Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 758734, 9 pages
http://dx.doi.org/10.1155/2014/758734

Research Article

A Hybrid Vector Quantization Combining a Tree Structure and a Voronoi Diagram

Yeou-Jiunn Chen,[1] Shih-Chung Chen,[1] and Jiunn-Liang Wu[2]

[1] *Department of Electrical Engineering, Southern Taiwan University of Science and Technology, Tainan 710, Taiwan*
[2] *Department of Otolaryngology, College of Medicine, National Cheng Kung University, Tainan 701, Taiwan*

Correspondence should be addressed to Jiunn-Liang Wu; jiunn@mail.ncku.edu.tw

Received 23 December 2013; Accepted 13 April 2014; Published 4 May 2014

Academic Editor: Xinkai Chen

Multimedia data is a popular communication medium, but requires substantial storage space and network bandwidth. Vector quantization (VQ) is suitable for multimedia data applications because of its simple architecture, fast decoding ability, and high compression rate. Full-search VQ can typically be used to determine optimal codewords, but requires considerable computational time and resources. In this study, a hybrid VQ combining a tree structure and a Voronoi diagram is proposed to improve VQ efficiency. To efficiently reduce the search space, a tree structure integrated with principal component analysis is proposed, to rapidly determine an initial codeword in low-dimensional space. To increase accuracy, a Voronoi diagram is applied to precisely enlarge the search space by modeling relations between each codeword. This enables an optimal codeword to be efficiently identified by rippling an optimal neighbor from parts of neighboring Voronoi regions. The experimental results demonstrated that the proposed approach improved VQ performance, outperforming other approaches. The proposed approach also satisfies the requirements of handheld device application, namely, the use of limited memory and network bandwidth, when a suitable number of dimensions in principal component analysis is selected.

1. Introduction

With the development of wireless networks and handheld devices, multimedia data has become a popular communication medium. Generally, it requires substantial storage space and network bandwidth. However, handheld devices that offer limited memory and network bandwidth are seriously restricted in multimedia data applications. To overcome these problems, vector quantization (VQ), which yields a simple architecture, rapid decoding ability, and high compression rate, is widely used to reduce the size of multimedia data [1–5]. Thus, the ability of VQ to balance storage and computational complexity is useful in handheld device applications that involve multimedia data. VQ then could be applied to different applications such as 3D medical image compression [6], speech coding [7], information hiding [8, 9], and artistic style learning [10].

A VQ is defined as a mapping function that maps a k-dimensional vector space R^k into a finite subset $C = \{c_1, c_2, \ldots, c_N\}$. Each vector c_i is called a codeword and a set C with N codewords is called a codebook. Generally, VQ involves three procedures: the codebook design, encoding, and decoding procedures. During the encoding procedure, an input vector x is turned into an index value that indicates the location of the chosen codeword in C, allowing the size of the multimedia data to be substantially reduced. Conversely, during the decoding procedure, a codebook lookup operation is applied to determine the optimal codeword, allowing the quantized multimedia data to be reconstructed. Therefore, the quality of reconstructed multimedia data depends on the codebook and chosen codewords. The codebook design procedure is used to design the codebook used in the encoding and decoding procedures. A typical codebook is obtained from numerous training vectors extracted from a set of multimedia data by using a clustering algorithm. The well-known Linde-Buzo-Gray (LBG) clustering algorithm [11] is widely applied in codebook construction.

Numerous approaches have been proposed to improve codeword search efficiency and these can be classified as the full-search equivalent and partial-search approaches. Full-search vector quantization (FSVQ) is the most simple and intuitive method, but it is inefficient at large codebook sizes [12]. Full-search equivalent approaches [13–16] have been proposed to save computational time. These approaches not only use rough distortion elimination instead of executing the entire dimensions, but also filter out impossible codewords by using certain rejection conditions. However, the search times using these approaches are larger than those when using partial-search approaches. Moreover, regarding multimedia data, a certain degree of distortion is tolerable; thus, developing a partial-search-based VQ is critical when using multimedia data in handheld devices.

Partial-search approaches primarily use specifying data structures to organize codebooks and search for optimal codewords within a local search domain. This enables such approaches to save substantial computational time. Among partial-search approaches, tree-structured vector quantization (TSVQ) is one of the most effective approaches [17]. In TSVQ, a binary tree is applied to reduce computational complexity from $O(kn)$ to $O(k \log n)$. However, the codeword determined by TSVQ is not sufficiently near the input vector, yielding poor quality of the reconstructed multimedia data. To improve TSVQ performance, multistage (MTSVQ) [18] and closest-couple TSVQ [19] have been used to determine the optimum codeword by enlarging the search space. However, these approaches may traverse wrong direction paths caused by the decision criteria in each node, degrading performance.

To avoid traversing wrong direction paths, projection-based VQ approaches [20–24] have been proposed. Most of these approaches applied principal component analysis (PCA) to preserve most information of data, which is called the ratio of preserved information. This allows a vector to be transformed into a low-dimensional coordinate system; geometric measurements are subsequently applied to determine the nearest codeword. Cui et al. applied a B^+ tree to rapidly locate an initial codeword and subsequently used K-nearest neighboring candidates to determine an optimal codeword [23]. Because the candidates of each leaf node may overlap, the candidates are not well organized for locating the optimal codeword. Voronoi diagrams [24–27] can be used to efficiently interpret the relations between neighboring codewords, and Chang and Wu [24] used a Voronoi diagram to model the geometric interpretation of neighboring codewords in a transformed two-dimensional (2D) coordinate system. By using Voronoi edges and vertices, an initial codeword was quickly found, and a planar-oriented ripple search algorithm was then applied to enlarge the search space. However, it was limited by the 2D space of the Voronoi diagram; this prevented it from precisely modeling the geometric interpretation.

