A Partial Search Mechanism for VQ-based Codebook Match

Cheng-Hsing Yang and Shiuh-Jeng Wang*
Department of Information Management
Kun-Shan University of Technology
* Department of Information Management
Central Police University
Taoyuan, Taiwan 333

Abstract

In this paper, a mechanism for accelerating the search on VQ-based codebook match is proposed. With our approach, all pixel blocks of vector representation in an image picture could be encoded efficiently into their corresponding indices, respectively, associated the closest codeword in the pre-generated codebook. The technique adopted in our scheme is inspired from the concept of space partition to the initial codebook, in the manner that the search range for the image block is reduced significantly. There is a key-codebook generated from the given codebook in system initialization comprised of numerous key-codewords, its size is smaller than the original codebook. In our scheme, the input image block is first directed to look for the closest key-codeword in the key-codebook. After that, a dominated set associated with the closest key-codeword is fixed. The purposed codeword closest corresponding to the input image block in the specified dominated set is ultimately picked up. In such a way, the time to obtain the exactly closest codeword is then shortened when comparing the traditional full VQ codeword search. Moreover, we propose a skill of spreading the dominated area in order to attain the most precise hit radio estimation for the closest codeword. Accordingly, the advantages we achieve are at least two and half times faster than that of full search in VQ implementation, as observed from the experiments. Besides, the mechanism we propose is also compatible with the search algorithms explored in past relevant literature. Thereby, our proposed scheme is feasible and extendable in VQ codeword match.

Keywords: VQ-based search, partition, distortion, image compression

---

*Correspondence addressee: Shiuh-Jeng Wang ,Dept. of Information Management, Central Police University, Taoyuan, Taiwan 333
E-mail: sjwang@sun4.cpu.edu.tw Tel:+886-3-3282321 Ext. 4807
Fax: +886-3-3272038

I. Introduction

With the fast progress of the technologies of computer and communication in the world, knowledge acquired by network is rather easy than before that by traditional way. In response to the trend, the most attractive things on data processing are both to read the information fast and save it efficiently. In general, the size of an image is usually enormous than a text file so that elaborated manipulations for the image are extremely required in the course of transmission and storage. There is a technique of vector quantization (VQ) that could achieve a notable benefit in high performance of rate-distortion for image compression. It has been proven to be a simple and efficient method to fulfill image compression [5]. For VQ, it can be defined as a mapping relation U from k-dimensional Euclidean space R^k into a finite subset C of R^k, i.e., the relation $U: R^k \rightarrow C$, where the finite set $C=\{c_i | i=1, 2, .., N\}$ is called the *codebook*. The entry c_i is *codeword* and N is the total size of codebook. In the manner of VQ scheme, there are three phases, the codebook generation, the image encoding in finding a closest codeword and the image decoding in addressing the indices numbered on the codebook, respectively. In the first phase, a famous algorithm, named Linde, Buzo and Gray (LBG), is the most popular method for providing the design of codebook generation [10]. Afterwards, the image need to partition into several non-overlapping blocks in the second phase, where each block of $w \acute{} h$ pixels is a size of k and $k= w \acute{} h$. Clearly, each image block is a k-dimensional vector. For an image block x, it would be compared to each codeword in codebook C to find a closest codeword c_q, which has the smallest distance from x. Here, the distance measured between x and c_q is defined as

$$d(x, c_q) = \left(\sum_{i=1}^{k} (x_i - c_{q,i})^2 \right)^{\frac{1}{2}}, \qquad (1)$$

where x_i and $c_{q,i}$ denote the ith entry of x and c_q in vector representation, respectively. The index of the closest codeword in the codebook is then recorded instead of the vector itself. Following the same procedure for all image blocks, encoding compression is thus accomplished. The final decoding phase uses the recorded indices to find the corresponding codewords from the codebook.

