Compression on the Block Indexes in Image Vector Quantization

Chung-Hsien Chiou(邱忠賢)* and Chang-Biau Yang(楊昌彪)**

* Department of Applied Mathematics,
** Department of Computer Science and Engineering,
National Sun Yat-sen University, Kaohsiung, Taiwan
Email:cbyang@cse.nsysu.edu.tw

Abstract

In this paper, we propose a new compression scheme for the block indexes. The goal of this method is to improve bit ratio scheme with the same image quality. We apply the TSP (traveling salesperson) scheme to reorder the codewords in the codebook. Then, the block indexes in an image are also remapped according to the reordered codebook. Finally, we compress the block indexes of the image with some lossless compression methods. Adding our TSP scheme as a step in VQ (vector quantization) index compression really achieves significant reduction of bit rates. Our experiment results show that the bpp (bits per pixel) in our method is less than the bpp of those without the TSP scheme.

1. Introduction

Vector Quantization (VQ) [1,3,4,6,8] is a kind of lossy compression method for image encoding. It uses a codebook containing block patterns with one corresponding index on each of them. The main idea of VQ is to represent a block of pixels by an index in the codebook. In this way, compression is achieved because the size of the index is usually a small fraction of that of the block.

Image compression falls into two categories: lossy and lossless. Examples of lossy methods include vector quantization (VQ), *fractal* [5], and etc. And there are also many lossless methods, such as *universal context modeling* (UCM) [11], *set partitioning in hierarchical trees* (SPIHT) [10], *context-based adaptive lossless image coding* (CALIC) [12], and etc. The lossy image compression can provide high compression ratio but it will lose some information of image. For the lossless image compression, there is no degradation in the restored images. The SPIHT coding is a powerful image compression algorithm; it produces an embedded bit stream from which the best reconstructed images in the mean square error sense can be extracted at various bit rates. The CALIC coding is accomplished with low time complexity and it provides better compression ratio.

In this paper, we shall propose a simple TSP (traveling person) scheme in the VQ (vector quantization) index compression. We apply the TSP scheme to reorder the codewords in the codebook such that the difference between the indexes in neighboring blocks of the image becomes small. Then, the block indexes in the image are remapped according to the reordered codebook. Thus, the variation between two neighboring block indexes is reduced. Finally, we compress the block indexes with some lossless compression methods.

The rest of this paper is organized as follows. We shall show, in Section 2, how to improve compression ratio with our TSP reordering scheme. Section 3 contains the experimental results and we shall compare the performance of our reordering scheme with various lossless compression algorithms. Finally, Section 4 concludes the paper.

2. The TSP Scheme for VQ Indexes

In this section, we shall propose our method: TSP-VQ (traveling salesperson for vector quantization) [2,7,13]. Our main idea is to reduce the difference between two neighboring block indexes. We use the traveling salesperson problem (TSP) as a guide to achieve this goal. The block indexes of an image obtained from the VQ compression method are usually not well organized, and they look like a set of random data. If we apply some lossless compression techniques, such as the arithmetic and Huffman coding, on the block indexes, we can often get a little improvement, but we can not get vary large improvement. In our experiment, the storage required for the block indexes with the arithmetic coding is nearly 10% less than that of the block indexes without any compression method.

Assume that if two block index values are equal, it means that they may have the same feature. We can treat a set of block indexes as a reduced image, where a block index represents a pixel in the reduced image. The newly compression methods, such as the SPIHT [10] and CALIC coding [12], are efficient in lossless image compression. Those methods almost consider the difference of two pixels, not only the pixels themselves. The question is how to decrease the difference of two pixels. The decrement of the difference of two neighboring block indexes is similar to that the difference distribution gathers at the near-zero area. The gathering of the difference will lead to an efficient compression.