In this study, a hybrid vector quantization (HVQ) combining a tree structure and a Voronoi diagram is proposed to accelerate the VQ quantization process. By retaining certain suitable principal components of PCA, sufficient information regarding relations between each pair of codewords is preserved in the projected coordinate system. A B-tree that indexed the principal components is used to rapidly determine an initial codeword. To precisely enlarge the search space of the initial codeword stored in a leaf node, a Voronoi diagram is applied to generate the Voronoi regions and their corresponding neighbors. However, because increasing the number of principal components causes a corresponding increase in the number of neighbors for each Voronoi region, the neighbors of each Voronoi region are clustered into N groups, and a greedy search algorithm is applied to rapidly search for the optimal codeword.

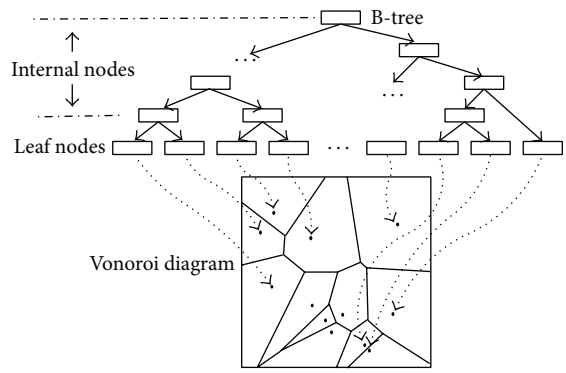

Figure 1: An example of structure for HVQ combining tree structure and Voronoi diagram.

The rest of this paper is organized as follows. Section 2 describes the HVQ, including the tree structure, Voronoi diagram, and greedy search algorithm. Section 3 presents the results of a series of experiments examining the performance of the proposed approach. In Section 4, the conclusion and recommendations for future research are presented.

2. Hybrid Vector Quantization

In this section, a new HVQ combining a tree structure and a Voronoi diagram is presented to improve performance beyond that of previously proposed methods. Figure 1 shows the HVQ structure, which combines a tree structure and a Voronoi diagram. First, a tree structure in a selected dimensional space is applied to quickly find an initial codewords. Second, Voronoi diagram is used to enlarge the search space of an initial codeword. Finally, a greedy search algorithm is proposed to search an optimum codeword in Voronoi diagram. The procedure for constructing the hybrid structure is detailed in the following subsections. PCA-based tree structure and Voronoi diagram are detailed in Sections 2.1 and 2.2. Then, the greedy search algorithm based vector search procedure is presented in Section 2.3.

2.1. PCA-Based Tree Structure. Because the expected minimal distance between two vectors in a high-dimensional space is large, codewords tend to be nearly equidistant to an input vector, rapidly degrading performance [28]. Selecting a suitable number of codeword dimensions could overcome this problem; thus, PCA efficiently reduces dimensions when

a ratio of preserved information is selected. In PCA, codewords are normalized to be zero mean and unit variance (denoted as x_i) and allowing a covariance matrix Σ to be obtained as follows:

$$\Sigma = \sum_{i=1}^{K} (x_i - \mu)(x_i - \mu)^T, \quad (1)$$

where μ denotes the mean of codewords and K is the size of the codebook. Using the covariance method, the eigenvalues $\Lambda = \{\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_K\}$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_K$, and the corresponding eigenvectors $A = \{a_1, a_2, a_3, \ldots, a_K\}$ can be computed. For a specified application, the computational complexity and ratio of preserved information in the projected subspace, S_b, is balanced by selecting a suitable b. The ratio of preserved information, Λ_b, is defined as follows:

$$\Lambda_b = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_b}{\lambda_1 + \lambda_2 + \cdots + \lambda_K}. \quad (2)$$

Using A, the codewords in C are easily projected to S_b, resulting in a set $C' = \{c_1', c_2', \ldots, c_N'\}$. C' is then used to construct a tree structure to rapidly determine an initial codeword. A pseudocode representation of the tree construction is as shown in Pseudocode 1.

Lines 1–3 are used to generate a root for tree structure and the recursive function *ConstructTree* allows the construction of the entire tree structure for C'. In *ConstructTree*, a predefined threshold δ is applied to limit the levels of the tree structure. When δ is set as 1, each leaf node is linked to a codeword in C' in a one-to-one and onto relationship. Regarding each internal node, $node.centroid_L$ and $node.centroid_R$ are used to traverse the paths, allowing an initial codeword stored in leaf node to be found.

