Tree-structured Encoder for Vector Quantization of Image Data *

Mohamed Kamel †
L. Guan ‡
† Department of Systems Design Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
‡ Mitra Imaging Inc.
Waterloo, Ont., N2V 1C5

Abstract

Vector quantization repeatly use the nearest neighbor search in the codebook design and in the encoding process. Efficient algorithms to perform this search and reduce the complexity of encoding can impact the utility of VQ as an effective encoding technique. One way to achieve this efficiency is to develop some structure on the codebook. In this paper, we focus on the tree-structured encoders. We analyze theoretically and experimentally the behavior of the search algorithm. We then propose a new tree-structured encoder which utilizes the properties of the distribution of vectors of image data. The effectiveness of the proposed technique is illustrated by comparing its performance to existing tree-structured techniques.

1 Introduction

A k-dimensional N level vector quantizer (VQ) is a function $Q : R^k \rightarrow C \subset R^k$, where R^k is a k-dimensional Euclidean space, and $C = \{\mathbf{c}_i \ ; \quad 1 \leq i \leq N\}$, called the codebook, is a finite subset of R^k containing N codewords, $\mathbf{c}_i$. To perform the quantization, an image is divided into smaller blocks of $m \times m = k$ pixels. Each block is represented as a $k-$dimensional vector, $\mathbf{q}$. As the codebook is assumed to be known, each $\mathbf{q}$ can be represented by the index of the codeword closest to it. This representation requires a number of bits much less than the number of bits required to represent $\mathbf{q}$, which leads to reducing the representation of images to a rate of $\log_2 N^{1/k}$ bits per pixel.

The key problem in VQ is the nearest neighbor search (NNS) for the best codeword. Methods that are proposed to solve the problem involve constructing some structures on the codewords to facilitate the searching. Tree structured VQ encoder is one of the methods. Two problems have been identified in [2] to design such a tree: (1) selecting a suitable hyperplane for each nonterminal node that will minimize the average search complexity, and (2) determining which Voronoi polyhedrons are split by a hyperplane.

*This work is partially supported by the Natural Science and Engineering Research Council of Canada

The first problem is difficult to solve. It can be transformed into the problem of constructing an optimal binary tree which has been identified by Hyafil et. al. [4] as a NP-complete problem. A heuristic method to design an approximately optimal tree is proposed by Cheng [2]. This method, at each step of space partitioning in the binary tree construction, selects a hyperplane among the bisects of the codewords.

In this paper we will address a way of constructing a tree structured encoder which utilizes image vector distribution properties according to the image model proposed in [3].

2 Tree Structured Encoders

The Voronoi polyhedron v_t of a codeword $\mathbf{c}_t$ is the locus of the points which are closer to $\mathbf{c}_t$ more than to any other codewords. This definition implies that

$$\bigcup_{i=1}^{N} v_i = R^k, \tag{1}$$

$$v_j \bigcap_{j \neq i} v_i = \Phi. \tag{2}$$

If a source vector $\mathbf{x}$ is contained in a Voronoi polyhedron v_t, the corresponding codeword $\mathbf{c}_t$ is the nearest codeword to $\mathbf{x}$. To find the best codeword for $\mathbf{x}$, it is sufficient to identify the Voronoi polyhedron that contains $\mathbf{x}$.

Designing a tree structured encoder involves constructing a structure on the codebook to efficiently identify a polyhedron which most likely contains the best codeword for a source vector.

Let H be a hyperplane in $k-$ dimensional space,

$$H = \{\mathbf{x} \mid \mathbf{x} \in R^k : \mathbf{a}^T\mathbf{x} + b = 0\}, \tag{3}$$

where $\mathbf{a}$ is a $k-$dimensional vector of coefficients and b is a constant. H splits R^k into the left half space R_l^k, and the right half space R_r^k, where

$$R_l^k = \{\mathbf{x} \mid \mathbf{x} \in R^k : \mathbf{a}^T\mathbf{x} + b \leq 0\}, \tag{4}$$

$$R_r^k = \{\mathbf{x} \mid \mathbf{x} \in R^k : \mathbf{a}^T\mathbf{x} + b > 0\}. \tag{5}$$

H also partitions the Voronoi polyhedrons.

