中華民國八十八年全國計算機會議

Image Coding by Using Smooth Side-Match Classified Vector Quantizer with Variable Block Size

Shiueng Bien Yang and Lin Yu Tseng*
Department of Applied Mathematics
National Chung Hsing University
Taichung, Taiwan 402, R.O.C.
Tel : 886-4-2874020, Fax : 886-4-2873028
E-mail: lytseng@amath.nchu.edu.tw

Abstract

Although the side-match vector quantizer (SMVQ) is efficient in reducing the bit rate, the image coding quality is in general degenerates. The reason is that if the gray level transition across the boundaries between the neighboring blocks is increasing or decreasing, SMVQ may not encode the block well. In this study, we proposed a smooth side-match method to select the state codebook according to the smoothness of the gray levels between the neighboring blocks. This method achieves the higher PSNR and the better visual perception quality than SMVQ does while the bit rate is the same. Moreover, a genetic clustering algorithm that can automatically find the proper number of clusters is proposed to design the codebooks because the well-known algorithm LBG has the shortcoming of requiring the user to supply it with the number of clusters. Therefore, the proposed smooth side-match classified vector quantizer (SSM-CVQ) is the combination of three techniques: classified vector quantization, variable block size segmentation and the smooth side-match method. As indicated by the experimental results, SSM-CVQ has the higher coding quality and the lower bit rate than other methods have. The Lena image can be coded by SSM-CVQ with 0.172 bpp and 32.49 dB.

Key words: *Smooth side-match classified vector quantizer, genetic clustering algorithm, image coding.*

* Author to whom correspondence should be sent.

1. Introduction

Vector quantization (VQ) is a useful technique for image data compression. Since the distortion measure, such as mean square error (MSE), is not good at preserving the property of edges, the edge degradation has been a problem in VQ. In [1], [2] and [3], DCT coefficients were used as edge oriented features. The performance of CVQ in the DCT domain is better than that of CVQ in the spatial domain. In [2], the variable block-size coding method was proposed to achieve the higher coding quality and the lower bit rate. In this study, the proposed SSM-CVQ consists of three codebooks for low-detail 16×16, 8×8 and 4×4 blocks and twenty-eight codebooks for high-detail 4×4 blocks. In all conventional CVQ's, the designer gives the number of codebooks before designing CVQ. For example, nine edge classifiers and sixteen edge classifiers are used to classify the high-detail blocks in [2] and [4] respectively. However, these numbers are in general not sufficient for the accurate edge classification. In real images, edges are more complicated than the synthetic ones. Since the genetic algorithm is good at searching ([5], [6]), the genetic clustering algorithm had been applied to the vector quantization problem ([7]-[9]). In SSM-CVQ, a genetic clustering algorithm is proposed to find the number of edge classifiers for high-detail blocks and then the genetic clustering algorithm is applied again to find the codebook for each edge classifier. This genetic clustering algorithm will search for a proper number of cluster centers and do the clustering simultaneously so the user need not bother

the number of clusters at all. The clusters obtained will be more natural in accordance to the characteristics of the data set. Although the LBG algorithm [10] had been widely used in the designing of the codebook, it suffers the drawback that the user must provide the number of clusters in advance while the user in general has no idea about how many clusters there should be in the data set.

SMVQ [11] is a well-known class of finite-state vector quantizers (FSVQ) and several variants of SMVQ had been proposed (e.g. [12], [13]). In [4], the combination of the CVQ and SMVQ is proposed and it has better coding quality than SMVQ. However, the above SMVQ's use the fixed block size and thus the bit rate can not be further reduced. Moreover, SMVQ selects the codewords such that the gray levels of pixels right across the boundaries of the neighboring blocks are as near as possible. However, if the gray levels of pixels across the boundaries of the neighboring blocks is increasing or decreasing, SMVQ may fail to select the codeword nearest to the currently encoded block. In this study, the smooth side-match method with variable block size is proposed and it achieves the better coding quality and visual perception quality than SMVQ does at the same bit rate. The smooth side-match method selects the state codebook according to the smoothness of the gray levels of pixels between the neighboring blocks. Therefore, SSM-CVQ, a combining the CVQ, the variable block size segmentation and the smooth side-match method, improves the coding quality, the visual perception quality and the bit rate.

The remaining parts of this paper are organized as follows. The genetic clustering algorithm is described in Section 2. The design of smooth side-match method is described in Section 3. The design of SSM-CVQ is stated in Section 4. Experimental results are given in Section 5 and the conclusions are described in Section 6.