2.2. Voronoi Diagram Structure. The search space of the initial codeword can be enlarged to reduce distortion. Using a Voronoi diagram to interpret relations of neighboring codewords in S_b [26, 27] allows the search space of the initial codeword to be enlarged. Using the PCA detailed in Section 2.1, C' is then used to generate the Voronoi diagram structure, which is a partition of S_b. A set of Voronoi regions for C' can then be obtained and denoted as $V = \{V(c_1'), V(c_2'), V(c_3'), \ldots, V(c_N')\}$. $V(c_i')$ is the Voronoi region of c_i' and must satisfy the following property:

$$V(c_i') = \left\{ x \in S_b \mid \left\| x - c_i' \right\| < \left\| x - c_j' \right\| \ \forall i \neq j \right\}. \quad (3)$$

It is clear that each Voronoi region contains only one projected codeword. Moreover, all points in $V(c_i')$ are closer to c_i' than to any other projected codewords; thus, $V(c_i')$ must be a nonempty set. A Voronoi region is bounded by Voronoi edges and Voronoi vertices, but some Voronoi regions are necessarily unbounded. In this study, a lifting-based algorithm, using tangent hyper planes, was used to construct a Voronoi diagram, as detailed in [28]. Figure 1 exemplifies a Voronoi diagram structure.

Increasing the number of dimensions of the Voronoi diagram structure substantially increases the number of neighbors of each Voronoi region. To reduce the search space, N centroids of neighbors are estimated using the LBG algorithm and used to partition the search space into N groups. An input vector is guided to efficiently search for an optimal codeword by using the partition with the closest centroid.

2.3. Greedy Search Algorithm. In this subsection, a greedy search algorithm is proposed as the VQ encoding procedure; this can be summarized in three steps. First, an input vector x is projected to S_b by using the eigenvectors A proposed in Section 2.1. Second, a tree-structure-based search algorithm is adopted to find an initial codeword. In this step, two centroids stored in the internal nodes of the tree structure are used to traverse a path, exhibiting minimal distortion. This allows finding a leaf node contained an initial codeword c_i' and linked to a Voronoi region $V(c_i')$. Third, the neighboring Voronoi regions of $V(c_i')$ are adopted to enlarge the search space, searching for a suitable optimal codeword. In this study, using neighboring regions to enlarge the search space is referred to as "rippling." To accelerate the search process by using the Voronoi diagram, the neighboring Voronoi regions are classified into M groups, and the centroid of the ith group is denoted as $centroid_i$. Increasing the number of ripples, the number of Voronoi regions to search correspondingly increases. To reduce this effect, a Voronoi region with minimal distortion is chosen from the neighboring Voronoi regions and is searched for the optimal codeword. A pseudocode representation of the greedy search algorithm is as shown in Pseudocode 2.

A recursive strategy is used in the TreeSearch and GreedyRipple functions, searching for an initial codeword and a corresponding Voronoi region and traversing the neighboring Voronoi regions to locate an optimal codeword. In the proposed approach, Lines 20–24 yield an optimal codeword in a Voronoi region and traverse the Voronoi regions in the Voronoi diagram structure.

3. Experimental Results and Discussion

To evaluate the proposed approach, mel-frequency cepstrum (MFCC) based vectors that comprise 39 dimensions (12-MFCC, 12-ΔMFCC, 12-ΔΔMFCC, Energy, ΔEnergy, and ΔΔEnergy) were used in this study [29]. Two sets of 6654 and 3026 vectors were generated based on the TCC300 [30] and treated as the training and testing sets, respectively. Using the training set, the LBA algorithm was adopted to construct a codebook comprising 2048 codewords (denoted as CB2048).

3.1. Experimental Results of the Ratio of Preserved Information. First, the effect of the ratio of preserved information was examined. Regarding vectors comprising 39 dimensions, the ratios of preserved information were set as 56.56%, 68.38%, and 83.23% for the remaining 2, 3, and 4 principal components, respectively, by setting the variances of normal distributions. Then, 5012 vectors and 1024 vectors were randomly generated to form the simulated training and testing sets, respectively. A codebook comprising 2048 codewords

```
Input: a set of projected codewords C'
Output: a tree-structure based codebook
Tree construction procedure:
(1)  root = new node    // obtain the root of the tree structure
(2)  ConstructTree(root, C')    // apply the recursive function to construct the tree
(3)  return root
Function ConstructTree(node, W)
(4)  According to the set of input codewords, W, apply the LBG algorithm to
     estimate two centroids, node.centroid_L and node.centroid_R
(5)  Let s_L = φ and s_R = φ
(6)  Each w ∈ W is assigned to s_L or s_R depending on whether w is closer to
     node.centroid_L or node.centroid_R
(7)  If the number of s_L is greater than a predefined threshold δ then {
(8)      node.son_L = new node
(9)      ConstructTree(node.son_L, s_L)
(10) }
(11) Else {
(12)     node.son_L = φ
(13)     Find a codeword c ∈ C' such that ‖c − node.centroid_L‖ is minimum
(14)     node.InitialCodeword_L = c
(15) }
(16) If the number of s_R is greater than a predefined threshold δ, then {
(17)     node.son_R = new node
(18)     ConstructTree(node.son_R, s_R)
(19) }
(20) Else {
(21)     node.son_R = φ
(22)     Find a codeword c ∈ C' such that ‖c − node.centroid_R‖ is minimum
(23)     node.InitialCodeword_R = c
(24) }
```

Pseudocode 1

Table 1: The experimental results of different ratios of preserved information.

Number of dimensions	Ratio of preserved information	Distortion of initial codeword	Optimal codeword with 1-ripple regions	
			Distortion	Number of traversed Voronoi regions
2	56.56%	4.105	3.227	6.4
3	68.38%	3.631	2.784	16.4
4	82.23%	3.385	2.197	39.1

was then constructed to analyze the characteristics of the proposed approach when using various ratios of preserved information.

