The Reduction of VQ Index Table Size by Matching Side Pixels

Ya-Feng Di [1], Zhi-Hui Wang [1], Chin-Feng Lee[2, *], Chin-Chen Chang[3]

[1] School of software, Dalian University of Technology
Dalian, China 116620
dyf.dlut@gmail.com, wangzhihui1017@gmail.com

[2] Department of Information Management, Chaoyang University of Technology
Taichung, Taiwan 41349
[*]Whom correspondence: lcf@cyut.edu.tw

[3] Department of Information Engineering and Computer Science, Feng Chia University
Taichung, Taiwan
e-mail: alan3c@gmail.com

Abstract. The vector quantization (VQ) technology is applied to compress an image based on a local optimal codebook, and as a result an index table will be generated. In this paper, we propose a novel matching side pixels method to reduce the index table for enhancing VQ compression rate. We utilize the high correlation between neighboring indices, the upper and the left of the current index, to find the side pixels, and then reformulate the index. Under the help of these side pixels, we can predict the adjacent elements of the current index and then partition the codewords into several groups for using fewer bits to represent the original index. Experimental results reveal that our proposed scheme can further reduce the VQ index table size. Compared with the classic and state-of-the-art methods, the results reveal that the proposed scheme can also achieve better performance.

Keywords: vector quantization; matching side pixels; image compression

1 Introduction

With the rapid development of information technology, more and more people utilize the Internet to communicate and exchange multimedia information, such as images, audios, and videos. Among them images occupy a large proportion, which brings storage issues that need urgent solution. In order to reduce the cost of storage and increase the transmission speed, the images need to be compressed before transmitting [1][2][3]. Vector quantization [4], as a typical and effective compression technique is widely used in field of image compression [5][6]. VQ produces an index table by quantizing a given image in block-wise manner. The indices can stand for the whole image, in which the size of the former is much smaller than the latter, so VQ

J.-S. Pan et al. (eds.), *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, Smart Innovation, Systems and Technologies 64,
DOI 10.1007/978-3-319-50212-0_25

can achieve a good compression rate. While the codebook design is a quite important work before compressing the image, it is usually trained with the well-known Linde-Buzo-Gray (LBC) algorithm [4].

However, people find that the index table can still be reduced to some degree, so some techniques which can further compress the index table are proposed to improve the compression effect. The famous compressing index table methods includes search-order-coding (SOC) and improved search-order-coding (ISOC) which can be applied to further losslessly compress VQ index table. SOC was proposed by Hsieh and Tsai in 1996 [7] and the algorithm encodes each index one by one in a raster scan order to find the same index with the current index along a predefined search path. After finding the same index, replace the original index with the search order code which is shorter than the original one. In 1999, Hu and Chang proposed an improved SOC method [8] to further compress an index table. Before doing the search-order-coding operation, sort the codewords in the codebook according their mean values in the descending order. By doing so, the index value can be more approximate with the neighboring indices. Besides the compression methods, a data hiding scheme based on search-order-coding and state-codebook mapping was proposed by Lin *et al* in 2015 [9]. Lin *et al* first employ the SOC method to reduce index table and then use the sate-codebook mapping method to deal with the indices which cannot be proposed by SOC. Lin *et al*'s method has good performance on reducing the index table; however, this method is a bit complicated and is not easy to implement. In this paper, we propose a novel method for reducing a VQ index table by matching side pixels. By exploiting the high correlation between the neighboring indices, we utilize the side pixels of the neighboring indices to predict and find the correct index. The experimental results demonstrate that our proposed scheme has a better performance in compression rate compared with SOC, ISOC and Lin *et al*'s method.

The reminder of this paper is organized as follows. Section 2 gives a brief introduction of the conventional VQ technique. The proposed scheme is described in Section 3 in detail. Experimental results are demonstrated in Section 4 and the conclusion part follows.

2 Related Work

In this section, a brief introduction of vector quantization is described and an example is also provided for better understanding.

Vector quantization (VQ) is proposed in 1980 by Linde *et al.* [4]. Due to its simple and cost-effective advantage, VQ is used in numerous applications, such as image and video compression. A complete VQ system consists of three parts, codebook design, compression and decompression. Designing a proper codebook has significant effect on the compression results. Generally, the popular Linde-Buzo-Gray (LBG) algorithm is utilized to train a representative codebook with several images. In the compression procedure, the image first needs be partitioned into a series non-overlapping blocks, and the size of each block is $l \times l$. Then each block is encoded with the index of the

best matched codeword in the codebook. The codebook has C codewords, and each codeword C_i is a $l \times l$ dimensional vector, where $i = 0, 1, 2, \ldots, C\text{-}1$. For the current block assumed as X, calculate the Euclidean distance between X and each codeword C_i with following equation (1)

$$Dis(X, C_i) = \|X - C_i\| = \sqrt{\sum_{j=1}^{l \times l} (x_j - c_{ij})^2}\ , \tag{1}$$

where x_j is the jth pixel of the current block, c_{ij} represents the jth element of the ith codeword C_i, $j = 1, 2, \ldots, l \times l$.

