Finite State Hierarchical Table-Lookup Vector Quantization for Images

Navin Chaddha, Sanjeev Mehrotra and R. M. Gray
Information Systems Laboratory,
Stanford University, Stanford, CA 94305

Abstract

This paper presents an algorithm for image compression using finite state hierarchical table-lookup vector quantization. Finite state vector quantizers are vector quantizers with memory. Finite state vector quantization (FSVQ) takes advantage of the correlation between adjacent blocks of pixels in an image and also helps in overcoming the complexity problem of block memoryless VQ for large block sizes by using smaller block sizes for similar performance. FSVQ algorithms typically try to preserve edge and gray scale gradient continuity across block boundaries in images in order to reduce blockiness.

Our algorithm combines FSVQ with hierarchical table-lookup vector quantization. Thus the full-search encoder in an FSVQ is replaced by a table-lookup encoder. In these table lookup encoders, input vectors to the encoder are used directly as addresses in code tables to choose the codewords. In order to preserve manageable table sizes for large dimension VQ's, we use hierarchical structures to quantize the vector successively in stages. Since both the encoder and decoder are implemented by table lookups, there are no arithmetic computations required in the final system implementation. To further improve the subjective quality of compressed images we use block transform based finite-state table-lookup vector quantizers with subjective distortion measures. There is no need to perform the forward or reverse transforms as they are implemented in the tables.

1. Introduction

Full-search vector quantization (VQ) [1] is computationally asymmetric in that the decoder can be implemented as a simple table lookup, while the encoder must usually be implemented as an exhaustive search for the minimum distortion codeword. VQ therefore finds application to problems where the decoder must be extremely simple, but the encoder may be relatively complex, e.g., software decoding of video from a CDROM.

Various structured vector quantizers have been introduced to reduce the complexity of a full-search encoder [1]. One such scheme is hierarchical table-lookup VQ (HVQ) [2], [3]. HVQ is actually a table-lookup vector quantizer which replaces the full-search vector quantizer encoder with a hierarchical arrangement of table lookups, resulting in a maximum of one table-lookup per pixel to encode. Recent work on table-lookup vector quantization has been on combining it with block transforms with subjective distortion measures [3], wavelet transforms [4] and applying it to low complexity scalable video coding [5]. In this paper we propose a finite state table-lookup vector quantizer which has low complexity at the encoder and decoder and uses variable rate coding with subjective distortion measures to improve the perceived quality of compressed images.

Another problem with memoryless VQ is that the encoder and the codebook design complexity increases rapidly with vector dimension. Thus the use of VQ is restricted to small vector dimensions, typically 4x4 blocks for images. There has been much work in the past on VQ with memory such as FSVQ [6, 7, 8] to overcome the disadvantages of small block sizes. The aim of FSVQ is to achieve the performance of a memoryless VQ with a large codebook while using a much smaller codebook.

An FSVQ can be thought of as a collection of codebooks, each corresponding to a state of the encoder and decoder. The union of all the state codebooks is called a super-codebook. The current state of the system is determined by the previous state and the index of the selected codeword in that state. In the absence of channel errors, the decoder can track the encoder state given the initial state and the codeword index sequence without requiring any side information. FSVQ's for images [7], [8] use the correlation between the previous adjacent blocks and the current block to restrict the reproductions of the current block to a small state codebook. There has been work in the past on fixed-rate FSVQ [7] and variable-rate FSVQ[8] for images.

We propose a finite state hierarchical table-lookup VQ (FSHVQ) for images in this paper. Our algorithm replaces the full-search VQ encoder used in the previous FSVQ systems [7], [8] with a hierarchical table-lookup vector encoder. Thus FSHVQ gains the advantages of FSVQ while maintaining the computational simplicity of table lookup encoding and decoding. There are no arithmetic computations required in the final system implementation. Another advantage of using hierarchical table-lookup VQ is that complexity of the encoder does not depend on the complexity of the distortion measure, since the distortion measure is pre-computed into the tables. Hence it is ideally suited to implementing perceptually meaningful, if complex, distortion measures, which we here do.