The proposed approach was examined using one ripple to find an initial codeword and corresponding optimal codeword. The experimental results in Euclidean distance are shown in Table 1. The effect of the tree structure was analyzed, indicating that the depths of the tree structures for codebooks comprising various numbers of dimensions were almost the same. Thus, the effect of the number of dimensions can be neglected. Consequently, only the number of neighboring Voronoi regions was considered in this experiment. Increasing the ratio of preserved information from 56.56% to 82.23%, the distortion was reduced from 4.105 to 3.385, and the distortion reduction rate was 17.54%. Thus, the ratio of preserved information substantially affected the accuracy of the initial codeword.

Second, the effect of the Voronoi diagram was examined using one ripple and treating the neighboring Voronoi regions as one group. Increasing the number of dimensions from 2 to 4 increased the average number of neighbors for a Voronoi region from 6.4 to 39.1, decreasing the distortion of the optimal codeword from 3.227 to 2.197. The distortion reduction ratio was 31.92%. Clearly, increasing the ratio of preserved information greatly improved accuracy and increased the computational complexity. Thus, selecting a suitable ratio of preserved information is necessary to attain a balance between accuracy and search speed.

Input: an input vector x, a root node of tree structure *root*, and a set of Voronoi regions V
Output: an optimal codeword $\widetilde{c}$
Greedy search algorithm:
(1) Project x to S_b, and denote the projected vector as $\widetilde{x}$
(2) Call $V(c_i') = TreeSearch(\widetilde{x}, root)$ to find a Voronoi region
(3) Call $\widetilde{c} = GreedyRipple(\widetilde{x}, V(c_i'), \Theta, \infty)$ to find an optimal codeword with a predefined number of ripples, Θ, and set the minimal distance is as infinite
(4) Return $\widetilde{c}$
Function $TreeSearch(x', node)$
(5) If $\|x' - node.centroid_L\| < \|x' - node.centroid_R\|$ then {
(6) If $node.son_L = \phi$ then
(7) Return a Voronoi region $node.InitialCodeword_L$
(8) Else
(9) Call $TreeSearch(x', node.son_L)$
(10) }
(11) Else {
(12) If $node.son_R = \phi$ then
(13) Return a Voronoi region $node.InitialCodeword_R$
(14) Else
(15) Call $TreeSearch(x', node.son_R)$
(16) }
Function $GreedyRipple(x', V(c'), \theta, \varepsilon)$
(17) If $\theta > 1$ then {
(18) Find a $centroid_i$ such that $\|x' - V(c').centroid_i\|$ is minimized
(19) Let Ω be the neighbors in group $centroid_i$ of Voronoi region $V(c')$
(20) Find a $\omega \in \Omega$ such that the distance, Δ, between x' and the codeword of ω is minimized
(21) If $\varepsilon > \Delta$ then {
(22) $\widehat{c} = GreedyRipple(x', \omega, \theta - 1, \Delta)$
(23) Return $\widehat{c}$
(24) }
(25) }
(26) Return c'

Pseudocode 2

3.2. Experimental Results of Finding Initial Codeword. Because the number of dimensions and levels of a tree structure substantially affect performance, CB2048 was adopted to analyze these two factors. To control the depths of the tree structures, the number of tree nodes was limited to 256, 512, 1024, and 2048. In addition, the number of dimensions was set as 2, 3, and 4.

Figure 2 shows the experimental results, indicating that distortion was reduced by either increasing the number of dimensions or depths of the tree structure. First, we analyzed the effect of the depth of tree structure. Increasing the nodes of the tree structure from 256 to 2048 yielded distortion reduction rates of 6.68%, 13.21%, and 16.57% for 2, 3, and 4 dimensions, respectively. Performance can be improved by increasing the number of depths in a tree structure, but the distortion reduction rate at 4 dimensions was much higher than was that at 2 dimensions. Therefore, the ratio of preserved information plays a critical role in establishing decision criterion for the tree structure.

Second, we analyzed the effect of the number of dimensions. Increasing the number of dimensions from 2 to 4 yielded distortion reduction rates of 12.75%, 16.06%, 19.22%, and 22.00% after limiting the nodes of tree structure to 256, 512, 1024, and 2048, respectively. These results indicated that the distortion reduction rates for controlling the number of dimensions were higher compared with those for controlling the nodes of the tree structure. Thus, the number of dimensions yielded a greater effect on performance than did the number of depths.

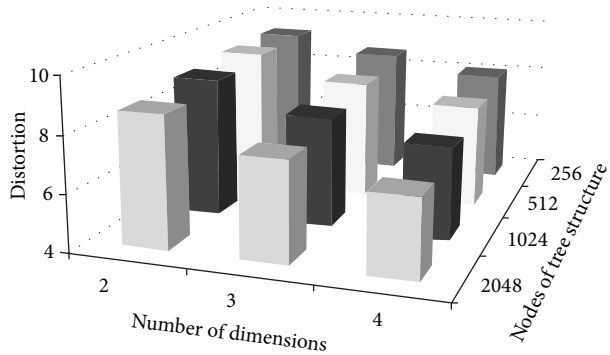

Figure 2: The experimental results of initial codewords in HVQ.

