Predictive Hierarchical Table-Lookup Vector Quantization with Quadtree Encoding

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

Abstract

In this paper we present an algorithm for image compression which involves adaptively segmenting a block of residuals resulting from prediction, while encoding them using hierarchical table lookup vector quantization. An optimum decomposition of the block allows an image to be adaptively quantized depending on the statistics of the residual block being encoded. This is useful since most images are nonstationary and have some regions with high detail and some with little. With an optimum decomposition, we can adaptively allocate bits by varying the block size to be quantized. Predictive vector quantization (PVQ) allows us to take advantage of the correlation between adjacent blocks of pixels being encoded by providing memory. The quadtree structure is used to represent the segmentation information and is sent as side information. To reduce encoding complexity, we use hierarchical table lookups so no arithmetic computations have to be performed to find the minimum distortion codeword. To further improve performance, we use a variable rate code to decrease the rate. Also, to improve the subjective quality of the image, we use subjective distortion measures.

1. Introduction

One of the main problems with full search vector quantization (VQ) [1] is that the encoder and codebook design complexity grows linearly with the number of codewords in the VQ codebook, which grows exponentially with the bit rate and vector dimension. However, the decoder is simply a table-lookup. So VQ is often used only in places where a simple decoder is required, but a complex encoder can be used.

Structured vector quantizers have been implemented to reduce the encoder complexity [1]. One such method, hierarchical vector quantization (HVQ) [2],[3], uses a set of hierarchical table lookups to perform the encoding. Since the tables can be built in advance, the actual encoder doesn't have to perform any arithmetic computations to find the minimum distortion codeword. Hierarchical table lookups are used so that larger vector dimensions can be used with a reasonable amount of memory. Also, since the tables can be built in advance, it is easy to incorporate subjective distortion measures. They can be precomputed when the table is built. Recently, there has been a lot of work in combining HVQ with other techniques such as wavelet (subband) coding [4], VQ with memory [5],[6] and applying it to low complexity software-only-encoding of video and images [4],[7]. Also, there has been work on combing HVQ with block transforms with perceptually weighted distortion measures [3].

Since the VQ encoder complexity grows rapidly with vector dimension, VQ can only be used to encode images with either small block sizes or large block sizes with relatively small codebooks. However, small codebooks with large block sizes often result in too much compression and too much distortion. To overcome the affects of small block sizes, vector quantization with block memory such as predictive vector quantization (PVQ) [8] and finite state vector quantization (FSVQ) [9] can be used. These algorithms use the correlation between adjacent blocks of pixels to achieve the performance of VQ with a large codebook while using a much smaller codebook. Prediction allows one to use a smaller codebook since the residuals resulting from prediction have a much smaller variance than the actual pixel values. Also because the correlation between adjacent blocks is used for prediction, grayscale continuities across blocks are better preserved, thus reducing blockiness.

Another problem with VQ is the inability of the codebook to adapt to nonstationarities in the image. All blocks are quantized using the same codebook. A method that can be used to improve performance is to adaptively allocate bits to different spatial regions of the image [10],[12]. Segmentation can be done to identify regions of the image which have low detail and can thus be highly compressed, and regions of high detail, such as edges, which cannot be highly compressed. Recently, there has been work on how to do this segmentation optimally [11]. An algorithm presented by Baker and Sullivan [11] shows how a quadtree decomposition can be used to optimally decompose a block based upon its statistics. Basically, this algorithm divides blocks which have too much distortion when being encoded into four subblocks and encodes these subblocks. In this fashion, regions of high detail get extra bits.

We propose to use a quadtree decomposition to optimally segment a block of residuals resulting from prediction. Since the error residuals resulting from prediction have varying statistics across the image, adaptively allocating bits by using different quantizers on different regions of the image can result in improved performance over using a single quantizer to encode the residuals. The residuals often have large regions of constant intensity and small regions of fine detail which make them ideal for adaptively selecting a quantizer. The segmentation is used to adaptively select the size of a block to be encoded. All encodings are done using table lookups, so no arithmetic computations are performed for the encoder. Although the segmentation information is added side information, the reduction in rate resulting from adaptively segmenting the residual blocks more than makes up for this.

The residuals can be encoded in two ways. One way is use a single block size to find the residuals and then encode them using an optimal quadtree segmentation. However, this has the problem that the inner subblocks after segmentation will not be predicted well and thus the correlation between adjacent subblocks would not be exploited. To alleviate this problem, one can recursively predict the subblocks from the surrounding subblocks rather than using a single block size for the prediction.

Section 2 of the paper explains table lookup vector quantization. In section 3, we explain how the predictive hierarchical table lookup vector quantization with optimal segmentation algorithm works. In section 4, we present simulation results, and in section 5, the conclusion.