This paper is organized as follows. Section 2 describes hierarchical table-lookup vector quantization. Section 3 describes the finite state hierarchical table-lookup vector quantization algorithm. Section 4 gives the simulation results and we conclude in Section 5.

2. Hierarchical Table-Lookup VQ

Hierarchical table-lookup vector quantization (HVQ) [2], [3] is a method of encoding vectors using only table lookups. It was used for speech coding in [2] and extended for image coding in [3], [4].

By performing the table lookups in a hierarchy, larger vectors can be accommodated in a practical way, as shown in Figure 1. In the figure, a $K = 8$ dimensional vector at original precision $r_0 = 8$ bits per symbol is encoded into $r_M = 8$ bits per vector (i.e., at rate $R = r_M/K = 1$ bit per symbol for a compression ratio of 8:1) using $M = 3$ stages of table lookups. In the first stage, the K input symbols are partitioned into blocks of size $k_0 = 2$, and each of these blocks is used to directly address a lookup table with $k_0 r_0 = 16$ address bits to produce $r_1 = 8$ output bits. Likewise, in each successive stage m from 1 to M, the r_{m-1}-bit outputs from the previous stage are combined into blocks of length k_m to directly address a lookup table with $k_m r_{m-1}$ address bits to produce r_m output bits per block. The r_m bits output from the final stage M may be sent directly through the channel to the decoder, if the quantizer is a fixed-rate quantizer, or the bits may be used to index a table of variable-length codes, for example, if the quantizer is a variable-rate quantizer.

Clearly many possible values for k_m and r_m are possible, but $k_m = 2$ and $r_m = 8$ are usually most convenient for the purposes of implementation. For simplicity of notation, we shall assume these values in the remainder of the paper. The sizes of the tables at different stages of the HVQ can be changed to provide a tradeoff [5] between memory size and PSNR performance.

The table at stage m may be regarded as a mapping from two input indices i_1^{m-1} and i_2^{m-1}, each in $\{0,1,.....,255\}$, to an output index i^m also in $\{0,1,.....,255\}$. That is, $i^m = i^m(i_1^{m-1}, i_2^{m-1})$. With respect to a distortion measure $d_m(x, \hat{x})$ between vectors of dimension $K_m = 2^m$, design a fixed-rate VQ codebook $\beta_m(i)$, $i = 0,1,...,255$ with dimension $K_m = 2^m$ and rate r_m, trained on the original data using any convenient VQ design algorithm [1]. Then set:

$$i^m = argmin_i d_m((\beta_{m-1}(i_1^{m-1}), \beta_{m-1}(i_2^{m-1})), \beta_m(i))$$

to be the index of the 2^m-dimensional codeword $\beta_m(i)$ closest to the 2^m-dimensional vector constructed by concatenating the 2^{m-1}-dimensional codewords $\beta_{m-1}(i_1^{m-1})$ and $\beta_{m-1}(i_2^{m-1})$. The intuition behind this construction is that if $\beta_{m-1}(i_1^{m-1})$ is a good representative of the first half of the 2^m-dimensional input vector, and $\beta_{m-1}(i_2^{m-1})$ is a good representative of the second half, then $\beta_m(i^m)$, with i^m defined above, will be a good representative of both halves, in the codebook $\beta_m(i)$, i=0,1,....,255.

It was shown in [3] how to incorporate perceptually significant distortion measures into HVQ based on weighting the coefficients of arbitrary block transforms (WTHVQ). Essentially, the transforms are pre-computed and built into the encoder and decoder lookup tables. Thus WTHVQ gains the perceptual advantages of transform coding while maintaining the computational simplicity of table lookup encoding and decoding. Figure 2 shows the building of a second stage table for WTHVQ from the corresponding codebooks.