2.1 An Algorithm for the Traveling Salesperson Problem (TSP)

The traveling salesperson (TSP) problem is defined as follows: Given a set of n points, find a shortest closed tour containing all points of the set. This is an NP-complete problem. Thus, it is very difficult to solve the TSP problem if there are many points in the input set. Several methods for this problem have been proposed. But those methods spend too much time for getting the optimal solution. For example, Bellman [2] proposed a dynamic programming method for solving the TSP problem, which requires $O(n^2 2^n)$ time, where n is the number of the input points. Here, we shall apply the TSP method to reorganize the codebook obtained in the VQ compression. In general, the codebook size is 256 or 512, thus the number of calculation units is about $2^{256} \approx 10^{77}$. So it is not practical to use those methods to solve the TSP for obtaining the optimal solution.

In this section, we shall propose a heuristic algorithm to solve the TSP problem. Our algorithm is simple and fast. Our main scheme is the *amphisbaena* (a snake with two heads) scheme that connects one edge from the head or tail in each pass.

Let us consider the undirected graph shown in Figure 1. First, we find the edge with the minimum cost in the graph. Let one point of this edge be the head of the list and another be the tail of the list (tour). Next, we connect the minimal edge which is connected to either the head or tail of the list. We repeat this procedure $n-2$ times. The time complexity of this algorithm is $O(n^2)$, which is much faster than those optimal algorithms.

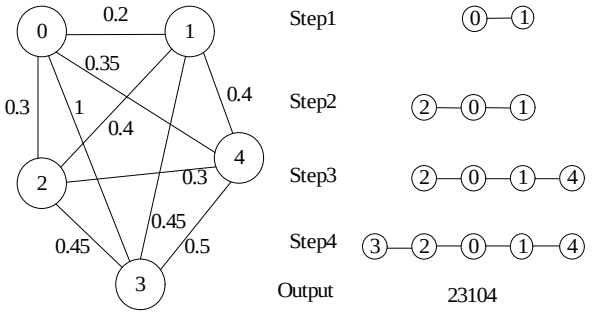


Figure 1: An example for solving the traveling salesperson problem algorithm.

Take the graph shown in Figure 1 as an example. The set of input points (nodes) is $S=(0,1,2,3,4)$. First, we find the minimal edge of the graph as the initial list (tour). The edge connecting *0* and *1* is the minimal. So the head of the list is set to *0* and the tail is set to *1*. Now, the list T is $(0,1)$. Next, we check all points which connects to either the head or tail of the list to find the next minimal edge. Note that we only check the points which have not been visited. The edge connecting *2* and *0* is the next minimal. So the head of the list is changed from *0* to *2* and the list T becomes $(2,0,1)$. Then, this procedure is performed repeatedly. The next minimal edge is the one connecting *1* and *4*. Thus, the list T becomes $(2,0,1,4)$. The final list T is $(3,2,0,1,4)$, which represents the tour of the TSP. Note that we do not find a close tour here, we only find an open tour containing all input points. Since $S_0=T_2$, $S_1=T_3$, $S_2=T_1$, $S_3=T_0$, $S_4=T_4$, we output the reordered list $R=(2,3,1,0,4)$. Our TSP algorithm is given as follows.

Algorithm Amphisbaena-TSP

Input: N points with the cost matrix $C_{N\times N}$.

Output: The reordered list $R=(R_1,R_2,\cdots,R_N)$ is a permutation of $(1,2,3,\cdots,N)$ such that $\sum_{i=1}^{N-1} C(R_i, R_{i+1})$ is nearly minimized.

Step 1: Let $S=(1,2,\cdots,N)$, and T be an empty list.

Step 2: Find p,q such that $C(p,q)=min_{i,j\in S}C(i,j)$. Remove p and q from S, then set T as (p,q).

Step 3: Find a,b such that $C(p,a)=min_{j\in S}C(p,j)$ and $C(q,b)= min_{j\in S}C(q,j)$. If $C(p,a)<C(q,b)$, then remove a from S, add a to the head of T and set $p=a$; otherwise, remove b from S, add b to the tail of T and set $q=b$.

