International Journal of Applied Science and Engineering
2015. 13, 2: 107-121

An Adaptive Data Hiding System Based on SMVQ

Po-Yueh Chen* and Wei-Shiang Chang

Department of Computer Science and Information Engineering,
National Changhua University of Education, Changhua, Taiwan, R.O.C.

Abstract: Based on Side-Match Vector Quantization (SMVQ), the authors propose two schemes to embed more secret information without significantly sacrificing the compression ratio. The data transform scheme improves the embedding capacity while maintaining the same compression ratio. The direct embedding scheme improves the embedding capacity at the expense of increasing the compression ratio slightly. According to various demands of a user, the integrated system can adaptively select a suitable method from the proposed ones. Experimental results show that the adaptive system provides the users with full flexibility and preserves a balance between the embedding capacity and compression ratio.

Keywords: Vector Quantization; data hiding; Side-Match Vector Quantization; adaptive

1. Introduction

Many image compression techniques, such as JPEG, JPEG2000, Vector Quantization (VQ), and Side-Match Vector Quantization (SMVQ) have been proposed recently [1-19]. They can be categorized into reversible [7-11] and irreversible [12-14] compression methods. Among these methods, the reversible VQ and SMVQ can be applied to the data hiding as well as data compression. Data hiding based on VQ/SMVQ [1-6, 9, 15-19] embeds the secret data in natural images to avoid malicious attacks.

In this paper, the authors propose two reversible data hiding schemes based on SMVQ. The idea proposed in [16] is adopted as the foundation of this research. First, image indices generated by SMVQ are divided into three portions, according to the index values. Indices in portion 1 are used for embedding the secret data whereas those in portion 2 and 3 are adopted for data compression and distortion recovery, respectively. However, the embedding capacity and the compression ratio are not generally optimal. Therefore, we propose an adaptive system which selects an appropriate scheme according to users demands on the embedding capacity. In other words, the proposed system attempts to provide a balance pair of embedding capacity and compression ratio according to various users' demands.

The rest of this paper is organized as follows. Section 2 briefly reviews the VQ, SMVQ and corresponding data hiding/compression methods. As the foundation of this study, Lee's method [16] is explained in details. Section 3 first describes the proposed methods and then provides a numerical example for illustrating the coding decoding process. To efficiently satisfy a user's embedding demand, an adaptive system is proposed as well. Experimental results are exhibited and analyzed in section 4. Finally, conclusions are summarized in section 5.

* Corresponding author; e-mail:pychen@cc.ncue.edu.tw

Received 27 September 2013
Revised 14 October 2014
Accepted 20 September 2014

2. Related works

Related theories/schemes are briefly reviewed in the following subsections. Interested readers are referred to the original literature for details.

2.1. Vector Quantization

Vector Quantization (VQ) is one of the common tools in image encoding. It uses an index to represent a pixel block and therefore greatly reduces the information amount of an image. The template blocks are stored in a codebook with corresponding indices. In general, the codebook is generated in advance by the LBG algorithm [20]. Figure 1 illustrates an example for the VQ encoding, with vector dimension of 16 and codebook size of 256. A 4 × 4 block (or a vector of size 16) in an image is encoded by searching the codebook for the most similar vector. The resulting index replaces the image block and provides substantial compression. As shown in Figure 1, index 1 and 253 are selected for representing the two plaid blocks because codeword 1 (CW1) and codeword 253 (CW253) resemble these two blocks, respectively. After all image blocks are processed, the original image is reduced to an image table as a result.