3. Finite State Hierarchical Table-Lookup VQ

Finite-state hierarchical table-lookup vector quantization (FSHVQ) is a form of VQ with memory and consists of two finite state machines one at the encoder and the other at the decoder. The main difference between FSHVQ and ordinary FSVQ is that the full-search encoder is replaced by a table-lookup HVQ in FSHVQ.

3.1. Problem Formulation

The image to be encoded is divided into rectangular blocks with k-pixels and is scanned left to right and top to bottom. The blocks are also scanned in the same order as the image.

Let x_n denote the vector to be quantized at time instant n, y_n the corresponding transmitted channel symbol and $\hat{x}_n$ the reproduced vector at the decoder. Let S be the finite state space whose members s are called states, s_n be the state at time instant n, C_s be the state codebook for state s, α be the FSHVQ encoder, β be the FSHVQ decoder and f be the next state function.

3.2. Distortion Measure

We define the distortion between the input x and the reproduction $\hat{x}$ as: $d(x, \hat{x}) = (Tx - T\hat{x})^t W_x (Tx - T\hat{x})$, where T is the transformation matrix of some fixed transform, and the weights W_x vary arbitrarily with x. This is a reasonably general class of perceptual distortion measures. The perceptual weighting W_x varies over the image, but because it is embodied in the distortion measure of the vector quantizer, no side information is required to describe the weighting to the decoder. With W_x and T being the identity matrix (I) we get the squared error distortion measure. This distortion is used in building the tables of Section 2 and also in the design of state codebooks.

3.3. FSHVQ Encoder

Thus for each state-codebook, C_s, a corresponding table-lookup vector quantizer encoder H_s is designed using the algorithm in Section 2. There are two types of finite state table-lookup HVQ encoders, one (H_s^d) in which all stage tables of the HVQ are different for each state and the other (H_s^c) in which only the last stage tables of HVQ are different for each state. The latter has the advantage of requiring less memory.

Thus at time instant n (see Figure 3), the encoder α performs table-lookup encoding on the current vector x_n using H_{sn} for state s_n and outputs a channel symbol y_n which is transmitted to the decoder without any other information about the state. Next the encoder calculates the new state using the next state function.

3.4. Next State Function

The next state function takes the past reproduction vectors as input and gives the next state as output. This operation is performed as a table-lookup. It is summarized by:

$s_n = f(y_n, y_{n-1}, \ldots)$.

3.5. FSHVQ Decoder

The FSHVQ decoder (Figure 4) is the same as an FSVQ decoder. It calculates the next state by applying the next state function to its previous state and the received codeword index. In the case of WTHVQ, the decoder codebooks have the inverse transform performed ahead of time.

3.6. Choice of States

For images we use the block to the left and at the top of the current block to define the state. Each neighboring block is classified using a classifier designed for it. The classifier (see Figure 5) is the side-match classifier of [8]. The idea is to look only at one row from the top block and one column from the left block for classification purposes. This provides continuity of edges and gray levels across blocks.

3.7. Variable Rate Coding

In order to further reduce the rate of the system, without increase in distortion, we use entropy-constrained vector quantization (ECVQ) [9] in the design of the state codebooks. Thus each state codeword has a conditional entropy code matched to its probability. These variable rate indices are built into the last stage HVQ tables (H_s). Thus the last stage HVQ table outputs a variable length code.

4. Simulation Results

In this section we give the simulation results. We give the PSNR (peak SNR) results on the 8-bit monochrome image Lena (512x512). The different state codebooks for these simulations were generated by training on 30 different images. First codebooks for the side-match classifier were designed using GLA [1]. Next the training set was partitioned into different subsets corresponding to the different states. Then state codebooks for each state were designed using ECVQ for different lambda values. The state HVQ tables were built from the state codebooks. There was very slight difference in PSNR between (H_s^d) and (H_s^c). Thus we give only results for the latter as it requires smaller memory.