Step 4: If S is not empty, then go to step 3.

Step 5: For each T_i in T, $1\le i\le N$, if $T_i=k$, then $R_k=i$.

For example, after algorithm Amphisbaena- TSP is performed, we get the reordered list $R=(2,3,1,0,4)$ for the graph in Figure 1. This algorithm is efficient enough for our index compression problem, since we use the minimal edge of the graph as the initial edge and connect the minimal edge in each pass. It is clear that we would get the effect in the algorithm that the edges with less cost gather more near the kernel of the list. Although, the heuristic TSP algorithm is not the best, for the time efficiency, the algorithm can fulfill our demand.

2.2 Our Indexes Compression Algorithm

Although the neighboring blocks in an image have high correlation, the variation between two neighboring block indexes in VQ is usually not so small. Therefore, the reordering on the codewords of the codebook can be applied to reduce the variation between two neighboring block indexes. Then we can compress the reordered block indexes with a lossless compression method. Our reordering method improves the bit ratio, but keeps the same image quality.

Now, we shall present our TSP-VQ algorithm more precisely. We generate the occurrence matrix t. In Figure 2, v_k, n and w represent three neighboring block indexes and their correlative locations are also shown.

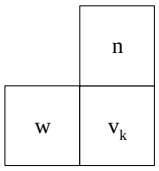


Figure 2: Labeling three neighboring blocks.

Each element $t(i,j)$ in matrix t represents the

number of occurrences that $v_k=i$, $n=j$ or $v_k=j$, $n=i$ or $v_k=i$, $w=j$ or $v_k=j$, $w=i$. Our goal is to reduce the difference of two neighboring block indexes. If two of the indexes appear in two neighboring blocks very frequently, then we have to renumber the indexes so that these two indexes are numbered nearly. So our problem is to find a longest tour to include all points of the set. But the TSP algorithm is to find a shortest tour to visit all points of the set. Hence, we first transform *t(i,j)* into *C(i,j)* to from the cost matrix *C* as follows:

$$C(i,j)=\frac{1}{1+t(i,j)}.$$

The TSP-VQ algorithm is given as follows.

Algorithm TSP-VQ

Input: The codebook, the VQ index list $V=(v_1,v_2,\cdots,v_M)$, and codebook size N.

Output: The compressed VQ index and the reordered codebook.

Step 1: Calculate the cost matrix,. $C(i,j)=(1)/(1+t(i,j))$, $1\le i,j\le N$, where $t(i,j)$ is the number of occurrences that $v_k=i$, $n=j$ or $v_k=j$, $n=i$ or $v_k=i$, $w=j$ or $v_k=j$, $w=i$.

Step 2: Apply Algorithm Amphisbaena-TSP to generate the reordered list R.

Step 3: For each v_i' in the reordered index list V', $1\le j\le M$, if $v_i=k$, then $v_i'=R_k$. Reorder the codebook according to R.

Step 4: Apply other lossless algorithms, such as CALIC and SPIHT, to compress V'.

Our algorithm adds the TSP process into the conventional vector quantization. The TSP algorithm can greatly reduce the difference of two neighboring block indexes. Our compression method with the TSP scheme is more efficient than that without TSP, such as CALIC and SPIHT.

3. Experiment Results and Performance Analysis

In this section, we shall illustrate the performance of several lossless compression methods with the TSP scheme. We establish one local codebook for each of the test images. All of these images are standard monochrome pictures; each is of 256 grey levels with resolution 512×512. We use the LBG [6] method to generate the codebook, and use the full search method to perform codeword searching in the codebook. We divide an image into a set of blocks with size 4×4 as 16-dimensional vectors. Both the sizes of the local and global codebook are 256. Since we have to store the codebook in the local VQ, the bpp (bits per pixel) for the standard local VQ is 0.5+0.125=0.625. While we do not store the codebook for the global VQ, the bpp for the standard global VQ is 0.5.