The codeword which has the minimal Euclidean distance with the current block X, is the best matched codeword. Then utilize the index of the best matched codeword to stand for the current block. After obtaining the indices of all blocks in the image, an index table is generated and need be stored. When decompressing the image, the receiver can refer the indices in the index table and the codewords in the codebook using a simple table look-up method to reconstruct the image. There is one thing to point that, VQ compression is a lossy algorithm.

Figure 1 shows a 512×512 image *Lena* and each block size is 4×4. A codebook with 256 codewords generated by the LBG algorithm is illustrated in Figure 2. Simultaneously, after the VQ compression phase, we can get an index table which has 512×512/ (4×4) = 16384 indices. To measure the compression efficiency, bit rate is the most commonly used metrics. The bit rate is 16384×8/ (512×512) = 0.5 bpp in the example.

Fig. 1. *Lena image*

index	codeword
0	(22, 35, 22, …, 24)
1	(57, 53, 51, …, 55)
2	(75, 52, 63, …, 58)
3	(81, 79, 81, …, 107)
4	(117, 92, 70, …, 94)
5	(86, 73, 92, …, 106)
6	(103, 109, 116, …, 105)
	. . .
253	(46, 55, 77, …, 45)
254	(167, 152, 122, …, 164)
255	(87, 142, 153, …, 104)

Fig. 2. An example of codebook

3 Proposed Scheme

Driven by the motivation of reducing the index table, in this section, we propose a novel method to reduce the VQ index table by matching the side pixels.

3.1 Side pixels matching

In the index table, except the indices encoded using the conventional VQ algorithm at the first row and first column, each of the residual indices has its upper and left adjacent index neighbors which can be utilized to compress further by the proposed matching side pixels. Figure 3 demonstrates some part of the generated index table in the case of block size 4×4. The variables t_1, t_2, …, t_{16} are the specific elements of the current index T. For a current index T, which belongs to the residual indices, side pixels refers to the elements $\{t_1, t_2, t_3, t_4, t_2, t_9, t_{13}\}$ at the first row and column elements in the current index and their adjacent elements in the neighboring blocks, which are the shaded pixels u_{13}, u_{14}, u_{15}, u_{16} in the upper index U and l_4, l_8, l_{12}, l_{16} in the left index L. We design a method of side pixel matching which utilizes $\{u_{13}, u_{14}, u_{15}, u_{16}, l_4, l_8, l_{12}, l_{16}\}$ to predict the adjacent elements $\{ t_1, t_2, t_3, t_4, t_2, t_9, t_{13}\}$ in the current index to realize the index table reducing.

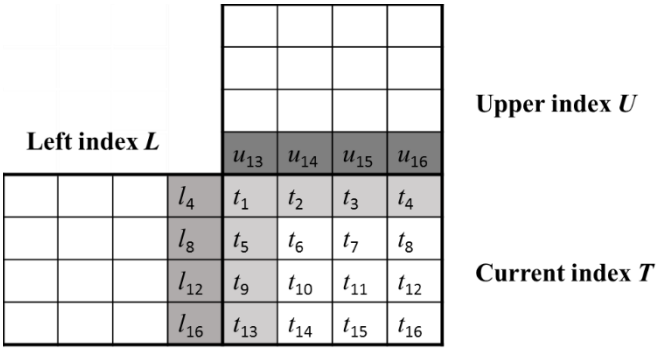

Fig. 3. Side pixels of the index T

3.2 Reduce the index table

The fewer bits to represent the indices, the higher compression rate can be achieved. First we utilize the side pixels to predict the adjacent elements of the current index. Assume the block size is $l \times l$, there are $2l$-1 adjacent elements in the current index. The first element in the corner of the current index is forecasted by calculating the average of its upper and left element. The other adjacent elements in the current index are directly predicted by equaling their neighboring elements. Take the index table in Figure 3 as an example, in the current index T, the value t_1 can be forecasted by solving the equation $t_1 = (l_4 + u_{13})/2$. The other values of the current index are

directly predicted by $t_2 = u_{14}$, $t_3 = u_{15}$, $t_4 = u_{16}$, $t_5 = l_8$, $t_9 = l_{12}$, $t_{13} = l_{16}$. Then calculate the Euclidean distance d_T between the predicted elements of the current index T and the corresponding elements of the codewords using following Equation (2)

$$d_T = \sqrt{\sum_{j=1}^{2l-1} (p_j - c_{ij})^2}, \tag{2}$$

where p_j is the jth predicted element of the current index, and c_{ij} is the corresponding element of the ith codeword C_i, j = 1, 2, 3, 4, 5, 9, 13.