Observe the three phases mentioned above, the encoding phase in searching the best-match codeword is very computationally intensive when the codebook is given beforehand and the full search algorithm (FSA) is employed to look up the codebook. Therefore, to design a efficient method in finding a closest codeword becomes importantly. In fact, there has been a lot of attention to promote the search approaches [1,6,7,8,9,13]. An improved version using tree-based structure came after LBG

algorithm, named tree-structure vector quantization (TSVQ), was subsequently proposed in [2] to overcome the defect of time-consuming in FSA. In TSVQ, the codebook is organized as a tree structure so that the closest codeword is searched from the root node to the leaf nodes level by level. Apparently, the massive computation cost of FSA is then improved dramatically. However, the extra payments are the more space storage and the deterioration of quality. In addition, there are also existed several approaches slightly mentioned below devoting to lessen the search time of finding a closest codeword. First of all, a partial distortion strategy in [1] is adopted to pass the overlarge codewords in distance calculation when comparing to the current available minimum distortion. A concept of geometric triangle inequality is next proposed to reduce the search space by using the mathematical inequality elimination in literatures [7,9]. Then a scheme of operating the topological structure of the codebook to remove the unnecessary codeword comparing was also proposed in [8]. In the other hand, there are some researchers [11,14] that devoted to shorten the time of computing Euclidean distance. In their ways, the squared numbers calculated in (1) are pre-stored in extra memory and then represented with a matrix form. In such a way, the time computation in (1) is significantly saved by means of preprocessing so that the fast implementation in VLSI and systolic architectures is feasible [4,12]. Afterwards, Chang and Chou [3] proposed an approximated approach to omit the infrequent used squared numbers itemized in matrix to further relieve the memory space in the hardware implementation.

In this paper, we exploit an idea to realize a fast search aiming at finding the closest codeword in VQ-based image representation. The technique used in our scheme is to further characterize the basic codebook in space partition. Meanwhile, the smaller magnitude of key-codebook is pre-generated by a special way of unbalanced-binary-tree under LBG-based algorithm. As a result, the speed boost in encoding the image blocks is significant achieved and the image quality is almost the same as that of the full search algorithm.

The rest of this paper is organized as follows. Section II is the preliminary to review the LBG algorithm. In the next section, three algorithms proposed by our scheme, named *KCG*, *PSM* and *UBT-LBG*, respectively, fulfill an efficient encoding phase in VQ. Empirical experiments are then followed in Section IV. Finally, the conclusions are given in Section V.

II. Preliminary

As we know, the basic method of searching a closest codeword in the codebook is to fully look over the codebook. Apparently, it is very time-consuming because the over table look-up is subject to the scenario of sequential search. As a matter of fact, the distribution of codewords is a keen concern in searching a specific codeword. If the codewords could be created well, the fidelity of an image processing would be increased. A well-known LBG algorithm a basic and main technique to generate a codebook, the codebook is then used to do encoding. In the following, we review a regular LBG algorithm that will be applied in our scheme.

LBG Algorithm: *Generate a codebook with the size N.*

Step 1: An initial codebook $C_1=\{c_1, c_2, c_j, \ldots, c_N\}$ is given, in which the N codewords is randomly generated.

Step 2: Set run-counter $t=1$ and average distortion $D_t=0$.

Step 3: A Lloyd iteration procedure is performed as follows:

- A codebook $C_t=\{c_1, c_2, c_j, \ldots, c_N\}$ is given, then looks for the closest codeword c_j in the C_t for each image block in the training set.
- For each codeword c_j, compute the mean value of the image blocks, which are closest to c_j, as an improved codeword c_j.
- The conducted $C_{t+1}=\{c_1, c_2, \ldots, c_j, \ldots, c_N\}$ is the improved codebook in place of C_t.

Step 4: Measure the average distortion D_{t+1} for the new generated C_{t+1}.

Step 5: Check the following inequality:

$$\left|\frac{D_{i+1}-D_i}{D_i}\right| \le \varepsilon, \qquad (2)$$

where ε is a threshold value. If the inequality is true, the resultant codebook is thus outcome; otherwise, set $t=t+1$ and go to Step 3.

■

Based on the LBG' s codebook generation, an efficient image process is fulfilled. In nature, an excellent codebook generation plays a very important medium for the fast search of closest codewords and the quality of image compressing. In the coming up section, the LBG methods will be applied to generate key-codebook to reduce the searching time of closest codewords.

III. Partial Search Mechanism for VQ-based Codebook Match

In this section, we discuss our mechanism for accelerating the closest codeword search in the codebook. Our approach is based on space partition. It directs searching algorithm to focus effort on partial region. The following symbols are defined to conveniently go through our scheme.