2. The Genetic Clustering Algorithm

The proposed algorithm CLUSTERING consists of two stages. The first stage is a pair nearest neighbor (PNN) algorithm [14]. The algorithm starts with the whole training data set and iteratively reduces it to smaller ones by combining the two closest vectors until the proper size of data set m is reached. Therefore, for the second stage, the initial training data set contains m small sets $B_1, B_2, \ldots, B_m$. Let the center of each set B_i be denoted by V_i for $1 \le i \le m$. Clearly, m is less than the size of the original data set. The objective of using PNN algorithm in the first stage is to reduce the computation time required in the second stage. Therefore, the clustering algorithm can process the large data set efficiently.

The sets $B_1, B_2, \ldots, B_m$ obtained in the first stage are taken as the initial clusters in the second stage. The second stage is a genetic algorithm, which will merge some of these B_i's if they are close enough to one another. The genetic algorithm consists of an initialization step and the iterative generations with three phases in each generation. They are described in the following.

Initialization step:

A population of N strings is randomly generated. The length of each string is m, which is the number of the sets obtained in the first stage. N strings are generated in such a way that the numbers of 1's in the strings almost uniformly distributes within [1,m]. Each string represents a subset of $\{B_1, B_2, \ldots, B_m\}$. If B_i is in this subset, the *i*th position of the string will be 1; otherwise, it will be 0. Each B_i in the subset is used as a seed to generate a cluster.

Before describing three phases, let us first describe how to generate a clustering from the seeds. Let $T=\{ T_1, T_2, \ldots, T_s \}$ be the subset corresponding to a string. Let initial clusters C_i be T_i for $i=1, 2, \ldots, s$. Let initial centers S_i of clusters C_i be V_i for $i=1, 2, \ldots, s$ and the size of cluster C_i is defined as $|C_i| = |T_i|$ for $i=1, 2, \ldots, s$, where $|T_i|$ denotes the number of objects belonging to the set T_i.

The generation of the clusters proceeds as follows. The B_i's in $\{ B_1, B_2, \ldots, B_m \}$-T are taken one by one and the distance between the center of the taken B_i and each center S_j is calculated. Then we have

$$B_i \subset C_j \quad \text{if} \quad \|V_i - S_j\| \le \|V_i - S_k\| \quad \text{for}$$

$1 \le k \le s$ and $k \ne j$.

If B_i is classified as in the cluster C_j, the center S_j and the size of the cluster C_j are recomputed as follows when B_i is added to C_j.

$$S_j = \frac{S_j * |C_j| + V_i * |B_i|}{|C_j| + |B_i|}$$

$$|C_j| = |C_j| + |B_i|.$$

After B_i's in $\{ B_1, B_2, \ldots, B_m \}$-T all have been considered, we will obtain the cluster C_j with center S_j generated by the seed T_j for $j=1, 2 \ldots, s$. We define $\{C_1, C_2, \ldots, C_s\}$ as the set of clusters generated by this string.

Reproduction phase:

Let C_i be one of the clusters generated by string R. In the following, we define D_{intra} to represent the intra-distance in the cluster C_i and D_{inter} to represent the inter-distance between this cluster C_i and the set of all other clusters.

$$D_{intra}(C_i) = \sum_{B_k \subset C_i} \|V_k - S_i\| * |B_k|$$

$$D_{inter}(C_i) = \sum_{B_k \subset C_i} (\min_{i \ne j} \|V_k - S_j\|) * |B_k|$$

where the summation is over all B_k's that are in the cluster C_i. Then we can define the fitness function of a string R as follows.

$$\text{Fitness(R)} = D_{inter}(C_i) * w - D_{intra}(C_i)$$

where w is a weight. If the value of w is small, we emphasize the importance of $D_{intra}(C_i)$. This tends to produce more clusters and each cluster tends to be compact. If the value of w is chosen to be large, we emphasize the importance of $D_{inter}(C_i)$. This tends to produce fewer clusters and each cluster tends to be loose. If R contains only 0's, Fitness(R) is defined to be 0. If R contains exactly one 1, $D_{inter}(C_i)$ is defined to be 0. After the calculation of fitness for each string in the population, the reproduction operator is implemented by using a roulette wheel with slots sized according to fitness.

Crossover phase:

If a pair of strings R and Q are chosen for applying the crossover operator, two random numbers p and q in [1,m] are generated to decide which pieces of the strings are to be interchanged. Suppose p<q, the bits from position p to position q of string R will be interchanged with those bits that are in the same position of string Q. For each chosen pair of strings, the crossover operator is done with probability p_c.

