Variable-Rate Tree Structured VQ for Image Compression

Young-Kee Jung, Jeong-Woo Lee and Yo-Sung Ho
Kwangju Institute of Science and Technology
572 Ssangam-dong Kwangsan-gu, Kwangju, 506-712, Korea
Phone: +82-62-970-2258, Fax: +82-62-970-2204, E-mail : ykjung@gogh.kjist.ac.kr

Abstract – For tree-structured vector quantization, there are two different approaches: tree growing and tree pruning. In this paper, we propose a new tree growing algorithm based on the brute force exhaustion method for constructing the subtrees within a tree. The performance of the proposed scheme is comparable to that of pruned tree structured vector quantizer (PTSVQ) at low rates, while it outperforms PTSVQ at high rates. However, the computational load involved in growing a tree increases as the height of the tree increases.

1. Introduction

It is well known that Vector Quantization (VQ) provides better performance with lower distortion than scalar quantization (SQ) at a given bit rate. In VQ, we group the source output into blocks or vectors. For example, we can take a block of L pixels from an image and treat each pixel value as a component of a vector of size or dimension L. Each image vector is then compared with a collection of representative codewectors taken from a previously generated codebook. The codewectors are also of dimension n. The best match codewector is chosen using a minimum distortion rule. After a minimum distortion codewector has been selected, its index k is transmitted using \( \log_2 N_c \) bits. At the receiver, this index is used as an entry to a duplicate codebook (a look-up table) to reproduce the corresponding codewector. The key elements in the construction of VQ codebook are codebook generation and codebook design. Codebooks are typically generated by using a training set of images that are representative of the images to be encoded. Although there are a number of ways of obtaining the vector quantizer codebook, most of them are based on one particular approach, popularly known as the Linde-Buzo-Gray (LBG)

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks Research Center at Kwangju Institute of Science and Technology.

algorithm. In finding the minimum distortion codevector for each image vector, a full search of the codebook can be performed at a computational cost of \( O(nN_c) \). The associated storage cost is \( nN_c \). For large codebooks, the search process becomes computationally intensive. This problem is related to codebook design. One of the most effective and widely used techniques for reducing the search complexity in VQ is to use a tree-structured codebook search. Tree-structured quantizers have had a fixed rate since they were based on trees of fixed length [1].

The performance of vector quantization for image compression can be improved by using a variable rate coder which is able to assign more bits to regions of an image that are active or difficult to code, and fewer bits to less active regions. In this case, there is a cost to be paid for the potential improvement offered by variable-rate coding. The cost is that a variable rate coding system requires extra buffering and bit control overhead if it is to be used with a fixed rate communications channel.

The central problem here is how to find the particular tree with a prescribed average length while minimizing the average distortion.

There are two approaches for optimal tree-structured vector quantizer: tree growing and tree pruning approach. The tree growing approach utilize the top-down method, where they start from the root node and build a tree until the desired rate is obtained [2]. This is based on the brute-force exhaustion, but has very high complexity. The tree pruning approach utilize the generalized BFOS algorithm in developing a variable-rate tree-structured vector quantizer by pruning back from a given tree to obtain a tree with a prescribed average length [3][4].

In this paper, we propose a new growing algorithm based on the brute force exhaustion algorithm for constructing the subtrees within a tree. The proposed growing tree method is more simple than the brute-force exhaustion algorithm by reducing the number of subtrees with small distortion loss and we compare its performance

222
with that of pruned tree structured vector quantizer (PTSVQ) in terms of Peak-Signal-to-Noise Ratio (PSNR).

2. The Distortion-Rate Framework

It is of interest to determine how best to trade off the rate and distortion by varying the size of the tree.

More specifically, if $T$ is a large tree, there corresponds to every pruned subtree $S$ of $T$ (denoted $S \subseteq T$) a variable-rate tree structured vector quantizer with average rate $u_1$ and average distortion $u_2$; the operational distortion-rate function

$$\tilde{D}_T(R) = \min \{u_1(S) 1u_2(S) \leq R\}$$

(1)

then specifies the optimal trade-off between rate and distortion in the restricted setting, where the quantizer is constrained to be some pruned subtree of a given tree $T$.

![Figure 1: Distortion/Rate Points of subtrees](image)

Because any tree $T$ has a finite number of pruned subtree, the operational distortion-rate function $\tilde{D}_T(R)$ has the staircase form shown in Figure 1.