After obtaining the distances, a sorting operation is utilized on them. Specifically, by sorting the distances in the ascending order between the seven values with the corresponding elements of the codewords in the codebook, we can get a distance list. Then we divide the sorted distances list into four non-overlapping groups, G_0, G_1, G_2 and G_3 as follows,

$$\begin{cases} G_0 = \{0,\ 1,\ 2,\ \ldots,\ n-1\} \\ G_1 = \{n,\ n+1,\ \ldots,\ 2n-1\} \\ G_2 = \{2n,\ 2n+1,\ \ldots,\ 4n-1\} \\ G_3 = \{4n,\ 4n+1,\ \ldots,\ N-1\} \end{cases},$$

where n stands for the number in the first group and N means the codewords number in the codebook. So we can use two bits stand for the correct index is in which group, Specifically, “00” represents G_0, “01” represents G_1, “10” and “11” stands for G_2 and G_3 respectively. As can be seen from the probability, if the correct index lies in the first three groups, we can use fewer bits represent the index instead of original 8 bits. Though the unfortunate situation still has the possibility to occur, experimental results reveal that the index lies in the first three groups occupy the majority. An example is given to illustrate the index table reducing procedure and prove the compression effect.

In the example, we set n equals 8 and the current index T is 7. And the codebook has 256 codewords. Then the Euclidean distance is calculated between the seven side pixels with the elements in the corresponding positions of the codewords. After sorting the codewords, the method partitions the sorted codewords into four groups. Figure 4 shows the grouping situation. If the correct CW_7 is in G_0, the correct index 7 can be represented by “00” adding three bits, that is five bits; if CW_7 is in G_1, the correct index code can be represented by adding two bits “01” to the head of the index binary representation such that CW_7 can be presented as five bits; if CW_7 is in G_2, the correct index 7 can be represented by left-padding “10” to the four-bit representation of CW_7 to form a six-bit index code; unfortunately if CW_7 falls in G_3, the correct index 7 can be represented by ten bits due to two leading bits “11” should be added to its original eight-bit representation so the length of index code is ten. Figure 5 reveals the distribution of correct index lying in groups when n equals 8. From the figure, we

can find that the probability of the correct index lies in the first three groups is far greater than that in the last group, which proves that our proposed method can reduce the index table significantly. Moreover, we assign the variable n with different values and compare the different compression rates to obtain the best performance outcome.

G_0	
0	CW_5
1	CW_{17}
2	CW_{78}
⋮	⋮
7	CW_{23}

G_1	
8	CW_{18}
9	CW_{35}
10	CW_{15}
⋮	⋮
15	CW_{179}

G_2	
16	CW_{20}
17	CW_{155}
18	CW_{234}
⋮	⋮
31	CW_{254}

G_3	
32	CW_{95}
33	CW_{28}
34	CW_{198}
⋮	⋮
255	CW_{223}

Fig. 4. An example of grouping result

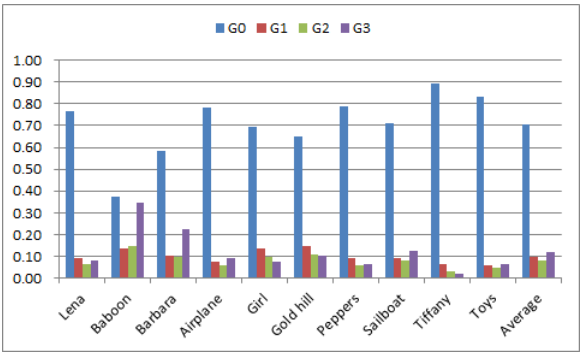

Fig. 5. The distributions of index grouping when $n = 8$

4 Experimental Results

In this section, experimental results are demonstrated to verify our proposed scheme with satisfactory performance in terms of reducing the index table after VQ operation. In our experiments, ten typical gray images sized 512×512 are used as the test images. Each image is partitioned into 4×4 blocks without overlapping and a codebook consisting 256 codewords is also prepared preliminarily. To obtain best experimental results, the variable n is assigned with 16, 8 and 4, respectively.