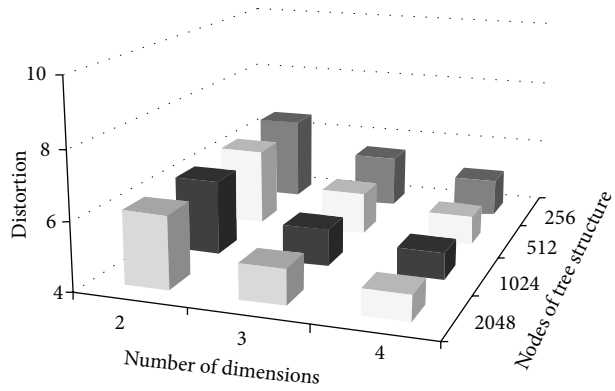

Figure 3: The experimental results of optimal codewords in HVQ with 1 ripple.

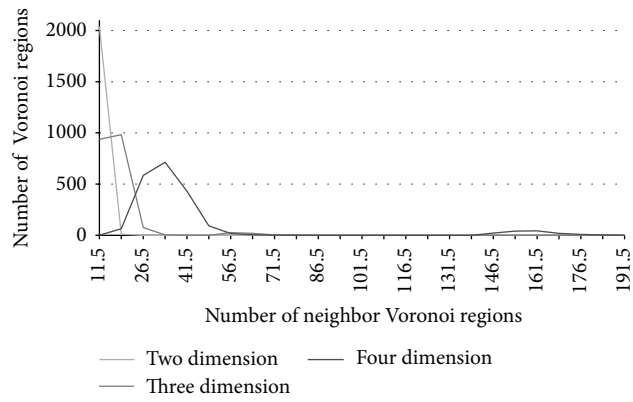

Figure 4: The histograms of number of neighboring Voronoi regions.

Finally, we analyzed the storage and computation requirements for the tree structure. To simplify the analysis, the tree structures were treated as balanced trees, and the number of traversed nodes in the tree structure was treated as $\log_2(L)$, where L is the number of nodes. Hence, the number of traversed nodes was considered 8, 9, 10, and 11 for 256, 512, 1024, and 2048 nodes, respectively. When the number of tree nodes increased from 256 to 2048, the traversed nodes would only increase three nodes. Therefore, the differences in computational times for various numbers of tree nodes can be ignored in many applications. However, the storage requirement necessitates 1792 additional nodes. This serious increase in required storage presents a critical problem in applications with limited memory. When we increased one dimension of tree structure, it needed additional 8 to 11 comparison operations for 256 and 2048 nodes, respectively. This would also increase one additional storage space in each node. However, the additional storage space of increasing the dimensions was smaller than that of increasing number of depths.

To overcome this disadvantage, we proposed enlarging the search space of the Voronoi diagram. The experimental results are shown in Figure 3. The distortion for HVQ with 2 dimensions and 256 tree structure nodes was 6.771. The distortion for HVQ with 4 dimensions and 2048 tree structure nodes was 6.525. These results are similar, allowing the storage requirement to be reduced without degrading performance; thus, HVQ is suitable in applications with limited storage, such as handheld devices and microcontroller units.

3.3. Experimental Results of Searching Optimal Codeword. CB2048 was adopted to examine the characteristics of the Voronoi diagram structure. The number of neighboring Voronoi regions of each Voronoi region was counted when the number of dimensions was 2, 3, and 4. These results are shown as histograms in Figure 4. The average numbers of neighboring Voronoi regions were 7.0, 17.1, and 42.4 for 2, 3, and 4 dimensions, respectively. Approximately 99.46%, 97.61%, and 93.31% of the Voronoi regions at 2, 3, and 4 dimensions exhibited less than 19, 34, and 79 neighbors, respectively. Therefore, the number of nodes in the search space enlarged by the ripple strategy is greatly increased by N^R, where N and R are the numbers of neighboring Voronoi regions and ripples. This increase in computational time is unsuitable for various applications.

To overcome this disadvantage, two approaches were proposed and examined: enlarging the search space based on the optimal neighboring Voronoi regions and retaining parts of the neighboring Voronoi regions to search for the optimal neighboring region. At 4 dimensions, most Voronoi regions had less than 79 neighbors; thus, the neighboring Voronoi regions were divided into two groups. Because the full-search strategy is inappropriate for most applications, HVQ, which enlarges the search space by using an optimal neighboring Voronoi region, was investigated. Retaining half of the neighboring Voronoi regions to search for the optimal neighboring Voronoi region (denoted as HVQ_GS) was then adopted to compare the performance. The maximal number of ripples was set at 3 because this yielded acceptable results.

Figure 5 shows the experimental results of the HVQ, expressed as the distortion and numbers of traversed Voronoi regions. In Figure 5(a), the distortion reduction rates are approximately 70% for the optimal codewords searched using one-ripple Voronoi regions in the Voronoi diagram structure. This indicated that the relations between codewords were correctly modeled using the Voronoi diagram structure. Furthermore, the reduction in distortion when the number of dimensions was increased was greater than it was when the number of ripples was increased. Therefore, the number of dimensions was determined to be the primary factor involved in reducing VQ distortion.

Figure 5(b) shows the computed number of traversed Voronoi regions. Enlarging the search space by one ripple yielded average numbers of traversed Voronoi regions of 6.31 and 37.32 at 2 and 4 dimensions, respectively. Enlarging the search space by three ripples yielded average numbers of traversed Voronoi regions of 13.93 and 77.72 at 2 and 4 dimensions, respectively. The number of traversed Voronoi regions was increased by increasing the number of dimensions; however, this increases the computational complexity.