- C: a given codebook with size N.
- KC: a generated key-codebook with size N_c.
- x_i: an image block of k-dimensional cardinality.
- c_i: a codeword planted in C.
- k_i: a key-codeword planted in KC.
- D_{k_i} *:* a dominated set which consists of the codewords dominated by a key- codeword k_i.

Algorithm KCG: *The algorithm is to generate a key-codebook and its dominated sets.*

Input: A codebook C and parameter $\boldsymbol{d}$.

Output: A key-codebook KC and the dominated sets for all key-codewords in KC.

Step 1: [Determine the key-codebook size N_c]

- Compute $N_c = \sqrt{N}$.

Step 2: [Generate each k_i in KC]

- Call Algorithm *UBT-LBG*.

Step 3: [Initiate D_{k_i} for each k_i in KC]

- For each k_i, construct the set D_{k_i} consisting of the codewords c_i's that are the closest to k_i.

Step 4: [Extend the region of D_{k_i} for each k_i in KC]

- For each c_i, suppose that c_i is the closest to k_i. Then insert c_i to the other dominated sets D_{k_i}'s, if the following inequality is satisfied:

$$d(c_i, k_j) \leq d(c_i, k_i) + \boldsymbol{d}, \qquad (3)$$

where $i \neq j$.

■

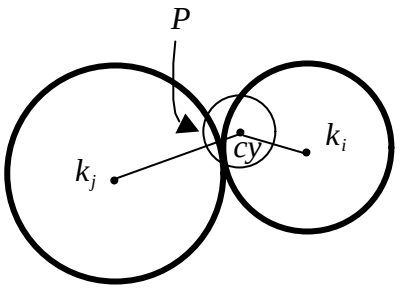


Figure 1. The c_i is supposed to be included to D_{k_j}, if the inequality in (3) holds.

Given an image block, Algorithm PSM searches the closest codeword by searching the closest key-codeword in *KC* first. Suppose that k_i is the key-codeword found by previous step. Then, it only handles those codewords in the dominated set D_{k_i}. Note that the key-codebook *KC* and its dominated sets both are created by Algorithm KCG beforehand. In particular, the creation of key-codebook in Algorithm KCG is similar to that of codebook generation in LBG method. In order to describing the concept of dominated set easily, we use a circle to depict the control region of c_i (or k_i), which is an area that c_i (or k_i) will be featured when any image block located in. The bold line drawn in Figure1 is used to separate key-codewords from codewords. Inspecting Figure 1, for example, c_i is the closest to k_i. So, c_i is included in D_{k_i}. Nevertheless, there possibly exists a phenomenon that the control region of c_i have an intersection, marked by region *P*, with D_{k_j}, but c_i is not contained in D_{k_j}. In the case, it will cause a condition that any image block located in the region *P* will direct to a wrong codeword. Namely, the input x_i in *P* won't be subject to a closest codeword c_i since c_i is only dominated by D_{k_i} in our basic algorithm. To avoid this situation, we have to extend the region of k_j to include c_i in gaining those possible closest codewords. Accordingly, a parameter $\boldsymbol{d}$ here is employed to measure the extension of area D_{k_i} until all the possible closest keywords are considered. Looking again in Step 4 of Algorithm KCG, c_i will be thus included in D_{k_j} if the formula (3) is satisfied.

Algorithm PSM: *Given an image block, the algorithm tries to find a closest codeword by partial search approach.*

Input: An image block x_i, key-codebook KC and the dominated sets for all key-codewords in KC.

Output: A closest codeword c_i.

Step 1: [Find the closest key-codeword from key-codebook]

- Search a key-codeword k_i from KC such that k_i is closest to x_i.

Step 2: [Find the closest codeword from the dominated set D_{k_i}]

- Search a codeword c_i from D_{k_i} such that c_i is closest to x_i.

Step 3: [Output c_i]

■

Algorithm UBT-LBG: **(Unbalanced-Binary-Tree generation in the LBG strategy)**

Input: N codewords in codebook C.

Output: N_c key-codewords planted into key-codebook KC.