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 to 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 trade-off [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 $r_m/K_m = 8/2^m$ bits per symbol, trained on the original data using any convenient VQ design algorithm [1]. Then set

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

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.

3. Predictive HVQ with Quadtree Coding

Predictive table-lookup hierarchical vector quantization [5] (PHVQ) with quadtree encoding is a form of vector quantizer with memory and has the ability to adaptively select how to optimally segment a block of residuals to be encoded. However, the full search encoder in the algorithm is replaced with a table-lookup HVQ encoder. Also the optimum segmentation and prediction can be done with table lookups.

It consists of a predictor which uses the side pixels of quantized adjacent blocks to predict the current block to be encoded. The quantized blocks are used to predict since this is what the decoder will have. It then encodes the residual using the quadtree segmentation algorithm to optimally select the block sizes.

3.1. Problem Formulation

The image to be encoded is divided into m x m blocks, each with k pixels, and is scanned left to right and top to bottom. The blocks are scanned in the same order as the pixels. Let x_n be the vector to be quantized at time n, and let y_n be the corresponding transmitted channel symbol.

3.2. Predictive VQ

A predictive vector quantizer [8] (PVQ) is one which encodes the residual between the actual block and the predicted block instead of the actual block itself. Our PVQ uses the side pixels of the quantized adjacent blocks to predict the current block to be encoded.

Let s_n represent a $2m$ dimensional vector of the side pixels of previously quantized blocks adjacent to the block x_n as shown in Figure 2. Then the residual of the k-dimensional block is: $e_n = x_n - As_n$, where A is a k x $2m$ matrix of the optimum linear prediction coefficients obtained by solving the Weiner-Hopf equation. Then y_n is chosen to be the index of the codeword in the codebook which minimizes the distortion for the vector e_n.

The decoder receives y_n which can be used to reproduce a quantized version of the residual. It also has the quantized versions of the adjacent blocks, so it is able to predict the current block. To the predicted block, it simply adds the quantized version of the residual.

3.3. Optimal Quadtree Segmentation

We use the quadtree data structure (Figure 3) to represent the optimal segmentation of a particular block [11]. The quadtree consists of a tree whose root corresponds to the largest block size that can be encoded. Each node in the tree either has four children, meaning that the block is further subdivided into four subblocks, or no children, which means that the block is encoded [11]. The four subblocks are simply obtained by splitting an m x m block into four $m/2$ x $m/2$ blocks. We use the algorithm by Sullivan and Baker to determine the optimal quadtree structure for each block.

Let $2^{l_0} \times 2^{l_0}$ be the minimum possible block size that can be encoded, and let $2^L \times 2^L$ be the maximum block size which can be encoded. Then, the optimal quantization of the image is the optimum quantization of each of the $2^L \times 2^L$ blocks. To obtain the optimum quantization of a $2^l \times 2^l$ block, we encode the $2^l \times 2^l$ block using a quantizer designed for that block size. Then, we compare the performance of this quantization with the performance of the optimum quantization of each of the four $2^{l-1} \times 2^{l-1}$ blocks. Let D_l and R_l be the rate and distortion of encoding the $2^l \times 2^l$ block using a quantizer designed for that block size and let D_{l-1} and R_{l-1} be the sum of the rates and distortions of the optimum encodings of the four subblocks. Then, if $D_l + \lambda R_l \le D_{l-1} + \lambda R_{l-1}$, the optimal encoding of the $2^l \times 2^l$ block is the encoding using the $2^l \times 2^l$ block quantizer, else it is the optimum encoding of the four subblocks, where λ is the Lagrange multiplier. The optimum encoding of a $2^{l_0} \times 2^{l_0}$ block is the actual encoding of the block since there are no quantizers for smaller block sizes available. The entire process of finding the optimum quantization is repeated for all $l_0 + 1 \leq l \leq L$. The larger the value of λ is, the more emphasis is placed on rate over distortion and subsequently more of the blocks are encoded using larger block sizes. The rate used to calculate whether to merge or not includes one bit of segmentation information required at each node of the quadtree. This one bit tells whether to split that particular block into four subblocks or whether to encode the actual block. All levels of the quadtree require this side information except for the minimum block size which cannot be split any further.