To reduce the computational complexity without considerably degrading accuracy, HVQ_GS was proposed.

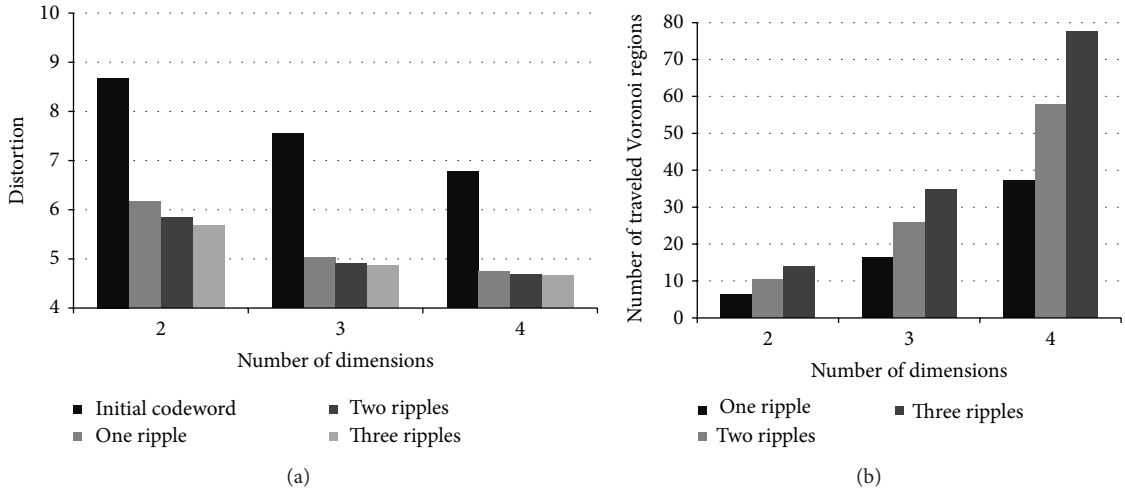

Figure 5: The results of HVQ expressed as (a) distortions and (b) numbers of traversed Voronoi regions.

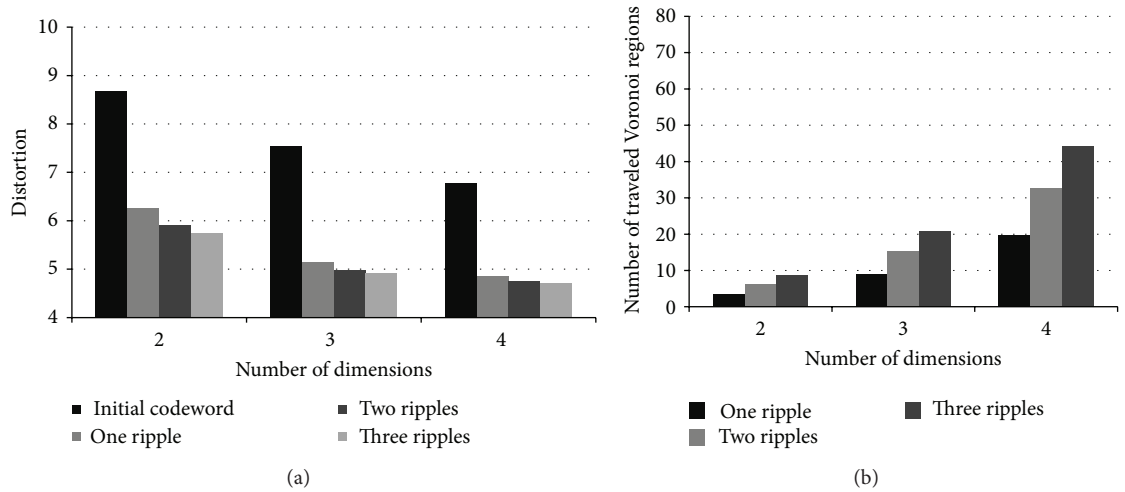

Figure 6: The results of HVQ_GS expressed as (a) distortions and (b) numbers of traversed Voronoi regions.

Figure 6 presents the experimental results. Comparing Figures 6(a) and 5(a) indicates that the accuracy slightly degraded. Comparing Figures 6(b) and 5(b) demonstrates that the proposed approach halved the number of traversed Voronoi regions. Thus, HVQ_GS was determined to be an appropriate solution for reducing computational complexity.

3.4. Comparisons with Other Approaches. Finally, other existing VQ algorithms were compared with the proposed approach. Chang and Wu [24] showed that PVDS outperforms other approaches such as DTPC [20], SCS [21], TSVQ [31], and HOSM [32]; therefore, PVDS was chosen as the baseline approach. Because the proposed approach involved applying a binary search algorithm based on 2D projection and Voronoi vertices to find an initial codeword, the number of dimensions for Voronoi diagram must be less than 3. Thus, HVQ and 2 dimensions, denoted as HVQ(2), were compared with PVDS. Three search strategies, namely, full search, tracking an optimal neighboring Voronoi region based on all neighboring Voronoi regions, and tracking an optimal neighboring Voronoi region based on half the neighboring Voronoi regions, were adopted and denoted as HVQ(4F), HVQ(4), and HVQ_GS(4), respectively.