Table 1 gives the PSNR at different rates for 16-state FSHVQ with a 4x4 block size on Lena. For example, a compression of 22:1 and with 16 states for finite state case, the PSNR for VQ, HVQ, FSVQ, FSHVQ are 28.2 dB, 27.5 dB, 31.2 dB and 30.5 dB respectively. These results for finite-state are with entropy coding and no perceptual weighting. Thus FSHVQ gains 3 dB over ordinary HVQ and 2.3 dB over VQ. The encoding time for Lena on a Sparc-10 workstation for FSVQ was of the order of 25-50s while for FSHVQ was 50-100ms. Thus the table-lookup encoding is approximately three orders of magnitude faster than the full-search encoding. The advantage of using perceptual weighting in FSHVQ is to increase the subjective quality of the compressed images. We have found that the subjective quality of the images compressed using perceptually weighted FSHVQ with generic block transforms is significantly better than the unweighted FSHVQ at all compression ratios. Perceptual weighting provides a 2 dB equivalent gain in PSNR over unweighted VQ's. Perceptually weighted FSHVQ gives better compressed image quality compared to JPEG but at a lower computational cost.

5. Conclusions

In this paper we have presented a finite state table-lookup HVQ. The encoder is implemented by only table lookups and thus reduces the complexity compared to a FSVQ encoder. We have incorporated perceptually significant distortion measures into FSHVQ based on weighting the coefficients of arbitrary transforms. Essentially, the transforms are pre-computed and built into the encoder and decoder lookup tables. Thus we gain the perceptual advantages of transform coding and also advantages of finite state VQ while maintaining the computational simplicity of table lookup encoding and decoding.

References

[1] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer Academic Pub., 1991.

[2] P.-C. Chang, J. May and R.M. Gray, "Hierarchical Vector Quantization with Table-Lookup Encoders," *Proc. Int. Conf. on Communications*, Chicago, IL, June 1985, pp. 1452-55.

[3] N. Chaddha, M. Vishwanath and P. Chou, "Hierarchical Vector Quantization of Perceptually Weighted Block Transforms," *Proc. Data Compression Conference*, March 1995.

[4] M. Vishwanath and P.A. Chou, "An efficient algorithm for hierarchical compression of video," *Proc. Intl. Conf. Image Processing*, Austin, TX, Nov. 1994, Vol. 3, pp. 275-279.

[5] N. Chaddha and M. Vishwanath, "A Low Power Video Encoder with Power, Memory, Bandwidth and Quality Scalability," to appear in *VLSI-Design'96 Conference*, Jan. 1996.

[6] J. Foster, R. M. Gray and M. O. Dunham, "Finite state vector quantization for waveform coding," *IEEE Tran. Info. Theory*, vol. IT-31, pp. 348-355, May 1985.

[7] R. Arvind and A. Gersho, "Image compression based on vector quantization with memory," *Optical Engineering*, vol. 26, pp. 570-580, July 1987.

[8] T. Kim, "New Finite State Vector Quantizers for Images," *Proc. ICASSP'88*, vol. 2, pp. 1180-1183, April 1988.

[9] P. Chou, T. Lookabaugh and R. M. Gray, "Entropy Constrained Vector Quantization," *IEEE Tran. ASSP.*, vol. 37, pp. 31-42, Jan. 1989.

Table 1: PSNR results for FSHVQ (4x4 blocks)

Rate in bits per pixel	PSNR in dB
0.0	10.8
0.04	15.2
0.15	19.5
0.20	23.6
0.22	25.3
0.25	26.8
0.29	28.7
0.32	30.1
0.34	30.5
0.37	30.7