Step 1: [Set the root node of UBT]

- Let V be a key-codeword, which is the centroid of the N codewords. And let V be the root node of UBT.
- Set C as the training set TS_V of V.

Step 2: [Determine the branch point of UBT]

- For all leaf nodes of UBT, choose a node V_{branch} with the largest size of training set.

Step 3: [Branch leaf nodes of UBT until there are $\sqrt{N}$ key-codewords generated]

- Call tree growing procedure with input V_{branch}.
- Back to Step 2 till the number of key-codewords is accumulated to $\sqrt{N}$.

Step 4: [Improve the key-codebook KC of containing N_c key-codewords]

- Let KC contain the N_c key-codewords created by previous steps.
- Apply LBG algorithm to improve codebook KC using the training set C.

■

Tree Growing Procedure:

Step 1: Given a key-codeword V, generate two key-codewords $V_1 = V$ and $V_2 = V*1.01$, where ' $*$ ' denotes the scalar multiplication.

Step 2: Apply LBG algorithm to improve the codebook containing V_1 and V_2 using the training set TS_V.

Step 3: Partition training set TS_V into two new training subsets TS_{V_1} and TS_{V_2}.

Step 4: Return the V_1, V_2, TS_{V_1} and TS_{V_2}.

■

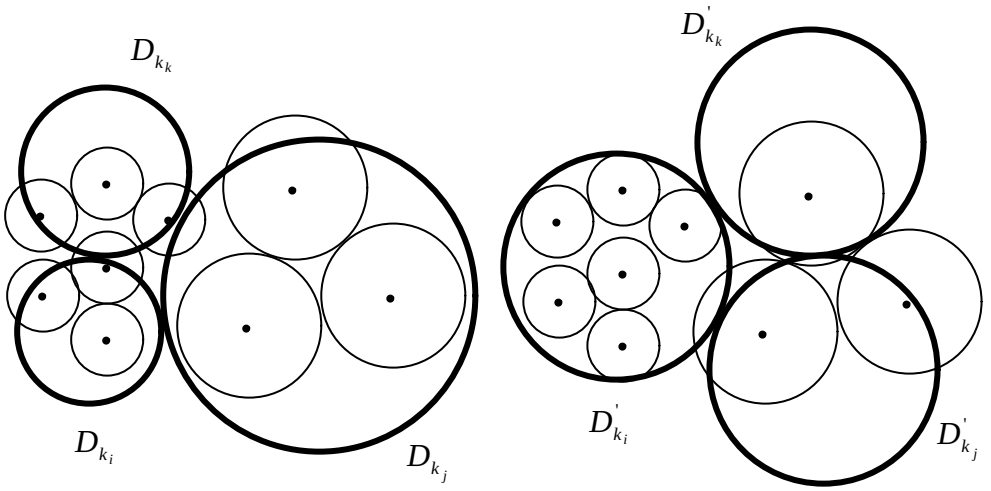


Figure 2. (a) Locate more key-codewords to the crowded regions. (b) No policy.

In our scheme the key-codebook is created by means of a model in UBT approach. In a sense, there are more key-codewords distributed in the crowded region in our manner. As shown in Figure 2 (a), the numbers of codewords involved by the dominated set D_{k_k}, D_{k_j} and D_{k_k} are 3, 3 and 3, respectively. In addition, as shown in Figure 2 (b), the numbers of codewords controlled by the sets of D'_{k_i}, D'_{k_j} and D'_{k_k} are 6, 2 and 1, respectively. To search the closest codewords for all image blocks, Figure 2(a) would take less time than Figure 2(b), because Figure 2(a) tries to avoid the condition that a dominated set would contain enormous codewords.

Property 1: *The scheme on the closest codeword search strategy we proposed is subject to recursive function call.*

Explanation. Assume the size of codebook given in our scheme is *N* and the search problem is *SP*. Theoretically, *SP*(*N*) can be transformed into two sub-problems in the following way such as:

$$SP(N) \Rightarrow SP(\mathrm{N}^{\frac{1}{2}}) + SP(K_1), \tag{4}$$

where K_1 is the size of a dominated set. Then, go on a splitting procedure in each sub-problem with a smaller size input. Therefore, we have the extended expression from as

$$\Rightarrow [SP(\mathrm{N}^{\frac{1}{4}}) + SP(\mathrm{K}_2)] + [SP(K_1^{\frac{1}{2}}) + SP(K_3)]$$

which is the recursive form in function call for each new smaller table to be performed search. Also, all sub-problem can be solved by any search approach. The numeric analysis of K_1 in experiment is given next section.