Mutation phase:

In the mutation phase, bits of the strings in the population will be chosen with probability p_m. Each chosen bit will be changed from 0 to 1 or from 1 to 0.

The user may specify the number of generations that he or she wants the genetic algorithm to run. The genetic algorithm will run this number of generations and retain the string with the best fitness. The user may specify the value of w used in calculating fitness function in order to emphasize on either the compactness of clusters or the enlargement of the distances among clusters.

3. The Design of Smooth Side-Match Method

SMVQ tries to make the gray levels of pixels right across the boundaries of the neighboring blocks as near as possible. In SMVQ, the selection algorithm selects a subset of the codewords with the smallest side-match distortions in the supercodebook as the state codebook. However, if the gray levels of pixels across the boundaries between the

neighboring blocks is increasing or decreasing, SMVQ usually does not encode a block well. In this study, a smooth side-match method is proposed to select the state codebook according to the smoothness of the gray levels of pixels between the neighboring blocks. The conventional side-match method is described in Subsection 3.1 and the smooth side-match method is depicted in Subsection 3.2.

3.1. The conventional side-match method

Let x be the m×n block to be encoded. Let the state space S be defined as S={u×λ: u is the codeword for some x's upper block and λ is the codeword for some x's left block}. The vertical correlation of state u and the horizontal correlation of state λ define the state s of x. The vertical side-match distortion of a codeword y in the supercodebook is defined as

$$vd(y)=\sum_{i=1}^{n}(y_{1,i}-u_{m,i})^2$$

and the horizontal side-match distortion is defined as

$$hd(y)=\sum_{j=1}^{m}(y_{j,1}-\lambda_{j,n})^2$$

Then, the side-match distortion corresponding to the codeword y is defined as

$$D(y)=vd(y)+hd(y).$$

The selection algorithm for the state s selects N_s codewords in the supercodebook with the smallest N_s side-match distortions $D(y_i)$'s for $i=1,\ldots, N_s$ as x's state codebook. SMVQ sorts the selected codewords in the state codebook according to the side-match distortions of codewords. Then, SMVQ picks the codeword nearest to x from the state codebook to replace x and the index of this codeword is regarded as the code of x.

3.2. The smooth side-match method with variable block size

The blocks coded by the conventional SMVQ are of size 4×4. SMVQ selects the first row of blocks and the first column of blocks in an image as its basic blocks. These basic blocks are directly encoded by the full search vector quantization using the supercodebook. In smooth side-match method, we select the basic blocks as follows. The image of size $m \times n$ is first divided into the blocks of size 16×16 and the diagonal blocks are selected for further deciding whether these blocks are basic blocks or not. Let B be a diagonal block of size 16×16. If B is a low-detail block, then B is defined as a basic block. Otherwise, B is divided into four blocks of size 8×8 as follows.

$$B=\begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}$$

where b_i for $1 \le i \le 4$ indicate the blocks of size 8×8 and the diagonal blocks b_1 and b_4 are selected for further deciding if they are basic blocks or not. If b_i, for i=1 or 4, is a low-detail block, then the block is defined as a basic block. Otherwise, the block is further divided into four blocks of size 4×4 as follows.

$$b_i=\begin{bmatrix} b_i^1 & b_i^2 \\ b_i^3 & b_i^4 \end{bmatrix}$$

where b_i^j for $1 \le j \le 4$ are blocks of size 4×4. Finally, the blocks b_i^1 and b_i^4 are defined as the basic blocks. All the basic blocks with different sizes are directly encoded with the supercodebooks and the other blocks are encoded by SSM-CVQ. Figure 1 shows an example of the basic blocks in block B. In [13], the number of the basic blocks in an image is about one half of that of SMVQ. In SSM-CVQ, the number of the basic blocks required in an image is usually less than 30% of that of SMVQ. The bit rate of SSM-CVQ is therefore further reduced.

Figure 2 shows an example of the gray levels of the pixels in the encoded block x and it's upper block u. The side-match distortion selects the codeword y such that the gray levels of $y_{1,1}$ and $u_{m,1}$ are as near as possible. However, the gray level of $y_{1,1}$ thus selected may deviate from the gray level of $x_{1,1}$ significantly as revealed by the example shown in Figure 2. In real images, the changing of gray levels is in general smooth among the neighboring pixels.

We use this property to define the smooth side-match distortion. We first define the variation, var(u, v), between the pixels u and v as follows.

var(u, v)=u-v.