Figure 7 shows the experimental results. HVQ(2) and PVDS performed similarly at one ripple. As the number of ripples increased, PVDS achieved a more satisfactory performance than did HVQ(2). HVQ(2) only selects optimal neighboring Voronoi regions to determine the optimal codeword; this substantially reduces the search space, easily attaining a local minimum solution. In low-dimensional space, searching all neighboring Voronoi regions could avert this situation. However, searching all neighboring Voronoi regions is unfeasible in high-dimensional space because of the large number of neighboring regions.

Reduction of the search space of a Voronoi diagram structure with high dimensions was investigated. At one ripple, the distortions of HVQ(4F), HVQ(4), and HVQ_GS(4) were

Table 2: The experimental results of full search, HVQ_GS(4), HVQ(4), PVDS, Tree_N, and DP_TSVQ.

	Full search	HVQ_GS(4)	HVQ(4)	PVDS	Tree_N	DP_TSVQ
Traversed nodes	—	54.77	59.83	45.28	50	59.18
Distortion	4.609	4.698	4.686	5.135	4.865	5.420

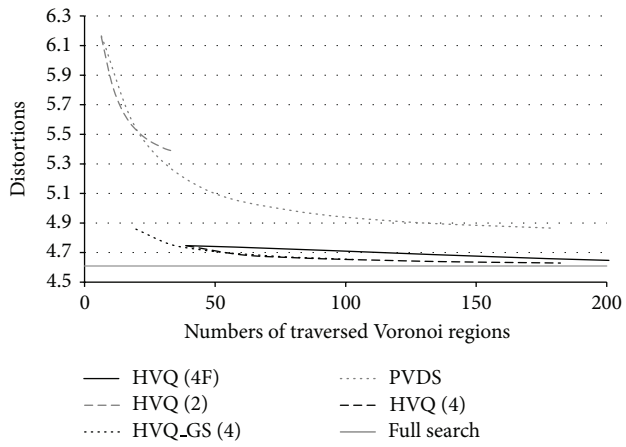

Figure 7: The performances of HVQ and PVDS whose x-axis and y-axis represent number of traversed Voronoi regions and distortion, respectively.

smaller than were those of PVDS and HVQ(2), nearly reaching the optimal performance of PVDS and HVQ(2). Moreover, HVQ increased the search depth to achieve optimal performance more efficiently compared with the full-search approach. Thus, a Voronoi diagram structure that uses high dimensions can precisely enlarge the search space, thereby enhancing performance. Furthermore, a tree structure that exhibits high dimensions can be used to precisely locate an initial codeword. Increasing the number of dimensions improves the performance of both the tree and Voronoi diagram structures. Therefore, HVQ that uses one ripple was a suitable approach to reduce the search space of a Voronoi diagram structure with high dimensions.

To evaluate its performance, the proposed approach was compared with that of tree-structure-based VQ and projection-based VQ. Dynamic path tree-structured VQ [33] was chosen to represent tree-structure-based VQ and was denoted as DP_TSVQ. To compare this approach with enlarging the search space by using the Voronoi diagram structure, a tree structure whose leaf nodes were linked to N-closest codewords was proposed and denoted as Tree_N. PVDS was selected to represent projection-based VQ. To objectively compare these approaches, the number of traversed Voronoi regions or N-nearest codewords was set at approximately 50. Table 2 lists the experimental results. The proposed approach outperformed the other approaches, and increasing the number of dimensions by using the tree structure and the Voronoi diagram enhanced performance. HVQ can balance computational complexity and accuracy when a suitable number of dimensions is selected.

4. Conclusion

In this study, an HVQ combining a tree structure and a Voronoi diagram is proposed to improve VQ efficiency. A tree structure integrated with PCA was successfully applied to reduce the search space in low-dimensional space, allowing an initial codeword to be efficiently located. To improve accuracy, a Voronoi diagram was proposed to precisely model the relations among each codeword, enlarging the search space of the initial codeword. The search space in high-dimensional space was greatly reduced by retaining parts of neighboring Voronoi regions to search for an optimal codeword. According to the experimental results, the proposed approach improved VQ performance and outperformed other approaches. Moreover, it satisfied the computational complexity, storage, and precision requirements when suitable numbers of dimensions were selected. Therefore, the proposed approach is suitable for implementation in handheld devices with limited memory, computational capability, and network bandwidth.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contract NSC 102-2221-E-218-001.

References

[1] Y. Nian, M. He, and J. Wan, "Low-complexity compression algorithm for hyperspectral images based on distributed source coding," *Mathematical Problems in Engineering*, vol. 2013, Article ID 825673, 7 pages, 2013.

[2] C. Lin, Y. Zhao, and C. Zhu, "Two-stage diversity-based multiple description image coding," *IEEE Signal Processing Letters*, vol. 15, pp. 837–840, 2008.

[3] W. Chan and D. Chemla D, "Low-complexity encoding of speech LSF parameters using constrained-storage TSVQ," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 521–524, April 1994.

[4] R. C. Chen, P. Y. Pai, Y. K. Chan, and C. C. Chang, "Lossless image compression based on multiple-tables arithmetic coding," *Mathematical Problems in Engineering*, vol. 2009, Article ID 128317, 13 pages, 2009.

[5] S.-G. Miaou and S.-T. Chen, "A data compression method for long sequence of wireless capsule endoscope images," *Journal of*

Medical and Biological Engineering, vol. 25, no. 3, pp. 107–116, 2005.