We use $v_t \dashv H$ to denote that v_t is on the left side of H, and $v_t \vdash H$ to denote that v_t is on the right side of H, and $v_t \dagger H$ to denote that v_t is split by H, that is

$$v_k \dashv H \rightarrow \forall \mathbf{x} \in v_k (\mathbf{x} \in R_l^k \wedge \mathbf{c}_k \in R_l^k), \quad (6)$$

$$v_k \vdash H \rightarrow \forall \mathbf{x} \in v_k (\mathbf{x} \in R_r^k \wedge \mathbf{c}_k \in R_r^k), \quad (7)$$

$$v_k \dagger H \rightarrow \exists \mathbf{x} \in v_k (\mathbf{x} \in R_r^k \wedge \mathbf{c}_k \in R_l^k) \vee (\mathbf{x} \in R_l^k \wedge \mathbf{c}_k \in R_r^k). \quad (8)$$

Without ambiguity we also use $\mathbf{x} \dashv H$ to denote that $\mathbf{x}$ is on the left side of H and $\mathbf{x} \vdash H$ to denote that $\mathbf{x}$ is on the right side of H.

Let C_l and C_r be the set of codewords whose polyhedron is on the left and right side of H, respectively, C_c be the set of codewords whose polyhedron is split by H, we have the following three sets:

$$C_l \equiv \{\mathbf{c}_k : v_k \dashv H\} \quad (9)$$

$$C_r \equiv \{\mathbf{c}_k : v_k \vdash H\} \quad (10)$$

$$C_c \equiv \{\mathbf{c}_k : v_k \dagger H\} \quad (11)$$

For such a partitioning, namely H, to test if a source vector $\mathbf{x}$ satisfies $\mathbf{x} \dashv H$, or $\mathbf{x} \vdash H$ can be easily done. According to the result, for instance $\mathbf{x} \vdash H$, the codewords in C_l can be rejected.

However, we can not reject any codewords in C_c according to the result of the test. If we consider two sets of codewords associated with the hyperplane H:

$$\overline{C}_l = C_l \cup C_c, \quad (12)$$

$$\overline{C}_r = C_r \cup C_c, \quad (13)$$

we obtain two necessary conditions for a codeword $\mathbf{c}_t$ to be the closest candidate of a given source vector $\mathbf{x}$:

$$\mathbf{c}_t = Q(\mathbf{x}) \in \overline{C}_l, \quad \text{if } \mathbf{x} \dashv H, \quad (14)$$

$$\mathbf{c}_t = Q(\mathbf{x}) \in \overline{C}_r, \quad \text{if } \mathbf{x} \vdash H. \quad (15)$$

These necessary conditions serve as the basis of a tree structured encoder.

The above conditions also mean that the computation costs of searching for the best codewords is reduced. The total savings due to the partitioning would be at least

$$| C | - \max\{| \overline{C}_r |, | \overline{C}_l |\}, \quad (16)$$

where C represents the set of all codewords in the space that H partitions.

When the partitioning is recursively done in subsets R_l^k and R_r^k, and accordingly in $\overline{C}_l$ and $\overline{C}_r$, further computational savings can be expected. Such a partitioning pattern can be represented efficiently using a binary tree structure [1] [2] in which each non terminal node corresponds to a partion by a hyperplane, and the terminal node corresponds to a set of codewords which have their Voronoi polyhedron partitioned by hyperplanes in upper level non terminal nodes.

Since the computational complexity of the tree structured encoder depends on the number of codewords corresponding to the the terminal node and the number of nonterminal nodes that a source vector $\mathbf{x}$ has to be checked against, in order to minimize the computational efforts we would like to maximize (16) in each partitioning. It immediately follows that

$$\min_{\textit{set of all bisects}} \max\{| \overline{C}_r |, | \overline{C}_l |\}, \quad (17)$$

would be an appropriate criterion for selecting the partitioning hyperplane.

The lower bound of the computational complexity, in terms of Euclidean distance calculation, of a tree structured encoder to identify the best codeword for a source vector $\mathbf{x}$ is $O(\log_2 N)$. In order to achieve this lower bound, it is necessary that at each nonterminal node the Voronoi polyhedrons of codewords are partitioned into half and no Voronoi polyhedron is split. However, usually C_c is not empty, therefore, in the children nodes the number of codewords is more than half of the codewords in their parent nodes. As the level of the tree increases, the number of codewords in each nonterminal node does not decrease logarithmically.

Under the assumption of uniform codewords distribution, a worst case estimation of $| C_c |$ was given by Wu [5]. In [5] the codewords were assumed to be uniformly distributed such that in the $k-$dimensional space the number of Voronoi polyhedrons that the best 1-dimensional "space" (a line) might split is $O(N^{1/k})$. Since a hyperplane $H \subseteq R^k$ is an $(k-1)$-dimensional space, it follows that the average minimum number of possible Voronoi polyhedrons that the "best" hyperplane might split under this worst situation would be $O(N^{(k-1)/k})$.