Since the diagonal blocks are selected as the basic blocks and they are encoded first in SSMVQ, the encoded image is therefore divided into two parts, the upper triangular region and the lower triangular region. In the upper triangular region, the neighboring blocks of the currently encoded block are defined to be its left block and its lower block. An example is shown in Figure 3(a). The vertical smooth side-match distortion is defined as

Upper_vd(y)=

$$\sum_{i=1}^{n}\left|\frac{(\mathrm{var}(d_{2,i},d_{1,i})+\mathrm{var}(y_{m,i},y_{m-1,i}))}{2}-\mathrm{var}(d_{1,i},y_{m,i})\right|$$

and the horizontal smooth side-match distortion is defined as

Upper_hd(y)=

$$\sum_{j=1}^{m}\left|\frac{(\mathrm{var}(\lambda_{j,n-1},\lambda_{j,n})+\mathrm{var}(y_{j,1},y_{j,2}))}{2}-\mathrm{var}(\lambda_{j,n},y_{j,1})\right|$$

Then, the smooth side-match distortion of the codeword y is defined as

D(y)=Upper_vd(y)+Upper_hd(y).

In the lower triangular region, the neighboring blocks of each encoded block are defined to be its right block and its upper block. An example is depicted in Figure 3(b). The vertical smooth side-match distortion is defined as

Lower_vd(y)=

$$\sum_{i=1}^{n}\left|\frac{(\mathrm{var}(u_{m-1,i},u_{m,i})+\mathrm{var}(y_{1,i},y_{2,i}))}{2}-\mathrm{var}(u_{m,i},y_{1,i})\right|$$

and the horizontal smooth side-match distortion is defined as

Lower_hd(y)=

$$\sum_{j=1}^{m}\left|\frac{(\mathrm{var}(r_{j,2},r_{j,1})+\mathrm{var}(y_{j,n},y_{j,n-1}))}{2}-\mathrm{var}(r_{j,1},y_{j,n})\right|$$

Then, the smooth side-match distortion of the codeword y is defined as

D(y)=Lower_vd(y)+Lower_hd(y).

The selection algorithm selects N_s codewords from the supercodebook as x's state codebook. These codewords y_i's have the smallest smooth side-match distortions $D(y_i)$'s for $i=1,\ldots, N_s$.

4. The Design of SSM-CVQ

In SSM-CVQ, three codebooks corresponding to the block size 16×16, 8×8 and 4×4 are built for low-detail blocks and twenty-eight codebooks corresponding to block size 4×4 are built for high-detail blocks. The advantage of SSM-CVQ is that the bit rate for low-detail blocks can be reduced and the coding quality for high-detail blocks can be improved. The bit rate for low-detail blocks can be reduced because they are encoded by smaller sized state codebooks and also the block size is selected as large as possible. The coding quality of high-detail blocks can be improved because these blocks are classified into more edge classifiers and larger sized state codebooks are used. In SSM-CVQ, the coding quality is also improved by replacing the LBG algoroithm with the proposed algorithm CLUSTERING and the visual perception quality is enhanced by using the smooth side-match method.

In the following, Subsection 4.1 describes the segmentation strategy. The designs of the codebooks for low-detail blocks and for high-detail blocks are described in Subsection 4.2 and Subsection 4.3, respectively. Subsection 4.4 describes the coding method in SSM-CVQ.

4.1. The segmentation strategy

Each training block of size 16×16 is first classified as a low-detail block or a high-detail block. The variance of each block of size 16×16 is calculated. If the variance of the block is less than the given threshold, this block is classified as a low-detail block. Otherwise, this block is divided into four blocks of size 8×8. This splitting process is continued until the blocks of size 4×4 are reached. We

use the quadtree [15] to address the blocks of different sizes. That is, if the block is divided into smaller blocks, the quadtree code is "1". Otherwise, the quadtree code is "0". When the size of the block is 4 × 4, if this block is a high-detail block, the quadtree code is "1". Otherwise, the quadtree code is "0". Figure 4(a) shows an example of a 16 × 16 block and Figure 4(b) depicts the corresponding quadtree. Figure 4(c) describes the quadtree code for this block.