The quality of the reconstructed image with VQ is measured by the peak signal-to-noise (PSNR), which is defined as follows:

$$PSNR=10\log_{10}\frac{255^2}{MSE}dB,$$

where the mean square error (MSE) for an $L\times L$ image is defined as

$$MSE=\frac{1}{L^2}\sum_{i=1}^{L}\sum_{j=1}^{L}(w_{ij}-\hat{w}_{ij})^2,$$

where $L\times L$ is the size of image, w_{ij} is the pixel value of the original picture at coordinate (i, j), and $\hat{w}_{ij}$ is the pixel value of the reconstructed picture at coordinate (i, j).

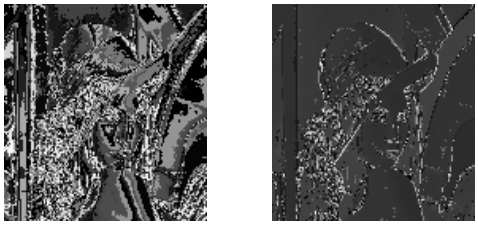
We also use two entropies to measure the performance in our experiments: self-entropy and differential entropy. Let E_k be the index value that occurs with probability $P(E_k)$, and E_{k-1} be the left neighbor index of E_k. Then the self-entropy of indexes is defined as

$$ENT=-\sum_k P(E_k)\log_2 P(E_K).$$

And the differential entropy in the horizontal direction of indexes is defined as

$$DENT=-\sum_k P(E_k-E_{k-1})\log_2 P(E_k-E_{k-1}).$$

In the local VQ, since the whole codebook is stored, we need not record any extra information about the reordered list. But in the global VQ, we need additional 256 bytes (2048 bits) for storing the reordered list.



(a) (b)

Figure 3: The index pattern in VQ for encoding Lena image. (a) without TSP. (b) with TSP.

Figure 3 show the index pattern in VQ for encoding Lena image without TSP and with TSP, respectively.

Table 1 and Table 4 show the performance (bpp) on the local and global codebooks with size 256, respectively. We use four lossless compression methods to encode the VQ indexes: the search-order coding (SOC) [3], set partitioning in hierarchical trees (SPIHT), context-based adaptive lossless image coding (CALIC) with arithmetic coding. The performance (bpp) is better if it is smaller. Both in the local and global VQ, the best performance on our experimental results is the VQ indexes encoded by CALIC+arithmetic with TSP.

Table 2 and Table 5 show the performance on the local and global codebook with size 256, respectively. We give the PSNR, entropy and differential entropy without and with TSP, respectively.

The self-entropy is the lower bound of bpp for the conventional variable-length coding (VLC), such as Huffman and arithmetic coding. The differential entropy can be viewed as the lower bound of bpp for the simple context-model coding. $P(E_k\text{-}E_{k-1})$ involves that E_k is estimated from E_{k-1}. This idea is similar to CALIC, which uses several neighboring pixels to estimate E_k. So, the bpp of the CALIC is less than the self-entropy and near the differential entropy. The entropy and differential entropy in Table 2 is associated with VQ indexes. The differential entropy is a kind of entropy dependent on the difference between two block index values. We use the TSP scheme to reduce the difference, so the differential entropy with TSP has significant reduction on the bpp. We give both of the differential entropy with TSP and without TSP in Table 2 and Table 5, respectively. Take the image Lena as an example, the bpp of the differential entropy with TSP in the local and global VQ has nearly 10% and 20% bpp decrement compared with those without TSP, respectively.