Table 1 illustrates the performance of the test images with various n values. We can find that all of the results are better than the original VQ compression bit rate which is 0.5 bpp. When n equals 4, the proposed scheme can get the smallest bit rate which means the compression rate is the most satisfactory.

Table 1. Comparison of bit rate (bpp) when *n* takes different values

	Lena	Baboon	Barbara	Airplane	Girl	Gold hill	Peppers	Sailboat	Tiffany	Toys	Average
n=16	0.3897	0.4402	0.4194	0.3923	0.3869	0.3900	0.3874	0.3977	0.3792	0.3877	0.3971
n=8	0.3440	0.4304	0.3899	0.3461	0.3443	0.3523	0.3384	0.3574	0.3225	0.3378	0.3563
n=4	0.3122	0.4429	0.3773	0.3121	0.3247	0.3386	0.3122	0.3122	0.2743	0.2974	0.3316

The conventional VQ compression technique cannot gain satisfactory compression rate, so some improved compression algorithms such as SOC and ISOC based on VQ have been also proposed to get lower bit rate. We also make comparisons with these two typical improved algorithms. Table 2 demonstrates the bit rate of these two techniques as well as our proposed scheme in which *Lena* is taken for example.

Table 2. Comparison of bit rate (bpp) with SOC and ISOC

Methods	*m*	BR
SOC	*m* = 2	0.357
	m = 4	0.341
	m = 8	0.359
ISOC	*m* = 2	0.329
	m = 4	0.322
	m = 8	0.344
Ours	0.3122	

Table 3. Comparison of bit rate (bpp) with Lin *et al*'s method

	Lena	Baboon	Airplane	Girl	Peppers	Sailboat	Tiffany	Average
n = 4	0.3122	0.4429	0.3121	0.3247	0.3122	0.3122	0.2743	0.3316
Lin *et al* [8]	0.3165	0.4526	0.3190	0.3662	0.3284	0.3883	0.2976	0.3527

5 Conclusions

In this paper, a novel VQ index table representation by matching side pixels is proposed. By exploiting the correlation between the side pixels of the neighboring indices, we can use fewer bits to represent the original index. Experiment results present that the proposed scheme can achieve better compression performance compared with the classic compression index table methods and the state-of-the-art methods such as SOC and ISOC methods.

In the future, we plan to explore the proposed index table reduction to information hiding. The proposed information hiding scheme can hide a huge amount of information in the index map of an image and allows complete reconstruction of the indexes of the image. We would like to propose an information embedding scheme to hide a huge amount of information in the reduction of an index map and allows complete reconstruction of the indexes of the image.

References

1. C. Shi, J. Zhang and Y. Zhang: Content-based onboard compression for remote sensing images. Neurocomputing, Vol. 191, pp. 330-340 (2016)
2. H. S. Li, Q. Zhu, M. C. Li, and H. Ian: Multidimensional color image storage, retrieval, and compression based on quantum amplitudes and phases. Information Sciences, Vol. 273, pp. 212-232 (2014)
3. L. Zhang, L. Zhang, D. Tao, X. Huang and B. Du: Compression of hyperspectral remote sensing images by tensor approach. Neurocomputing, Vol. 147, No. 1, pp. 358–363 (2015)
4. Y. Linde, A. Buzo, and R.M. Gray: An algorithm for vector quantization design. IEEE Transactions on communications. Vol. 28, No. 1, pp. 84-95 (1980)
5. M. Lakshmi, J. Senthilkumar, and Y. Suresh: Visually lossless compression for Bayer color filter array using optimized Vector Quantization. Applied Soft Computing, Vol. 46, pp. 1030-1042.
6. Y.K. Chan, H.F. Wang, and C.F. Lee,: “A refined VQ-Based image compression method,” *Fundamenta Informaticae*, Vol. 61, No. 3-4, pp. 213-221 (2004)
7. C. H. Hsieh and J. C. Tsai: Lossless compression of VQ index with search-order coding. IEEE Transactions on Image Processing, Vol. 5, No. 11, pp. 1579-1582 (1996)
8. Y. C. Hu and C. C. Chang: Low complexity index-compressed vector quantization for image compression. IEEE Transactions on Consumer Electronics, Vol. 45, No. 1, pp. 1225-1233 (1999)
9. C. C. Lin, X. L. Liu and S. M. Yuan: Reversible data hiding for VQ-compressed images based on search-order coding and state-codebook mapping. Information Sciences, Vol. 293, pp. 314-326 (2015)