4.2. The design of codebooks for low-detail blocks

In the design of the low-detail codebooks, a few DCT coefficients are enough to capture the property of the block. Figure 5 shows an example of the DCT coefficients with the block size m×n. For low-detail blocks, three codebooks for three different sizes of blocks are designed separately. Since the gray levels of pixels in the low-detail block do not vary much, the higher frequency coefficients are not significant and only a few DCT coefficients are required to represent the block. The use of fewer coefficients does reduce the complexity of designing the codebooks. The selected coefficients for a block of size m×m are c(i, j)'s where $0 \le i \le m/2$ and $0 \le j \le (m/2)-i$. These DCT coefficients are viewed as a vector for representing the block. Use the algorithm CLUSTERING to cluster these vectors and then, the inverse DCT (i.e. IDCT) transforms the center of each cluster into a spatial vector. These spatial vectors constitute the codewords of the codebook.

4.3. The design of codebooks for high-detail blocks

In the design of the high-detail codebooks, the size of all training blocks is 4 × 4. Six DCT coefficients were used for edge classification in [2]. c(0, i)'s represent the horizontal features and c(i, 0)'s represent the vertical features for $1 \le i \le 3$. These six coefficients viewed as a vector are also used for edge classification in the proposed SSM-CVQ. In SSM-CVQ, instead of attempting to predefine the number of edge-oriented classifiers, we apply the algorithm CLUSTERING to the whole high-detail training blocks and the number of clusters is automatically obtained. Let there be q clusters for high-detail blocks. The center of each cluster is regarded as an edge classifier. That is, there are q codebooks for high-detail blocks. The algorithm CLUSTERING is then applied again to build the codebook for each edge classifier. We use ten DCT coefficients, c(i, j) for $0 \le i \le 3$ and $0 \le j \le 3-i$, as features to do the clustering. Then, the center of each cluster is transformed into a spatial vector by IDCT and this vector represents a codeword in the codebook. Therefore, when a high-detail block is encoded, this block is first classified into one of the edge classifiers and then encoded by a codeword in this edge classifier's codebook.

4.4. The coding method in SSM-CVQ

In SSM-CVQ, there are three codebooks for low-detail blocks and q codebooks for high-detail blocks. Each codebook is regarded as a supercodebook. Hence, there are q+3 supercodebooks in SSM-CVQ. When we want to encode a low-detail block, SSM-CVQ selects the supercodebook according to the block size and then, this block is encoded by the smooth side-match method using a smaller sized state codebook. So, we may use fewer bits to encode the low-detail regions that occupy large areas of an image and the bit rate is therefore reduced. When we want to encode a high-detail block, SSM-CVQ selects the supercodebook according to the characteristics of edges in this block. This high-detail block is then encoded by the smooth side-match method using a larger sized state codebook. The coding quality of the high-detail region is improved because the high-detail block is further classified into q classes and the use of larger sized state codebooks.

5. Experimental Results

In our experiments, five images, each of size 512 × 512, were used as the training images. The parameters used in the second stage of CLUSTERING are described as follows. The population size was 50, the crossover rate was 80% and the mutation rate was 5%. 200 generations were

run and the best solution was retained. The 486 personal computer was used in the experiments. At first, we searched the number of edge classifiers for high-detail blocks. The various values of w's were separately used in the second stage of the algorithm CLUSTERING. If w was chosen to be large, CLUSTERING produced fewer clusters and the average distortion tended to be large. The center of each cluster was regarded as an edge classifier and edge classifier indices were coded by the Huffman code based on the sizes of clusters. In Figure 6, several values of w, namely, 1.5, 2, 2.5, 3, 3.5, 4 and 4.5, were used separately in algorithm CLUSTERING and 98, 72, 58, 37, 28, 16 and 10 clusters were obtained, respectively. As indicated by Figure 6, when the values of w were larger than 3.5, the average distortion increased dramatically. Therefore, twenty-eight edge classifiers were selected for designing high-detail codebooks.