[6] W.-J. Chen and H.-M. Chen, "Inverse diamond search algorithm for 3D medical image set compression," *Journal of Medical and Biological Engineering*, vol. 29, no. 5, pp. 266–270, 2009.

[7] M. Djamah and D. O'Shaughnessy, "Fine-granular scalable MELP coder based on embedded vector quantization," in *Proceedings of the 10th Annual Conference of the International Speech Communication Association*, pp. 2603–2606, Brighton, UK, September 2009.

[8] S. R. Tsui, C. T. Huang, and W. J. Wang, "A new adaptive steganographic method based on gradient adjacent prediction and side-match vector quantization," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 4, pp. 215–224, 2013.

[9] C. C. Chang, Y. C. Chou, and C. Y. Lin, "An indicator elimination method for side-match vector quantization," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 4, pp. 233–249, 2013.

[10] H. Chen, B. B. Liu, H. Luo, and Z. M. Lu, "Fast image artistic style learning using twin-codebook vector quantization," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 3, no. 1, pp. 66–70, 2012.

[11] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications Systems*, vol. 28, no. 1, pp. 84–95, 1980.

[12] C. C. Chang and I. C. Lin, "Fast search algorithm for vector quantisation without extra look-up table using declustered subcodebooks," *IEE Proceedings—Vision, Image, and Signal Processing*, vol. 152, no. 5, pp. 513–519, 2005.

[13] L. Torres and J. Huguet, "Improvement on codebook search for vector quantization," *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 208–210, 1994.

[14] H. Park and V. K. Prasanna, "Modular VLSI architectures for real-time full-search-based vector quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 4, pp. 309–317, 1993.

[15] C.-D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Transactions on Communications*, vol. 33, no. 10, pp. 1132–1133, 1985.

[16] C.-H. Lee and L.-H. Chen, "High-speed closest codeword search algorithms for vector quantization," *Signal Processing*, vol. 43, no. 3, pp. 323–331, 1995.

[17] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 5, pp. 562–574, 1980.

[18] M. Djamah and D. O'Shaughnessy, "An efficient tree-structured codebook design for embedded vector quantization," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '10)*, pp. 4686–4689, Dallas, Tex, USA, March 2010.

[19] C.-C. Chang and T.-S. Chen, "New tree-structured vector quantization with closest-coupled multipath searching method," *Optical Engineering*, vol. 36, no. 6, pp. 1713–1720, 1997.

[20] V. Ramasubramanian and K. K. Paliwal, "Voronoi projection-based fast nearest-neighbor search algorithms: box-search and mapping table-based search techniques," *Digital Signal Processing*, vol. 7, no. 4, pp. 260–277, 1997.

[21] S. C. Tai, C. C. Lai, and Y. C. Lin, "Two fast nearest neighbor searching algorithms for image vector quantization," *IEEE Transactions on Communications*, vol. 44, no. 12, pp. 1623–1628, 1996.

[22] J. Cui, S. Zhou, and J. Sun, "Efficient high-dimensional indexing by sorting principal component," *Pattern Recognition Letters*, vol. 28, no. 16, pp. 2412–2418, 2007.

[23] J. Cui, Z. An, Y. Guo, and S. Zhou, "Efficient nearest neighbor query based on extended B+-tree in high-dimensional space," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1740–1748, 2010.

[24] C.-C. Chang and W.-C. Wu, "Fast planar-oriented ripple search algorithm for hyperspace VQ codebook," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1538–1547, 2007.

[25] R. C. T. Lee, S. S. Tseng, R. C. Chang, and Y. T. Tsai, *Introduction to the Design and Analysis of Algorithms: A Strategic Approach*, McGraw-Hill, Singapore, 2005.

[26] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley & Son, New York, NY, USA, 1992.

[27] H. Edelsbrunner and R. Seidel, "Voronoĭ diagrams and arrangements," *Discrete & Computational Geometry*, vol. 1, no. 1, pp. 25–44, 1986.

[28] S. Berchtold, D. A. Keim, and H. P. Kriegel, "The x-tree: an index structure for high-dimensional data," in *Proceedings of the 22nd International Conference on Very Large Data Bases*, pp. 28–39, San Francisco, Calif, USA, 1996.

[29] Y. J. Chen and J. L. Wu, "A novel speech enhancement using forward-backward minima-controlled recursive averaging," *Journal of the Chinese Institute of Engineers*, vol. 37, no. 3, pp. 395–406, 2013.

[30] Y.-J. Chen, "Identification of articulation error patterns using a novel dependence network," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 11, pp. 3061–3068, 2011.

[31] R. M. Gray and Y. Linde, "Vector quantization and predictive quantizers for gauss-Markov sources," *IEEE Transactions on Communications*, vol. 30, no. 2, pp. 381–389, 1982.

[32] S.-J. Wang and C.-H. Yang, "Hierarchy-oriented searching algorithms using alternative duplicate codewords for vector quantization mechanism," *Applied Mathematics and Computation*, vol. 162, no. 2, pp. 559–576, 2005.

[33] C. C. Chang, F. J. Shiue, and T. S. Chen, "Tree structured vector quantization with dynamic path search," in *Proceedings of the International Workshop on Multimedia Network Systems*, pp. 536–541, Aizu, Japan, September 1999.