IV. Experiments Analysis and Result

In our scheme, all the experiments are executed on a platform of IBM PC with a Pentium III of 450 MHZ. All images used in these experiments are 512×512 monochrome still image. There are six images (Girl, Gold, Lena, Pepper, Toys and Tiffany), whose pixels contain 256 gray levels, as our training set and each image isdivided into 4×4 pixels blocks. Each block isa 16-dimensional vector. A well-known Linde-Buzo-Gray (LBG) algorithm is subsequently applied with a threshold ε=0.001 to generate a codebook C with size 256. The quality of image compressing in VQ-based encoding ismeasured by the peak signal-to-noise ratio (PSNR), which is defined as

$$\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSE}}\mathrm{dB}. \tag{5}$$

For an $m \times m$ image, the mean-square error (MSE) is defined as

$$\mathrm{MSE} = \left(\frac{1}{m}\right)^2 \sum_{j=1}^{m}\sum_{j=1}^{m}(x_{i,j} - \tilde{x}_{i,j})^2, \tag{6}$$

where $x_{i,j}$ and $\tilde{x}_{i,j}$ denote the original and qualified gray level of pixel in the image, respectively.

There are several simulation results shown in the below tables to illustrate the efficiency performed in our scheme. In Table 1 and Table 2, the missing distribution for a closest codeword match in C is counted among the six images by PSM with $\boldsymbol{d} = 0$, in which the key-codebook are created by LBG algorithm and UBT-LBG algorithm, respectively. Let x be an image block. Also, let c_1 and c_2 be the closest codewords found by PSM and full search, respectively. If $d(x, c_1) \neq d(x, c_2)$, then there is a

missing hit occurred. Row 2 of Table 1 and Table 2 show the number of missing hits such that the differences between $d(x, c_1)$ and $d(x, c_2)$ are less than or equal to 10. Note that all missing hits in Row 2 will disappear by our PSM approach when δ is set to 10. Note that there are 16384 blocks for each image. So, the ratio of missing for each image is less than (2843 / 16384) = 17.35%. Figure 2 could be used to explain why the values of Row 2 in Table 2 are larger than that in Table 1. For UBT-LBG method, it shows that the crowded region will appear more key-codewords. So, there will be more interlaces between the control regions of key-codewords and codewords. Thus, the values of Row 2 in Table 2 are larger. When δ is increased, the interlaced codewords will be easily included into the dominated sets of key-codewords. This is why Table 2 needs smaller δ to avoid all missing hits than that of Table 1.

Table 3 and Table 4 depict the PSNR values and run times of PSM and full search. DS is the sum of $|D_z|$ for all key-codeword z in KC. When $\delta = 0$, the PSNR values of PSM by LBG and UBT-LBG methods are 98.59% and 98.46% in comparing to that of full search, respectively. Also, it is almost 100% when $\delta = 90$ and $\delta = 80$ for LBG and UBT-LBG methods, respectively. The values of DS in Table 4 are usually larger than that in Table 3, because there would be more codewords be included when the controlling region is enlarged in UBT-LBG method. But the run times in Table 4 are better than that in Table 3. This is the reason that we use UBT-LBG method to create key-codebook for uniform distributing codewords to D_k of all key-codeword k. In Table 4, DS is 1008 when $\delta = 80$. So, the value of K_1 depicted in Property 1 is about (1008 / 16) = 63. Given an image block, 256 codewords need to be compared in full search approach. But, it is about 16 + 63 = 78 codewords that need to be compared in our PSM approach.

The experiments measured in our scheme further show that the encoding time of our proposed algorithm is faster twice more than that of full-search in VQ scheme.