"Lena" image and "F-16" image that were not in the training set were used to test the performance of SSM-CVQ. In SSM-CVQ, the sizes of three low-detail codebooks were 68, 72, 78 for 16×16, 8×8 and 4×4 blocks respectively and the average codebook size for high-detail blocks was 128. The state codebook size was defined as 16 for low-detail blocks and 32 for high-detail blocks. Table 1 and Table 2 lists the coding quality and the bit rate of these two images coded by SSM-CVQ. In Table 3, there are several combinations of the various techniques used in designing the CVQ's. The clustering algorithm (PNN+LBG) indicates the combination of PNN and LBG algorithms ([14], [16]). To compare the various combinations in Table 3, each CVQ was designed with the same structure. The CLUSTERING algorithm was first used to find the number of edge classifiers and designed each codebook, then the LBG algorithm and the (PNN+LBG) algorithm were applied separately to obtain the same number of edge classifiers and the same size for each codebook. As indicated in Table 3, using of the CLUSTERING algorithm has better coding quality than the LBG algorithm or the (PNN+LBG) algorithm. The reason is that the algorithm CLUSTERING in general finds a better clustering than the LBG algorithm and the (PNN+LBG) algorithm do. In Table 3, the smooth side-match method also has better coding quality than the side-match method. Figure 7 shows a magnified portion of the image "Lena". The image encoded by the smooth side-match method has better visual perception quality than that encoded by the side-match method when the bit rate is the same. This is because the smooth side-match method is based on the smoothness of the gray level transitions across the boundaries of the neighboring blocks. Table 4 shows the average training time of designing the codebooks by the three clustering algorithms. The algorithm CLUSTERING has better performance than the LBG algorithm and the (PNN+LBG) algorithm. The entropy coding was used to reduce the bit rate in [4] and [11], we also used the entropy coding for the indices of state codebooks to further reduce the bit rate. Table 5 lists a comparison of the performances of several coding methods. The SSM-CVQ outperforms the other methods in both the PSNR and the bit rate.

6. Conclusions

In SSM-CVQ, the algorithm CLUSTERING is applied to find the number of edge classifiers for high-detail blocks and to design the codebooks. The algorithm CLUSTERING searches for a proper number of cluster centers and does the clustering simultaneously. As indicated by the experimental results, the coding quality is improved in SSM-CVQ because the algorithm CLUSTERING can obtain better clustering than the LBG algorithm. The application of the smooth side-match method also achieves the higher coding quality and the better visual perception quality than the conventional side-match method when the bit rate is the same. We apply the smooth side-match method to SSM-CVQ to lower the bit rate. Moreover, the low-detail blocks are encoded by a smaller sized state codebook and the block size is selected

as large as possible, hence, the bit rate for low-detail blocks can be further reduced. Classifying high-detail blocks into more edge classifiers and using the larger sized state codebooks also improve the coding quality of the high-detail blocks. The experimental results have shown that SSM-CVQ has the higher coding quality, the better visual perception quality and the lower bit rate.

References

[1] D. S. Kim and S. U. Lee, "Image vector quantizer based on a classification in the DCT domain," *IEEE Trans. Commun.*, vol. 39, no. 4, pp. 549-556, 1991.

[2] M. H. Lee and G. Crebbin, "Classified vector quantization with variable block-size DCT models," *IEE Proc.-Vis. Image Signal Process.*, vol. 141, no. 1, pp. 39-48, 1994.

[3] J. Vaisey and A. Gersho, "Image compression with variable block size segmentation," *IEEE Trans. Signal Process.*, vol. 40, no. 8, pp. 2040-2060, 1992.

[4] R. F. Chang and W. M. Chen, "Adaptive edge-based side-match finite-state classified vector quantization with quadtree map," *IEEE Trans. Image Processing*, vol. 5, no. 2, pp. 378-383, 1996.

[5] M. Srinivas and M. Patnaik, "Genetic algorithm-A survey," *IEEE Computer*, vol.27, no.6, pp.17-26, 1994.

[6] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[7] L. Y. Tseng and S. B. Yang, "Genetic algorithms for clustering, feature selection and classification," *Proc. of IEEE International Conference on Neural Networks*, 1997.

[8] L. Y. Tseng and S. B. Yang, "A general-tree-structured vector quantizer for image progressive coding," *Proc. of Eusipco-98 Ninth European Signal Processing Conference*, pp. 493-496, 1998.

[9] L. Y. Tseng and S. B. Yang, "A genetic approach to the design of general-tree-structured vector quantizers for speech coding," *Proc. of IEEE International Conference on Acoustic, Speech and Signal Processing*, pp. 69-72, 1998.

[10] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.* vol. 28, no. 1, pp. 84-95, 1980.

[11] T. Kim, "Side match and overlap match vector quantizers for images," *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 170-185, 1992.

[12] R. F. Chang and W. T. Chen, "Image coding using variable-rate side-match finite-state vector quantization," *IEEE Trans. Image Processing*, vol. 2, no. 1, pp. 104-108, 1993.

[13] T. S. Chen and C. C. Chang, "A new image coding algorithm using variable-rate side-match finite-state vector quantization," *IEEE Trans. Image Processing*, vol. 6, no. 8, pp. 1185-1187, 1997.

[14] W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acous., Speech and Signal Processing*, vol. 37, no. 10, pp. 1568-1575, 1989.

[15] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys*, vol. 16, pp. 188-216, 1984.