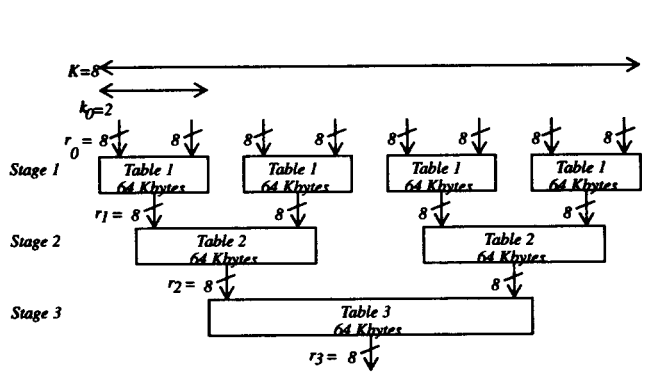

Figure 1. A 3-stage HVQ encoder (k_m=2, r_m=8)

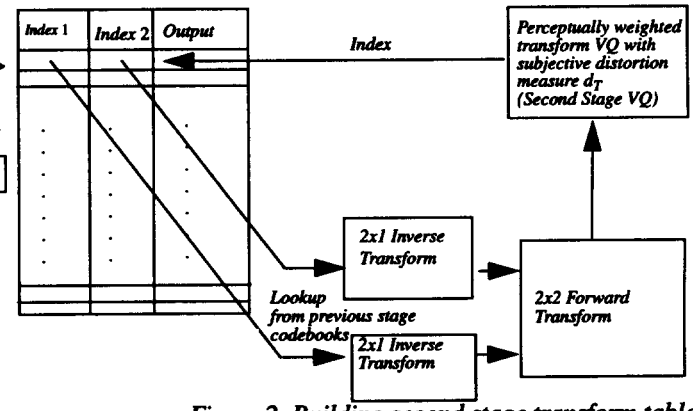

Figure 2. Building second stage transform table

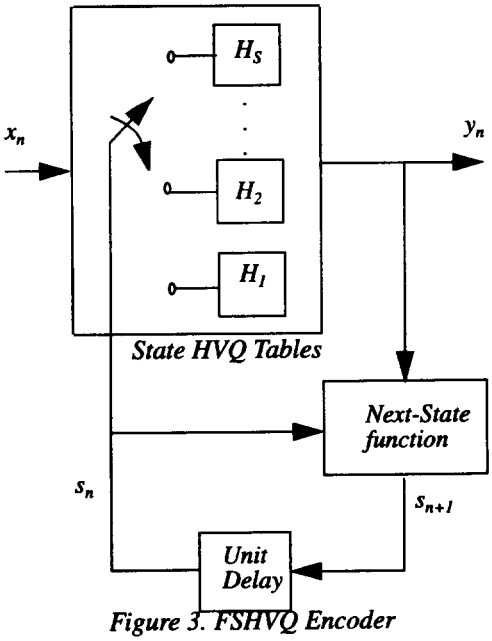

Figure 3. FSHVQ Encoder

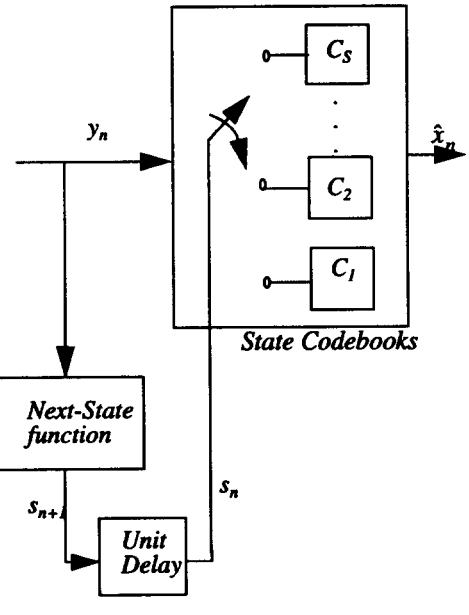

Figure 4. FSHVQ Decoder

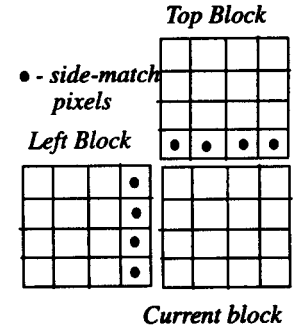

Figure 5. Choice of States