V. Conclusions

We proposed a partial search mechanism to expedite the codeword search in this paper. This method can be technically adopted to save the time cost on a best-match codeword in a given codebook. As a whole, there are four prominent key features involved in our scheme to improve the search algorithm in VQ-based image process, which are partial range located, recursive segmentation search, codeword duplicated checking and UBT-LBG key-codebook generation, respectively. In the first feature, it could reduce the time-consuming in full table look-up. The second feature could foster the extension in the closest codeword match. The third feature enables the lowest missing hit-ratio to be achieved. The forth feature, UBT-LBG key-codebook

generation, enables the codewords to be uniformly distribute into the dominated sets of key-codewords. This will improve the run time. It's obviously shown that the time cost in encoding an input image is at least faster twice than that of full search, as observed from the experimental display. Moreover, our scheme could be explored into an in-depth search in hierarchy by recursive function call together with the fast computation in Euclidean distance evaluation so that the performance of fast search on the best-match codeword could be bolstered up in nature.

References

[1] C.D. Bei and R.M. Gay, "An improvement of the minimum distortion encoding algorithm for vector quantization," IEEE Trans. Commun., Vol. 33, pp. 1132-1133, 1985.

[2] A. Buzo, A.H.Gray, R.M.Gray and J.D. Markel, "Speech coding based upon vector uantization," IEEE Trans. Acoustics, Speech and Signal Processing, Vol.18, pp. 562-574, 1980.

[3] C.C. Chang and J.S. Chen, "An efficient computation of Euclidean distance using an approximated look-up tables, " to be appeared in IEEE Trans. Circuits and Systems for Video Technology.

[4] G.A. Davidson, P.R. Cappello, and A. Gersho, "Systolic architectures for vector quantization, " IEEE Trans. Acoust., Speech, Signal Process, Vol. 36, Oct. pp. 1651-1664, 1988.

[5] A. Gersho and R.M. Gray, Vector Quantization and Signal Compression. Boston, MA: Kluwer, 1992.

[6] C.H. Hsieh and Y.J, Liu, "Fast search algoirthm for vector quantization of images using multiple triangle inequalities and wavelet transform," IEEE Trans. Image Processing, Vol. 9, No. 3, 2000.

[7] C.M. Huang, Q. Bi, G.S. Stiles and R.W. Harris, "Fast full search equivalent encoding algorithms for image compression using vector quantization," IEEE Trans. Image Processing, Vol. 1, pp. 413-416, 1992.

[8] C.H. Lee and L.H. Chen, "A fast vector search algorithm for vector quantization using mean pyramids of codevector," IEEE Trans. Commun., Vol. 43, pp. 1697-1702, 1995.

[9] W. Li and E. Salari, "A fast vector quantization encoding method for image compression, "IEEE Trans. Circuits Syst. Video Technol., Vol. 5, pp. 119-123, 1995.

[10] Y. Linde, A. Buzo and R.M. Gray, "An Algorithm for vector quantizer design, " IEEE Trans. Commun. COM-28, 1980, pp. 84-95.

[11] H. Park and V.K. Prasana, "Modular VLSI architectures for real-time full-

search-based vector quantization," IEEE Trans. Circuits Systems for Video Technology, Vol. 3, No. 4, pp. 309-317, 1993.

[12] P.A. Ramamoorthy, B. Potu and T. Tran, "Bit-serial VLSI implementatin for vector quantizer for real-time image coding," IEEE Trans. Circuits and Systems for Video Technology, Vol. 36, Oct., pp. 1281-1290, 1989.

[13] V. Ramasubamanian and K.K. Paliwai, "Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding," IEEE Trans. Signal Processing, Vol. 40, pp. 518-531, 1992.

[14] S. A. Rizvi and N. M. Nasrabadi, "An efficient Euclidean distance computation for vector quantization using a truncated look-up table," IEEE Trans. Circuits and Systems for Video Technology, Vol. 5, Aug. pp. 370-371, 1995.

Table 1. The missing distribution of image blocks for the key-codebooks created by LBG method.