Table 3 and Table 6 show the percentage for the bpp (bits pre pixel) improvement between the CALIC+arithmetic with TSP and some other compression method on the local and global codebook with size 256, respectively. We illustrate the method without index compression, SOC, entropy and CALIC+arithmetic (without TSP). The bpp's of the SOC and CALIC+arithmetic (without TSP) for the local and global VQ are shown in Table 1 and Table 2, respectively. The bpp of the entropy and differential entropy of the local and global VQ are shown in Table 2 and Table 5, respectively. Take SOC in Table 3 for example, it is equal to *(SOC-CAL)/(SOC)*, where *SOC* is the bpp of SOC and *CAL* is the bpp of CALIC+arithmetic with TSP. So the percentage on each entry means the improvement of CALIC+arithmetic with TSP over the SOC. As another example in Table 3, the improvement of CALIC+arithmetic with TSP over CALIC+arithmetic without TSP on image Lena is 10.5%

Table 7 shows the performance (bpp and PSNR) on the global codebook with size 128. The address VQ [9] has better image quality since which uses PM/RVQ [9]. The bpp for CALIC+arithmetic with TSP is less than that for the address VQ. The address VQ has better image quality, but the size of the address VQ codebook is about 100000. It is too large.

We need store a codebook in the local VQ, since the local codebook is designed for a single image. So the bpp for the entropy of the local VQ is usually slightly greater than that of the global VQ. And the bpp improvement in the local VQ is less than that in global VQ. Our experimental results agree with this.

4. Conclusion

SOC is a simple lossless compression method for encoding VQ block indexes. In this paper, we propose a novel compression scheme for the block indexes. We first design a fast heuristic for solving the TSP problem. Although the algorithm can not obtain the optimal solution, it is time efficient. Its time complexity is $O(n^2)$, where n is the number of input points (also the size of the codebook). Then, we reduce the variation between two neighboring block indexes by the TSP scheme. Although in the global VQ, we need extra space to record the reordered list, the space for storing the reordered list is significantly smaller than those for the VQ indexes.

Experiment results also show that the bpp (bits per pixel) for our method combined with CALIC +arithmetic is less than SOC and the self-entropy. The TSP-VQ with CALIC+arithmetic reduces about 20% space than the entropy in the global VQ under the same image quality. The TSP scheme is also useful for the lossless compression methods, such as CALIC and SPIHT, which can reduce the bpp.

In the future, our method may be worth applying to other schemes, such as text compression and lossless compression.

Reference

[1] C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization", IEEE Transactions on Communications, Vol. C-33, No. 10, pp. 1132--1133, Oct. 1985.

[2] R. Bellman, “Dynamic programming treatment of the traveling salesman problem”, Journal of the ACM, Vol. 9, pp. 61-63, 1962.

[3] C. H. Hsieh and J. C. Tsai, “Lossless compression of vq index with search-order coding”, IEEE Transactions on Image Process, Vol. 5, pp. 1579-1582, Nov. 1996.

[4] M. C. Hung and C. B. Yang, “Fast algorithm for designing better codebooks in image vector quantization”, Optical Engineering, Vol. 36, No. 12, pp. 3265-3271, Dec. 1997.

[5] A. E. Jacquin, “Fractal image coding: a review”, IEEE Proceedings, Vol. 81, pp. 1451-1465, Oct. 1993.

[6] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design", IEEE Transactions on Communications, Vol. C-28, No. 1, pp. 84--95, Jan. 1980.

[7] J. D. C. Little, K. G. Mutry, D. W. Sweeney, and C. Karel, “An algorithm for the traveling salesman problem”, Operations Research, Vol. 11, pp. 972-989, 1963.

[8] B. Marangelli, “A vector quantizer with minimum visible distortion”, IEEE Transactions on Signal Processing, Vol. 39, No. 12, pp. 2718-2721, Dec.1991.

[9] N. M. Nasrabadi and R. A. King, “Image compression using address-vector quantization”, IEEE Transactions on Communications, Vol. 38, pp. 2166-2173, Dec. 1990.

[10] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees”, IEEE Transactions on Circuits and System For video Technology, Vol.

6, No. 3, pp. 243-250, June 1996.

