" is not selected, the obtained codevector will never be the closest one and then cause large distortion. This is why the performance of TSVQ is always lower than full search VQ.

To reduce the effect of this problem, we use the following Strategies:

Multi-Path TSVQ F. Chang et. al. [8] have proposed a multi-path TSVQ to improve the performance of TSVQ. Instead of finding one nearest node from two searching nodes, the multi-path TSVQ finds two nearest nodes from the four searching nodes. The next four search nodes are the children of the two found nearest nodes. The computation of the multi-path TSVQ is the double of that of conventional TSVQs. The reason that multi-path TSVQ searches more nodes than the TSVQ is because sometimes the "ancestor" of the closest codevector is not closer than other search nodes, then the tree search will cause some errors. Therefore, the multi-path TSVQ searches more nodes to reduce the effect. Table 1 shows the improvement of the multi-path TSVQ. This experiment uses a codebook of 256 codevectors and 4 test images.

Neighbors Searching After obtaining the nearly closest codevector by the multi-path TSVQ, we can search the neighbors of the obtained codevector. Since the codevector is very close to the input vector, the closest codevector must be very close to the obtained codevector. In other words, the closest codeword may be in the neighbors of the obtained codevector.

Table 2 shows the results of applying this method which achieves 0.2~0.3 dB higher compared to those without searching the neighbors.

Codebook Design The codebook of traditional balanced TSVQ is designed with the splitting method of the generalized Lloyd algorithm (GLA) [7]. But the codebook generated by this method can not ensure that the "ancestor" of the closest codevector will be closer to the input vector than its "brothers". To achieve better coding quality, the codebook used in the TSVQ has to be modified. In this paper, we propose the following scheme to construct the tree structure codebook.

1. Use the LBG algorithm [9] to produce the codebook until the codebook size reaches the desired figure N. These N codevectors are the leaves of the tree.
2. For the obtained N-size codebook, we first identify the pairs whose leaves are closest to each other. This location process begins at looking for the furthest pair (i.e. pair distance is the longest). The codevectors in the same pair become the "brothers" in the searching tree. And the "father" is constructed by averaging the two codevectors in the pair. Then we obtain the level of $\frac{N}{2}$-node of the searching tree. This process will continue until the root level is reached.

We call this method as "Average Tree Construction", because the "father" is always the average of the two children. Experiment results (Given in Table 3) show that codebook constructed by this method will achieve higher quality (0.2~0.3 dB in PSNR) compared to codebook generated by LGB. This improvement results from the fact that the father is the "center" of the two children, therefore the "ancestor" of the closest codeword will be closer to the input vector than the conventional TSVQ whose codebook is generated by the GLA.

2.3. Design of Finite State VQ

The second part of the proposed algorithm is a state machine which uses the information of previously decoded vectors to predict the possible index I_{TSVQ} generated by the TSVQ. That is, the state machine will select a proper state codebook which may contain the newly generated index I_{TSVQ}. *In other words, the elements of the state codebook are the possible indices of I_{TSVQ}, rather than the entries of the codevectors used in conventional FSVQs. This strategy leads to save large memory space and to reduce computational complexity.* If the selected state codebook really contains the new index I_{TSVQ}, an index I_{state} which indicates the position of the I_{TSVQ} in the selected state codebook is transmitted to decoder. Since the selected state codebook of the decoder is the same as that of the encoder, the decoder can use this I_{state} to recover the corresponding I_{TSVQ} and then reconstruct images from the closest codevectors. Because the size of the state codebook is smaller than the larger codebook of the TSVQ, fewer bits for I_{state} instead of the larger size I_{TSVQ} are needed for transmission. Simulation results show that the size of this state codebook lies in the range of [4..16]. However, if the selected state codebook does not contain the newly generated index I_{TSVQ}, an identifier code ID together with the I_{TSVQ} will be transmitted to the decoder. With this ID code, the decoder knows how to assemble the decoded images by using either the I_{TSVQ} or the I_{state}.

In the meantime, we can adaptively change the content of selected state codebooks to reflect the local statistics of input image/video sequences. This is done by inserting the recently used index on the top of state codebooks. This results in an adaptive version of multi path tree search based FSVQ algorithm whose output is a variable rate. If the state machine is designed well, and selected state codebooks usually hit the I_{TSVQ}, then the average bit rate becomes lower than that obtained from FSVQ without adaptation.