[16] J. H. Li and N. Ling, "A novel VQ codebook design technique," *IEEE. Trans. Consumer Electronics*, vol. 43, no. 4, pp. 1206-1212, 1997.

[17] J. W. Kim and S. U. Lee, "A transform domain classified vector quantizer for image coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 2, pp. 3-14, 1992.

Table 1. The PSNR's by SSM-CVQ.

Class region	PSNR (dB)	
	Lena	F-16
Low-detail 16×16	35.32	34.47
Low-detail 8×8	33.48	33.02
Low-detail 4×4	32.58	32.13
High-detail 4×4	28.01	27.72
Overall	32.49	32.08

Table 2. The bit rate by SSM-CVQ.

Class Region	Bit-rate (bpp)	
	Lena	F-16
Quadtree Code	0.029	0.035
VQ Index	0.144	0.161
Edge classifier Index	0.023	0.028
Overall	0.196	0.224

Table 3. The comparison of the coding qualities by the various combinations.

Method	Lena	F-16
	PSNR (dB)	PSNR (dB)
(1) LBG+Side-Match	31.61	31.24
(2) LBG+Smooth Side-Match	31.98	31.56
(3) (PNN+LBG)+Side-Match	31.67	31.28
(4) (PNN+LBG)+Smooth Side-Match	32.01	31.70
(5) CLUSTERING+Side-Match	32.14	31.75
(6) CLUSTERING+Smooth Side-Match	32.49	32.08

Table 4. The average designing time for each codebook by three methods.

Algorithm	Ave. Training Time (sec)
LBG	3712
PNN+LBG	3191
CLUSTERING	3028

Table 5. The comparison of the coding qualities of the "Lena" image by the various coding methods.

Technique	PSNR (dB)	Bit-rate (bpp)
NewSMVQ [13]	30.97	0.340
QT-CVQ [2]	32.05	0.309
Classified FSVQ [4]	30.44	0.270
DCT-CVQ[17]	31.41	0.32
JPEG	32.01	0.268
SSM-CVQ (no entropy coding)	32.49	0.196
SSM-CVQ (entropy coding)	32.49	0.172

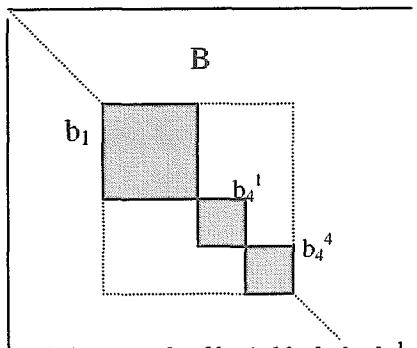


Figure 1. An example of basic blocks b_1, b_4^1 and b_4^4 in B.

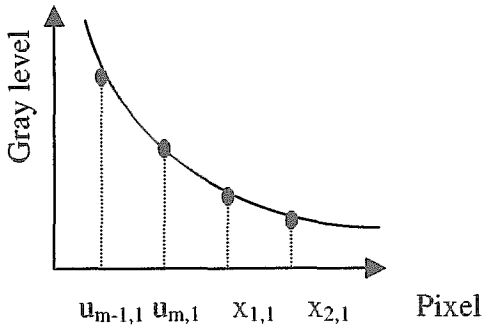


Figure 2. The neighboring pixels $u_{m-1,1}$, $u_{m,1}$, $x_{1,1}$ and $x_{2,1}$ in the encoded block x and its upper block u.