3. Pruned Tree Structured VQ

As previously mentioned, complete tree-structured quantizers can be pruned back to obtain variable rate tree-structured quantizers. To perform the pruning optimally in a distortion-rate sense, the generalized BFOS algorithm can be used with $u_1$ as the expected length and $u_2$ as the expected distortion. Let $u_1$ and $u_2$ be two monotonic linear tree functionals with $u_1$ monotonically increasing and $u_2$ decreasing. Let $U(S) = (u_1(S), u_1(S))$ be the vector-valued function on the set of all subtrees of $T$ with $u_1$ and $u_2$ as components. Consider the set of points $(u(S); S \subseteq T)$ and its convex hull. By monotonicity, the singleton tree consisting of just the root $t_0$ has the smallest $u_1$ and the largest $u_2$; the full tree $T$ itself has the largest $u_1$ and the smallest. Therefore, $U(t_0)$ is the upper left corner of the convex hull; $U(T)$ is the lower right corner. This is the case shown in Figure 1 where $u_1$ represents the average rate and $u_2$ represents the distortion. Generalized BFOS algorithm is that the optimal subtrees are nested in the sense that if $(U(T), U(S_1), U(S_2), \ldots, U(S_n), U(t_0))$ is the list of vertices clockwise around (the lower boundary of) the convex hull of all distortion/rate pairs, then $t_0 < S_n < \cdots < S_2 < S_1 < T$. Hence it is possible to start with the full tree at $U(T)$ and prune back to the root at $U(t_0)$ producing a list of nested subtrees which trace out the vertices of the lower boundary of the convex hull.

4. A New Tree Growing Algorithm

In principle, the problem of equation (1) can be solved by brute force exhaustion. However, this solution is impractical if the original full tree is not small and it is difficult to know the number of subtrees in a tree. Accordingly, we utilize the modified brute force exhaustion method for constructing the subtrees within a tree. As shown in Figure 2, the growing tree of i-th depth includes that of all depth up to and including i-th level. This scheme is simpler than the brute-force exhaustion method. This method also reduces the number of subtrees to be constructed in a tree.

This scheme utilizes both balanced and unbalanced tree structures. Left child and right child node belong to unbalanced and balanced tree structures, respectively. That is, to make the subtrees at i-th level, we utilize the depth first algorithm for constructing left subtrees and the breadth first algorithm for constructing right subtrees. We assume that total distortion of two symmetric subtrees is about the same. In level 3, for example, reference subtree of its level is selected by arriving the bottom node using depth first algorithm, and that is extended as follows: this method is shown in Figure 3.

| Subtree 1 | A |
| Subtree 2 | A-1 |
| Subtree 3 | A-1-2 |
| Subtree 4 | A-1-2-3 |
| Subtree 5 | A-B |
| Subtree 6 | A-B-1 |
| Subtree 7 | A-B-1-2 |
| Subtree 8 | A-B-1-2-3 |
The number of the possible subtrees at level 3 is 8. As the subtrees could be made at ith level include the subtrees up to ith level, the number of subtrees at level 3 is 12.

![Diagram of Tree Growing Method](image)

Figure 2. Example of Tree Growing Method (L = 3)

The complexity of growing method is calculated as follows:

\[
N_0 = 1, \quad N_1 = 1, \quad N_2 = 2
\]

\[
N_s = 2^{s-1} \cdot \left[ \sum_{i=2}^{s-1} (2^{i-1} - 1) + 1 \right]
\]

\[
N = \sum_{n=0}^{L} N_s
\]

where \( N \) is the number of subtrees at level L, \( N_i \) is only the number of subtrees at level i not including up to i-1 level.

5. Simulation Results

In our experiments, we used several 512x512 image as a training sequence. All vectors have 2 x 2 dimension. PTVQ is performed with codebook made by TSVQ with rate 6.0 bits/sample. The proposed growing tree method is also performed with codebook at 6.0 bits/sample. That is, all possible subtrees are made at level 6. In this case, the number of subtrees is 1108. Fig. 4,5,6,7,8 demonstrates the performance and the results of variable-rate pruned TSVQ compared with those of the full-search VQ, the fixed-rate tree-structured VQ and the growing tree method.

As you can see in figure 4, the proposed growing tree method has the best performance, because this method is based on the brute-force algorithm. The performance of growing tree VQ is comparable to that of PTVQ at low rates, while it outperforms PTVQ at high rates. However, the computational load involved in growing a tree increases as the height of the tree increases. The pruning algorithm was implemented on a full tree of depth 6. To improve the performance of PTVQ, the larger initial tree has to be grown, which is then pruned.

![Graph of Simulation Results for LENA](image)

Figure 4. Simulation Results for LENA

6. Conclusions

We analyze the pruned tree-structured vector quantizer(TSVQ) and propose a new growing tree method. The proposed growing method doesn’t guarantee all subtrees of full trees, but guarantee lower complexity than the brute-force exhaustion method. We can easily estimate the number of subtrees in a tree. The pruning method finds a nested sequence of pruned subtrees of a given tree by optimally trading increases in distortion for decreases in the number of leaves. The performance is similar with PTVQ and superior at the certain condition.

References


Figure 3. Growing Method of subtrees at level 3: A, A-1, A-1-2, A-1-2-3

Figure 5. Original Image

Figure 6. Fixed rate TSVQ (0.5 bits/sample)

Figure 7. Full Search VQ (0.5 bits/sample)

Figure 8. Pruned TSVQ (0.5 bits/sample)