No. of Missing Hits	Girl	Gold	Lena	Pepper	Toys	Tiffany
difference ranged in 0~10	562	587	629	632	700	609
difference ranged in 10~20	361	422	258	290	286	199
difference ranged in 20~30	166	192	157	116	153	81
difference ranged in 30~40	55	89	54	58	40	27
difference ranged in 40~50	12	60	35	19	32	12
difference ranged in 50~60	9	6	7	12	10	3
difference ranged in 60~70	4	9	7	10	6	12
difference ranged in 70~80	4	1	4	5	5	1
difference ranged in 80~90	4	1	2	5	1	1
difference ranged in 90~100	0	0	0	0	0	0
difference over 100	2	0	0	1	0	0
Total missing number	1179	1367	1153	1148	1233	945
Max Difference	144.31	89.48	81.25	126.15	89.48	89.48

Table 2. The missing distribution of image blocks for the key-codebooks created by UBT-LBG method.

No. of Missing Hits	Girl	Gold	Lena	Pepper	Toys	Tiffany
difference ranged in 0~10	1168	1582	1230	1203	1105	879
difference ranged in 10~20	791	902	554	603	266	383
difference ranged in 20~30	153	188	162	140	76	103
difference ranged in 30~40	67	129	61	57	58	31
difference ranged in 40~50	22	32	32	26	41	6
difference ranged in 50~60	1	8	4	4	4	3
difference ranged in 60~70	0	1	1	5	1	0
difference ranged in 70~80	0	0	4	0	2	0
difference ranged in 80~90	0	1	0	1	0	0
difference ranged in 90~100	0	0	0	0	0	0
difference over 100	0	0	0	0	0	0
Total missing number	2202	2843	2048	2039	1553	1405
Max Difference	54.81	80.94	77.90	80.94	79.93	54.33

Table 3. The decoding comparison using LBG algorithm between FSA and PSM search manner.

	FSA (Full Search)		PSM(δ = 0) DS = 256		PSM(δ = 50) DS = 630		PSM(δ = 90) DS = 989	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Girl	32.14656	4.94	31.87138	1.32	32.13145	1.86	32.14453	2.42
Gold	30.62092	4.94	30.30123	1.31	30.61687	1.76	30.62092	2.25
Lena	31.49239	4.94	31.09754	1.31	31.48081	1.86	31.49239	2.42
Pepper	31.58969	4.95	31.21851	1.31	31.51622	1.76	31.58900	2.36
Toys	31.41352	4.89	30.68427	1.15	31.38148	1.54	31.41352	1.92
Tiffany	32.70695	4.94	32.10939	1.26	32.66555	1.76	32.70695	2.25
Average	31.66167	4.93	31.21372	1.28	31.63206	1.76	31.66122	2.27
Ratio	1	1	98.59%	25.88%	99.91%	35.61%	100.00%	46.01%

Table 4. The decoding comparison using UBT-LBG between FSA and PSM search manner.

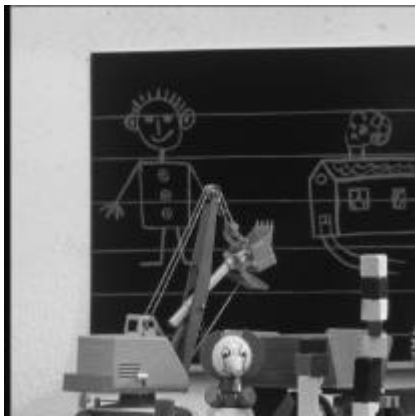
	FSA (Full Search)		PSM(δ = 0) DS = 256		PSM(δ = 50) DS= 701		PSM(δ = 80) DS = 1008	
	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
Girl	32.14656	4.94	31.62128	1.15	32.14614	1.82	32.14656	2.25
Gold	30.62092	4.94	30.24782	1.10	30.61613	1.65	30.61956	2.08
Lena	31.49239	4.94	31.09033	1.16	31.48373	1.76	31.49239	2.20
Pepper	31.58969	4.95	31.09995	1.15	31.58121	1.76	31.58798	2.14
Toys	31.41352	4.89	30.76057	1.10	31.40758	1.59	31.41352	1.87
Tiffany	32.70695	4.94	32.22769	1.15	32.70484	1.75	32.70695	2.09
Average	31.66167	4.93	31.17461	1.14	31.65660	1.72	31.66116	2.11
Ratio	1	1	98.46%	23.01%	99.98%	34.90%	100.00%	42.67%

Girl

Gold

Lena

Pepper



Toys

Tiffany

Fig. 3. Six images with Girl, Gold, Lena, Pepper, Toys and Tiffany