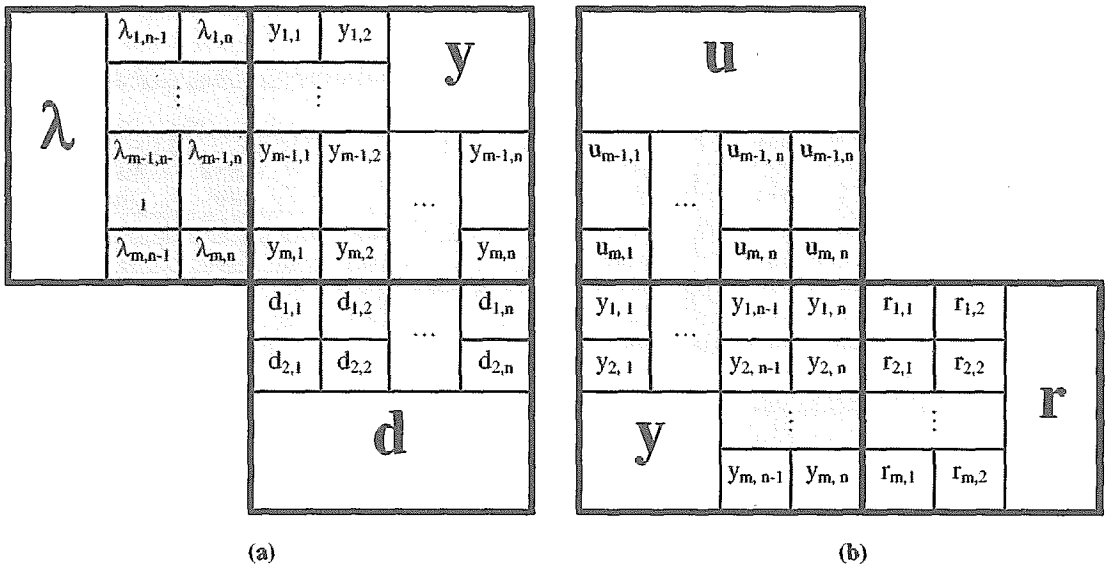
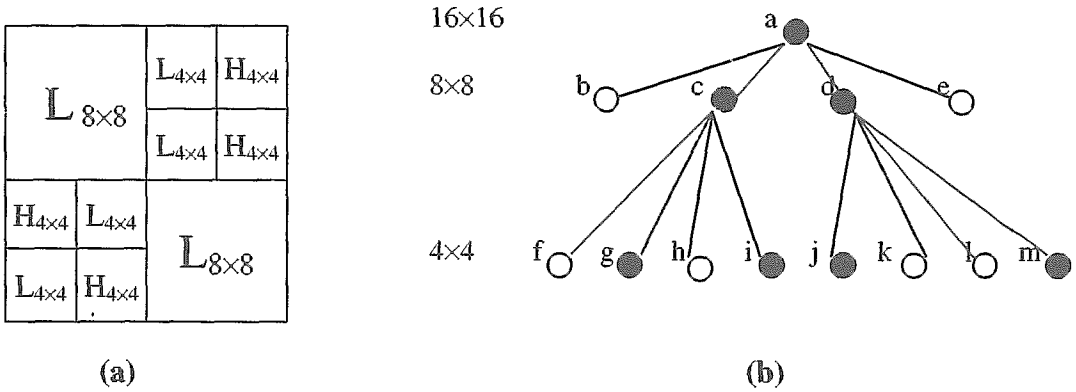


Figure 3. The neighboring blocks used in SSMVQ.



a b c d e f g h i j k l m

quadtree code= 1 0 1 1 0 0 1 0 1 1 0 0 1

(c)

Figure 4. An example of the quadtree for a block of size 16×16.

c(0, 0)	c(0, 1)	.....	c(0, n-1)
c(1, 0)	c(1, 1)	.....	c(1, n-1)
⋮	⋮	⋱	⋮
c(m-1, 0)	c(m-1, 1)	.....	c(m-1,n-1)

Figure 5. The DCT coefficients for the block of size m×n.

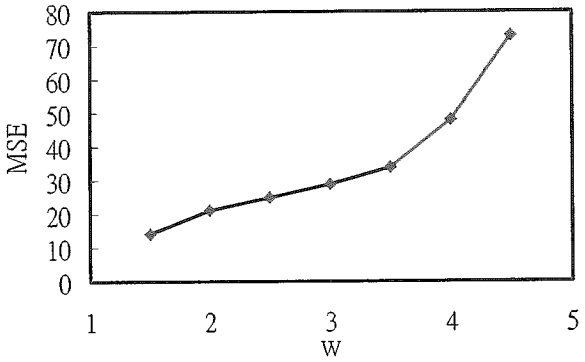
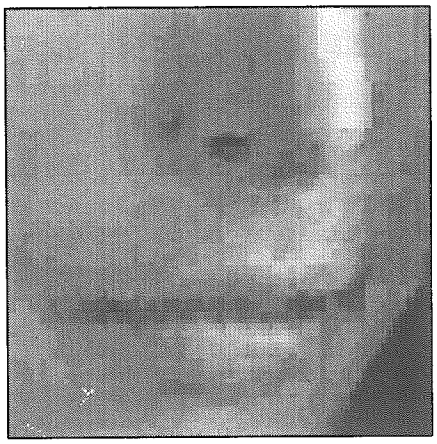


Figure 6. The relationship between MSE and the values of parameter w.

(a) The image encoded by the smooth side-match method.



(b) The image encoded by the side-match method.

(c) The original image.

Figure 7. The reconstructed images at 0.172 bpp.