According to the way the space is partitioned, the following recurrence function was given in [5] to estimate the average number of codewords at level t of the tree:

$$| C^{(t)} | = \frac{| C^{(t-1)} | + [(| C^{(t-1)} |)^{(k-1)/k}]}{2} \quad (18)$$

where $C^{(t)}$ stands for the expected number of codewords in nodes at t levels, $0 < t$, and $| C^{(0)} | = N$. Wu [5] showed that the efficiency of the binary tree partitioning algorithm decreases as the number of dimensions of the space increases.

The above estimation is based on the worst case scenario. In order to investigate the efficiency of the tree structured VQ on real image data, an experiment to build up a tree structured VQ using the previous method has been conducted. At each level of the partitioning we record the number of codewords, $| \overline{C}_l |$, $| \overline{C}_r |$, that are passed down to children nodes, then

take the average on the numbers of codewords at the same level of the tree with respect to the total number of nodes in that level of the tree. This generates a relationship between the number of codewords at nodes in different levels and the height of the tree.

The curve of the relationship between the number of codeword, $|\overline{C}_l|$, $|\overline{C}_r|$ versus the depth of the tree, as well as the theoretical worst case formulated in (18) is plotted in Fig.1.

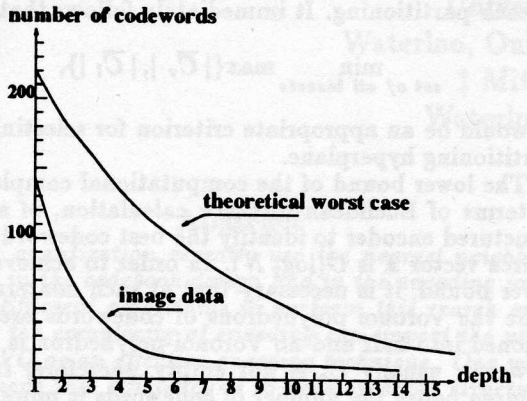

Figure 1: Relationship between the number of codewords vs the depth of tree

The search complexity of the tree-structured encoder comes from two factors: the cost of locating the corresponding terminal nodes which contains the best codeword, and the best codeword searching within the terminal node bucket. We observed that at the first few levels when the tree grows from one level to the next the codewords held in the children nodes will reduce drastically. Although we could extend the height of the tree h until there is only one codeword in the terminal node, it may not help us very much to reduce the computational costs if we consider that an operation of checking a vector against a hyperplane in a nonterminal node takes just as much computational efforts as calculating an Euclidean distance between a source vector and a codeword.

In order to see this, we added the costs of checking input vector against hyperplanes, at every nonterminal node on the path from the root to the terminal node, to the costs of distance calculations within the terminal node and plotted the gain in computational efficiency as the tree grows one level at a time. The curve which is labeled as the theoretical worest case is shown in Fig.2.

It becomes obvious from the plot that we don't get much gain by partitioning the search space after the tree has grown to a certain number of levels. Further partitioning after that adds one more level to the tree and may create two children nodes that contain only one codeword less than their parent. When we consider that examining a source vector against the hyperplane on the nonterminal node takes the same amount of computational effort, the net gain from this partitioning would be zero.

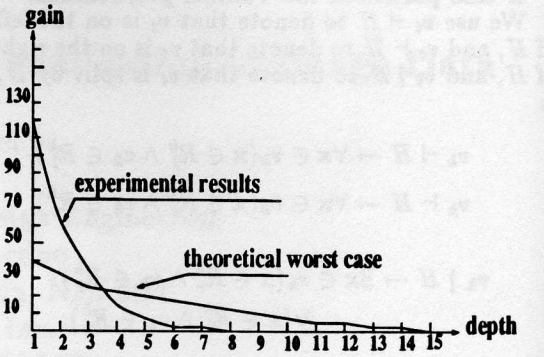

Figure 2: Gain of partitioning vs the depth of tree

3 Modified Tree-structured Encoder

We observed that the hyperplane which splits in the direction orthogonal to the center line would not split as many Voronoi polyhedrons as a hyperplane that splits in other directions on average. It becomes quite natural that a partitioning should be conducted in the direction that is orthogonal to the direction where the most codewords lie. As the partitioning continues, the "shape" of the space becomes more and more like a "ball", in which every Voronoi polyhedron shares a "face" with every other Voronoi polyhedrons whose centroids are in the same node. Under this circumstance further partitioning will split a large number of polyhedrons, especially when the partitioning is in high dimensional space. Therefore increasing the height of the tree may not reduce the computational costs at all.

From a recently proposed image model [3] we know that the partitioning is only effective at the first few levels. The observation, therefore, leads us to consider another way of improving the tree structured VQ other than pursuing further partitioning at lower levels of the tree. We use some parallel hyperplanes to partition the space through the direction which is normal to the center line, then perform the tree structured partitioning within the subspace. The advantage of this partitioning strategy is that because of the features of the organization we can "skip" some higher levels of the tree in the encoding process.