3.4. PHVQ with Quadtree segmentation

There are two ways to combine PHVQ with quadtree segmentation. One way is to simply use prediction to first generate an $m \times m$ predicted block. Then the residual block between the actual block and predicted block can be encoded using a quadtree decomposition. However this doesn't take full advantage of the correlation between adjacent subblocks which are encoded. A better method is to use adjacent subblocks in the quadtree decomposition to find the residuals to encode. So, in this algorithm we first predict all the blocks using the smallest block size predictor. The residuals are encoded using HVQ. Then we use another predictor to generate residuals for the next block size. By using different predictors for each of the block sizes and using the adjacent pixels of the actual subblock being encoded, we take full advantage of the correlation between adjacent subblocks. If the decoder recursively decodes each of the four subblocks of a block which is split, then it is able to calculate the predicted blocks using the side pixels of the adjacent subblocks.

In our algorithm, all the full search encoders used in the quadtree decomposition are simply replaced with hierarchical table lookups. Therefore, no arithmetic computations are required to perform these encodings. Also, since hierarchical table lookups result in intermediate indices corresponding to a particular compression ratio, individual encoders do not need to be actually used for the various block sizes if one simply uses a quadtree decomposition on a block of residuals. The segmentation simply tells at which stage to stop the hierarchical table lookups. Even if one does the encoding using recursive predictions and encodings, the number of table lookups simply grows linearly with the number of levels in the quadtree. The prediction can also be done through the table lookups [5]. Also, the segmentation can easily be done by table lookups by storing the average distortions for each of the possible encodings at each stage of the table.

3.5. Variable Rate Coding

In order to further reduce the rate without increasing distortion, we use entropy constrained vector quantization [13] (ECVQ) to design the codebooks. This gives us a variable rate code since the transmitted symbols have a length matched to their probabilities. The variable rate indices can be built in to the last stage HVQ table.

4. Simulation Results

We give PSNR results here for the 8-bit per pixel monochrome image Lena (512x512). For the first method where a single predictor is used to get a block of residuals, we first generated a set of training vectors by obtaining residuals for a particular block size. The side pixels were used for generating the residuals. The residuals were recursively blocked to generate codebooks with various vector dimensions. To obtain quantizers with different bit rates, we generated codebooks with the same number of codewords but for different block sizes. For the recursive predictor and encoder, we generated residuals corresponding to each of the block sizes used in the quadtree decomposition and generated the codebooks for various vector dimensions. All the codebooks were generated using ECVQ algorithm and had 256 codewords.

Once the codebooks were generated, various values of λ were used to vary the compression in the encoding. For large values of λ, with a maximum block size in the quadtree decomposition being 8x8, most of the blocks were encoded as 8x8 blocks and thus a compression of 64:1 would have resulted if a fixed rate code were used (excluding the one bit of side information required for each block). For small values of λ, we would have a compression ratio of approximately 2:1 with a fixed rate code. In figure 4, we show PSNR curves for VQ, PVQ, VQ with quadtree, PVQ with quadtree on a block of residuals, and PVQ with quadtree with recursive prediction and encoding. For most rates, PVQ with quadtree on a block of residuals is approximately 1 dB better than PVQ and 1 dB better than VQ with quadtree alone. With PVQ with quadtree and recursive prediction and encoding, we gain about 2 dB more. With PHVQ with quadtree, we lose approximately 0.7 dB over PVQ with quadtree. The PHVQ with quadtree with recursive predicting and encoding is approximately 4-5 dB better than fixed rate HVQ. The compressed image using JPEG is shown in figure 5 and the recursive PVQ compressed image at the same bit rate (0.33 bpp) is shown in figure 6. As one can see, the JPEG image is much blockier and less perceptually pleasing. In figure 7, we show the segmentation map for the image in figure 6. This shows how the bits are adaptively allocated and how the background regions get less bits.

5. Conclusions

We have presented an algorithm for adaptively selecting a quantizer to encode a block of residuals resulting from prediction. Since prediction is used, memory can take advantage of the correlation between adjacent blocks of pixels to allow us to use a smaller codebook than would be required with memoryless VQ. Since an optimal segmentation of the block of residuals is performed in the encoding, we are able to change our compression depending on the statistics of the block being encoded. This is beneficial since the residuals resulting from prediction are nonstationary. Also, since we use HVQ to encode, we are able to gain the advantages of perceptual weighting, VQ with memory and adaptive quantization while maintaining the computational simplicity of table lookups.

Acknowledgments

Sanjeev Mehrotra was supported under a National Science Foundation Graduate Research Fellowship.

References

[1] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression,* Boston, MA: Kluwer Academic Pub., 1992.

[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, P. Chou and R.M. Gray, "Constrained and Recursive Hierarchical Table-Lookup Vector Quantization," *Proc. Data Compression Conference*, March 1996.

[6] N. Chaddha, S. Mehrotra and R.M. Gray, "Finite State Table-Lookup Hierarchical Vector Quantization for Images," *Proc. ICASSP'96*.