3. EXPERIMENT RESULTS

We have applied this algorithm to implement the Side Match VQ (SMVQ) [5], which is one of FSVQs, to observe the coding quality. Simulation results are obtained from two different approaches: one is the multi path tree search FSVQ or called the Modified SMVQ, and the other is the adaptive version which is called the Adaptive Modified SMVQ. The adaptive modified SMVQ adaptively updates the state codebook by exploiting least recently used strategy (LRU). This LRU is implemented by placing the most recently used element on the top of selected state codebooks. Here, the codebook of the TSVQ is generated by the "Average Tree Construction" technique mentioned earlier, where a 4-path TSVQ is used. We have changed both sizes of the codebook of the TSVQ and the state codebook of the state machine. *The larger the codebook size of TSVQ is, the higher PSNR coding results will be obtained. For a given TSVQ codebook size, there is an optimal state codebook size to reach a minimum bit rate.* The more complex the encoded image is, the larger the optimal state codebook size is. Figure 3 shows the comparison of the PSNR vs bit-rate of this proposed algorithm with that obtained from the original SMVQ. Note that each point is derived at an optimal state codebook size for different TSVQ codebook sizes. Simulation results show that the Modified SMVQ implemented by our proposed approach does achieve higher PSNR than that obtained from the original SMVQ. The reason why this modified SMVQ improves the coding quality is because our proposed algorithm eliminates the disadvantage that the nearest codevector is not in the state codebook as found in conventional FSVQs. And the Adaptive Modified SMVQ outperforms the others is because state codebook of the Adaptive Modified SMVQ can trace the local statistics of input image/video sequences so that the prediction of the I_{TSVQ} is more accurate and the average output bit rate becomes lower.

Compared to other FSVQ algorithms, our approach offers higher picture quality at the same bit-rate. In the meantime, the iteration bound is now removed in our algorithm because only one input data has to be compared instead of an input vector. Although the TSVQ is added and hence computational requirements become higher, we can exploit the pipelining inherent in the TSVQ to improve speed. Thus our approach does outperform available VQ solutions in terms of picture quality, bit-rate, and hardware complexity.

4. CONCLUSION

In the proposed algorithm, we have combined the advantages of both tree-search and finite-state VQs. Simulation results have shown that good picture quality can be achieved at lower output bit-rate. Currently VLSI chips for both encoder and decoder are under development in order to set up a demonstrator VQ-based system in the near future.

REFERENCES

[1] R.M. Gray and A. Gersho, *"Vector Quantization and Signal Processing"*, Kluwer Academic Publishers, 1991.

[2] C.Y. Lee and S.C. Juan, *"An ASIC Architecture for Real-Time Image/Video Coding Based on Fixed-Basis Distortion Vector Quantization"*, Proc. of ISCAS'92, May 10-13, 1992, San Diego, California.

[3] C.Y. Lee and S.J. Juan, *"VLSI Implementation of a Modified-VQ Encoder Suitable for Image/Video Applications"*, Proc. of EURSIPCO'92, Aug. 26-28, Brussels, Belgium.

[4] J. Foster, R.M. Gray, and M.O. Dunham, *"Finite State Vector Quanitization for Waveform Coding"*, IEEE Trans. on Info. Theory, Vol. IT-31 pp. 348-355, May 1985.

[5] T. Kim, *"Side Match and Overlap Match Vector Quantizers for Image"*, IEEE Trans. on Image Processing, Vol. 1, No. 2, April 1992.

[6] W.T. Chen, R.F. Chang, and J.S. Wang, *"Image Sequence Coding Using Adaptive Finite-State Vector Quantization"*, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 2, No. 1, March 1992.

[7] A.Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, *"speed coding based upon vector quantization"*, IEEE Trans. Acoust. Spec. Signal Processing, vol. 28, pp. 562-574, Oct 1980.

[8] F. Chang, W.T. Chen, and J.S. Wang, *"Image Sequence Coding Using Adaptive Tree-structure Vector Quantization with Multipath Searching"*, IEEE Int. Conf. Acoust, Speech, and Signal Proc., 1991 pp.2281-2284.

[9] Y.Y. Linde, A. Buzo, and R.M. Gray, *"An Algorithm for Vector Quantizer Design"*, IEEE Trans. on Commun., Vol. COM-28, No. 1, pp. 84-95, 1980.