Suppose that the distribution of image vectors is along the center line l, and the projection point of a vector $\mathbf{x}$ on l is very easy to know. Furthermore, the necessary condition for a codeword to be the best codeword to a source vector $\mathbf{x}$ would be that the projection interval of the Voronoi polyhedron of the best codeword on l has to include the projection point of $\mathbf{x}$.

With these observations we can develop an algorithm which constructs a group of trees (forests) for the encoder. The modified tree-structured encoder will first direct the input source vector, $\mathbf{x}$, right to the "bin" on l which contains the projection of the Voronoi polyhedron of the nearest codeword, then perform the tree-structured search procedure within the "bin".

The algorithm that designs such an encoder is given as follows:

Algorithm Tree_Encoder_Design
1. **Projection** : Project the Voronoi polyhedron of every codeword on l, use a hyperplane orthogonal to l to partition space into subspaces;
2. **For** every subspace **do**
   2.1. **Tree_Construction** : constructing a tree encoder;
   end

end

The procedure **Projection** is a preprocessing which (1) assigns each training vector $\mathbf{x} \in X$ to the nearest codeword in C, (2) for every Voronoi polyhedron of codewords, gets the maximum and minimum projection points on l, (3) partitions the space into a number of subspaces using hyperplanes normal to l so that the number of codewords associated with every subspace will be approximately the same, the codewords are stored as initial trees, and (4) associates each subspace a hyperplane set which contains hyperplanes, each hyperplane is associated with the set of codewords on the left, right and split by the hyperplane in the subspace.

Accordingly we have the encoding procedure as the following:

Algorithm Modified_Tree_Encoder
1. Project data vector $\mathbf{x}$ to l;
2. Select a tree generated on the space where $\mathbf{x}$ is projected to;
3. Tree_Encoder($\mathbf{x}$, *tree*);

where Tree_Encoder may be any of the existing tree encoders, such as the one in [2].

4 Comparison of Tree-structured Encoders

Both the tree-structured encoder (TVQ) suggested in [5], and the modified tree-structured encoder(MTVQ), as described in the previous section, have been implemented and tested on four images. The results are shown in Table 1.

methods	TVQ		MTVQ	
image	time	ADC	time	ADC
Lena	10.41 (sec)	10	9.19 (sec)	8
Girl	19.78 (sec)	11	17.9 (sec)	8
Gold	18.95 (sec)	10	18.70 (sec)	10
UW	18.19 (sec)	10	17.38 (sec)	9

Table 1: Performance comparison of Tree-structured partitioning VQ results

The results show a consistent improvement of the modified tree-structured VQ over its original version in terms of CPU time and average distance calculation (ADC) for encoding a data vector.

The complexity of exhaustively testing all N^2 bisects between codewords against the training set of M vectors costs $O(N^2M)$, and there are $2^{h-1}-1$ nonterminal nodes in a full binary tree. On each node every bisect needs to be tested, therefore, the total cost of the algorithm is $O(N^2M+(2^{h-1}-1)N^2)$. The modified tree structured VQ will implement the tree structured VQ in each interval on l. Suppose there are i intervals and a tree generated on an interval will not be higher than h', the total cost of generating the forest will be in the order of $O(N^2M+i(2^{h'-1}-1)N^2)$.

5 Conclusion

Tree structured VQ encoders provide a reasonable compromise in terms of reducing the number of nearest neighbor search of the codeword at the cost of small loss of optimality of the encoding. Based on a theoretical image model which explains the correlation of image vectors and experimental observations, we proposed a new tree-structured encoder which utilizes properties of the distribution of vectors of image data. The effectiveness of the proposed technique is illustrated by comparing its performance to existing tree-structured techniques.

References

[1] J. L. Bentley, " Multidimensional binary binary search trees used for associative searching," *Communication of ACM*, Vol. 18, No. 9, pp. 509-517, Sept. 1975.

[2] De-Yuan Cheng and Allen Gersho, " A Fast Codebook Search Algorithm for Nearest-Neighbor Pattern Matching", *Proceedings of 1986 International Conference on Acoustics, Speech, and Signal Processing*, pp. 265-268, Tokyo, Japan, 1986.

[3] Guan, L. (1991). Fast Algorithms for Vector Quantization of Image Data, Ph.D dissertation, Department of Systems Design Engineering, University of Waterloo.

[4] L. Hyafil, and R. L. Rivest, "Constructing Optimal Binary Decision Trees is NP-Complete", *Information Processing Letters*, Vol. 5, No.1, pp15-17, May 1976.

[5] Xiaolin Wu, "A Tree-Structured Locally Optimal Vector Quantizer", *Proceeding of 10th International Conference on Pattern Recognition*, Vol. 2, pp176-181, 1990.