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

[8] H.-M. Hang and J.W. Woods, "Predictive vector quantization of images," *IEEE Trans. Comm.*, COM-33, pp. 108--1219, Nov. 1985.

[9] 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.

[10] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 36, pp.1445-1453, Sept. 1988.

[11] G.J. Sullivan and R.L. Baker, "Efficient Quadtree Coding of Images and Video," *IEEE Trans. Image Proc.*, vol. 3, pp. 327-331, May 1994.

[12] J. Vaisey and A. Gersho, "Image Compression with variable block size segmentation," *IEEE Trans. Signal Processing*, vol. 40, pp. 2040-2060, Aug. 1992.

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

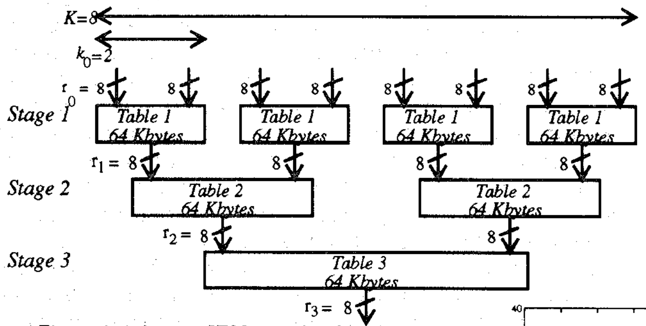

Figure 1. A 3-stage HVQ encoder ($k_m=2$, $r_m=8$)

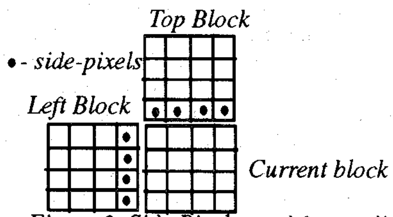

Figure 2. Side Pixels used for predicting block

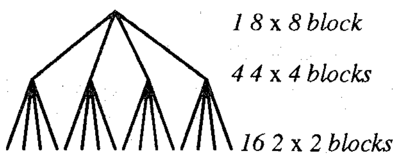

Figure 3. A three level quadtree with all nodes having 4 children

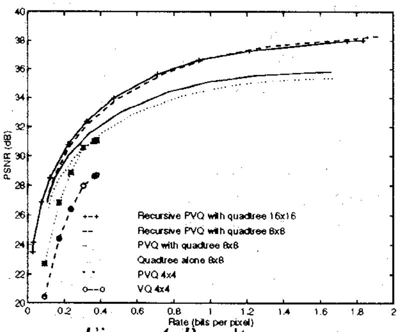

Figure 4. Results

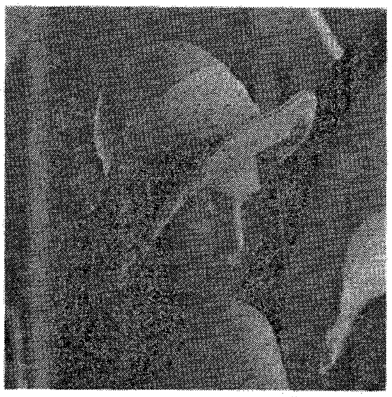

Figure 5. Lena compressed with JPEG at 0.33 bpp; PSNR = 32.6 dB.

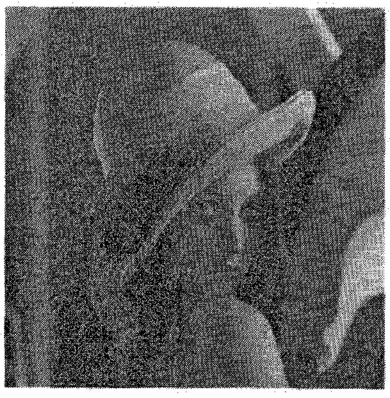

Figure 6. Lena compressed with recursive PVQ with quadtree at 0.33bpp; PSNR = 32.3 dB; Block sizes used in encoding: 2x2 up to 8x8

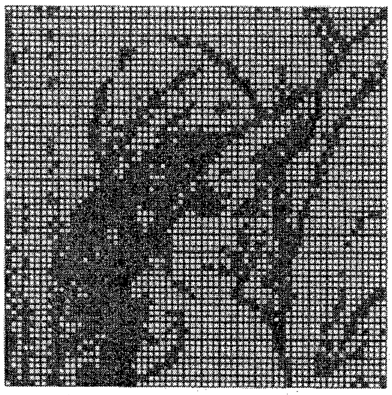

Figure 7. Segmentation map for Lena shown in figure 6 Each block gets 8 bits. The background regions are encoded using larger block sizes