Table 1. The improvement of multi path TSVQ vs conventional TSVQs and full-search VQ.

schemes	*test images*			
	lena	pepper	girl	jet
TSVQ	25.837	26.732	24.732	25.428
multi-path TSVQ	29.09	29.619	31.97	29.505
full-search VQ	29.718	30.296	32.79	30.616

Table 2. The experiment results of searching the neighbors of the closest codevector.

schemes	*test images*			
multi-path TSVQ	lena	pepper	girl	jet
with 8 neigh.	29.311	29.844	31.82	29.929
with 4 neigh.	29.232	29.706	31.611	29.776
without neigh.	29.09	29.619	31.57	29.505

Table 3. The improvement of multi-path TSVQ with the "Average Tree Construction" method.

schemes	*test images*		
	lena	pepper	jet
TSVQ with new codebook	29.267	29.852	29.794
TSVQ with LGB codebook	29.09	29.619	29.505
full-search VQ	29.718	30.296	30.616

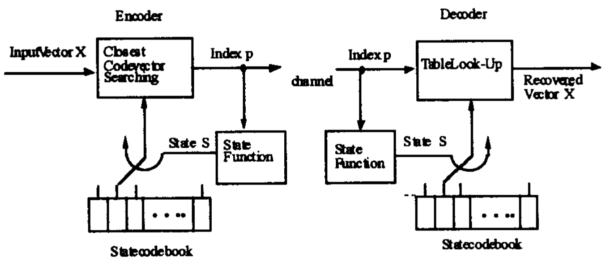

Fig.1. Functional diagram of a typical FSVQ.

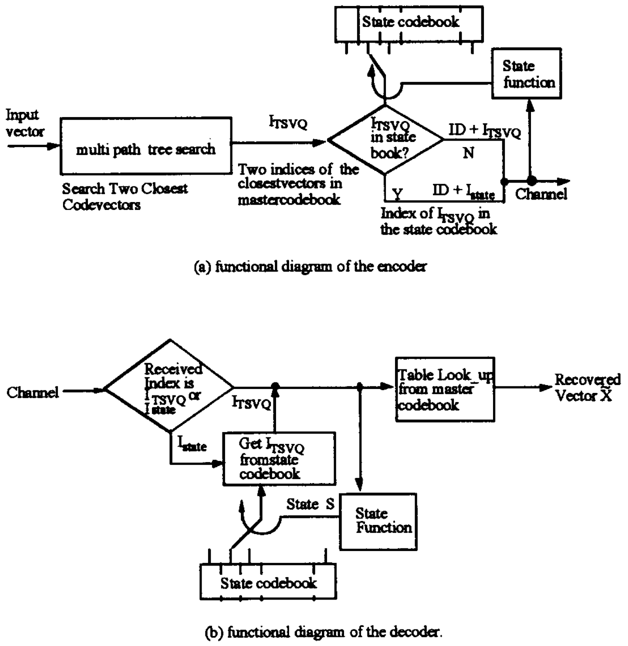
State codebook
State function
Input vector
multi path tree search
Search Two Closest Codevectors
I_{TSVQ}
Two indices of the closestvectors in mastercodebook
I_{TSVQ} in state book?
ID + I_{TSVQ}
N
Y
ID + I_{state}
Channel
Index of I_{TSVQ} in the state codebook
(a) functional diagram of the encoder
Channel
Received Index is I_{TSVQ} or I_{state}
I_{TSVQ}
I_{state}
Get I_{TSVQ} fromstate codebook
Table Look_up from master codebook
Recovered Vector X
State S
State Function
State codebook
(b) functional diagram of the decoder.

Fig.2. Functional diagram of the proposed FSVQ system. (a) encoder and (b) decoder.

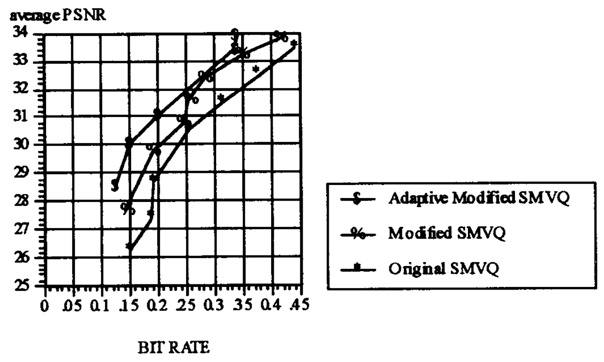

Fig.3. PSNR vs. different bit rates. Simulation results on Clair sequence are obtained respectively from original SMVQ and our modified SMVQ implemented by the proposed architecture. The adaptive one modifies state codebook by using LRU strategy. The master codebook is generated by using LBG algorithm from the luminance of four JPEG test images: "balloons", "Barbara", "boats", and "chairlady".