[11] M. J. Weinberger, J. J. Rissanen, and R. B. Arps, "Application of universal context modeling to lossless compression of gray-scale images", IEEE Proceedings, Vol. 5, pp. 575-586, Apr. 1990.

[12] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding", IEEE Transactions on Communications, Vol. 45, No. 4, pp. 437-444, Apr. 1997.

[13] C. I. Yang, J. S. Wang, and R. C. T. Lee, "A branch-and-bound algorithm to solve the equal-execution-time job scheduling problem with precedence constraint and profile", Computers and Operations Research, Vol. 16, pp. 257-269, 1989.

Table 1: The performance (bpp) of various compression methods with the local codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.625.

Image	bpp				
	SOC	CALIC+ Arithmetic		SPIHT+ Arithmetic	
		Without TSP	With TSP	Without TSP	With TSP
Barb	0.543	0.558	0.493	0.623	0.547
Lena	0.532	0.572	0.512	0.622	0.571
Peppers	0.525	0.547	0.461	0.612	0.530
Sailboat	0.536	0.517	0.469	0.615	0.517

Table 2: The PSNR, entropy, differential entropy of TSP-VQ with the local codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.625.

Image	PSNR	Entropy	Differential entropy	
			Without TSP	With TSP
Barb	28.07	0.575	0.533	0.467
Lena	31.49	0.582	0.524	0.474
Peppers	32.34	0.543	0.495	0.437
Sailboat	29.38	0.533	0.532	0.489

Table 3: The percentage of the bpp improvement between TSP-VQ and various entropies. TSP-VQ uses CALIC+arithmetic coding with the local codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.625.

Image	Without index compression	SOC	Entropy	CALIC+ arithmetic (without TSP)
Barb	21.1%	9.2%	14.3%	11.6%
Lena	18.1%	3.8%	12.0%	10.5%
Peppers	26.2%	12.2%	15.1%	15.7%
Sailboat	25.0%	12.5%	12.0%	9.3%

Table 4: The performance (bpp) of various compression methods with the global codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.5.

Image	Bpp				
	SOC	CALIC+ Arithmetic		SPIHT+ Arithmetic	
		Without TSP	With TSP	Without TSP	With TSP
Barb	0.418	0.446	0.367	0.499	0.428
Lena	0.374	0.404	0.317	0.476	0.387
Peppers	0.378	0.397	0.303	0.481	0.386
Sailboat	0.412	0.438	0.355	0.438	0.411

Table 5: The PSNR, entropy, differential entropy of TSP-VQ with the global codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.5.

Image	PSNR	Entropy	Differential entropy	
			Without TSP	With TSP
Barb	26.78	0.440	0.428	0.352
Lena	30.87	0.414	0.368	0.300
Peppers	31.68	0.394	0.366	0.290
Sailboat	29.12	0.417	0.423	0.367

Table 6: The percentage of the bpp improvement between TSP-VQ and various entropies. TSP-VQ uses CALIC+arithmetic coding with the global codebook. Image size: 512 × 512, codebook size: 256, bpp without index compression: 0.5.

Image	Without index compression	SOC	Entropy	CALIC+ arithmetic (without TSP)
Barb	26.6%	12.2%	16.6%	17.7%
Lena	36.6%	15.2%	23.4%	21.5%
Peppers	39.4%	19.8%	23.1%	23.7%
Sailboat	29.0%	13.8%	14.9%	18.9%

Table 7: The performance (bpp) of various compression methods with the global codebook. Image size: 512 × 512, codebook size: 128, bpp without index compression: 0.437.

	Lena		Peppers	
	PSNR	bpp	PSNR	bpp
SOC	29.67	0.305	27.88	0.310
Address VQ	30.6	0.256	29.78	0.260
CALIC+ arithmetic	29.67	0.242	27.88	0.226
CALIC+	29.67	0.262	27.88	0.248

Huffman				