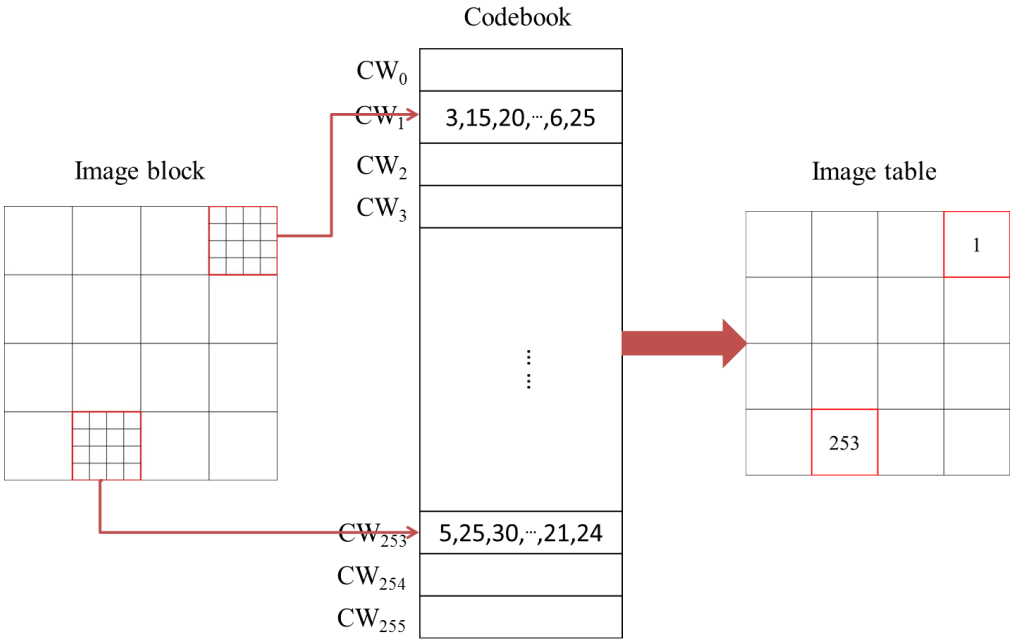

Figure 1. An example for the VQ encoder

2.2. Side-Match Vector Quantization

SMVQ [17], proposed by Kim in 1992, performs VQ on blocks in the first row and in the first column whereas the remaining blocks are predicted using their neighboring encoded blocks. It further improves the compression ratio as a result. Figure 2 demonstrates an example of SMVQ encoding. The top block U and the left block L are shaded, meaning that they are already encoded by VQ. Block X is then predicted as follows: $X_0 = (U_{12}+L_3) / 2$, $X_1 = U_{13}$, $X_2 = U_{14}$, $X_3 = U_{15}$, $X_4 = L_7$, $X_8 = L_{11}$, $X_{12} = L_{15}$. These seven pixel values are used to search for the closest (with the minimum Euclidean distance) codeword in the codebook. The index of that codeword

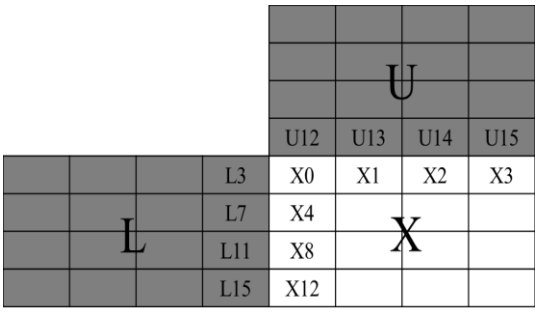

Figure 2. An example for the SMVQ

is then used for representing block X. Due to the reversibility of VQ and SMVQ, they can be applied to data hiding applications.

2.3. Chang's method

In 2007, Chang et al. proposed a VQ-based embedding method which can recover the VQ index table without any information loss [9]. The main goal is to expand the amount of embedding whereas maintaining the image quality is not an issue. Codebook is divided into numbers of clusters. Some bits of the index are set aside as an indicator, indicating that the VQ index table data is transferred from one cluster into another when data embedding occurs.

Assume there are N codewords in the original VQ codebook Ψ. Then there are $N' = \lfloor (N - 2^{B-1})/(2^{B-1} \times 3) \rfloor \times 2^{B-1} \times 3$ codewords in the modified codebook Ψ', where B is the number of secret bits to be embedded in one VQ index. Some images are adopted for training the codebook Ψ into Ψ' and the resulting codebook Ψ' is divided into $2^{B-1} \times 3$ clusters, each of size $m = \lfloor (N - 2^{B-1})/(2^{B-1} \times 3) \rfloor$. One third of the clusters, which have the highest frequencies, are applied for embedding the secret data. Figure 3 demonstrates an encoding example in [3] with 6-clusters C_0-C_5, where indices 1-10 in C_0 and C_1 are used to embed secret data. If the original index is 1 and the secret data to be embedded is "01", 21 is stored to record such an embedding. If the original index is 8 and the secret data to be embedded is "00", $(I_1)13$ is stored to record such an embedding. If the original index is 13, which belongs to cluster C_2, no secret data can be embedded and 3 is stored. Because indices 1-10 correspond to the 10 codewords with highest frequencies, the embedding capacity is generally satisfactory. Furthermore, the original image can be retrieved without loss since this embedding procedure is reversible.

2.4. Yang's method

Instead of embedding the secret data in the top third of the codebook, Yang et al. [15] adopted the first half of the codebook for embedding. As a result, the embedding capacity increases. This approach combines SMVQ with VQ to enhance the compression ratio. Figure 4 illustrates an example for the method proposed in [15] with 4-clusters C_0-C_3, where indices 1-14 in C_0 and C_1 are used for embedding secret data.. If the original index is 1 and the secret data to be embedded is "01", 8 is stored to record such an embedding. If the original index is 13 and the secret data to be embedded is "10", $(I_0)20$ is stored to record such an embedding. If the original index value is 17, no secret data can be embedded and $(I_1)17$ is stored.

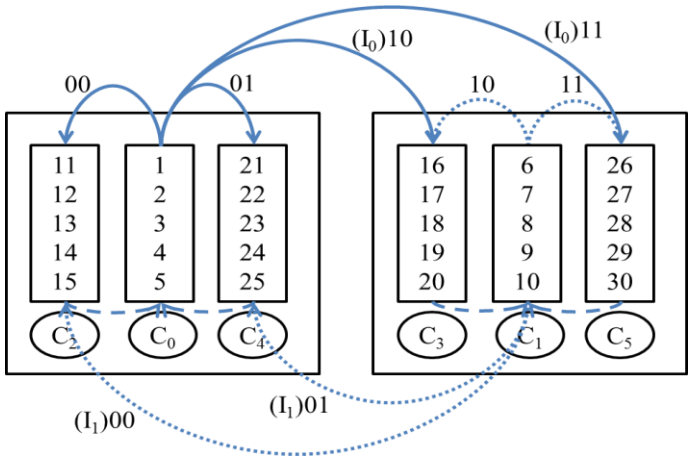

Figure 3. An example of Chang's method for embedding two secret bit in one index

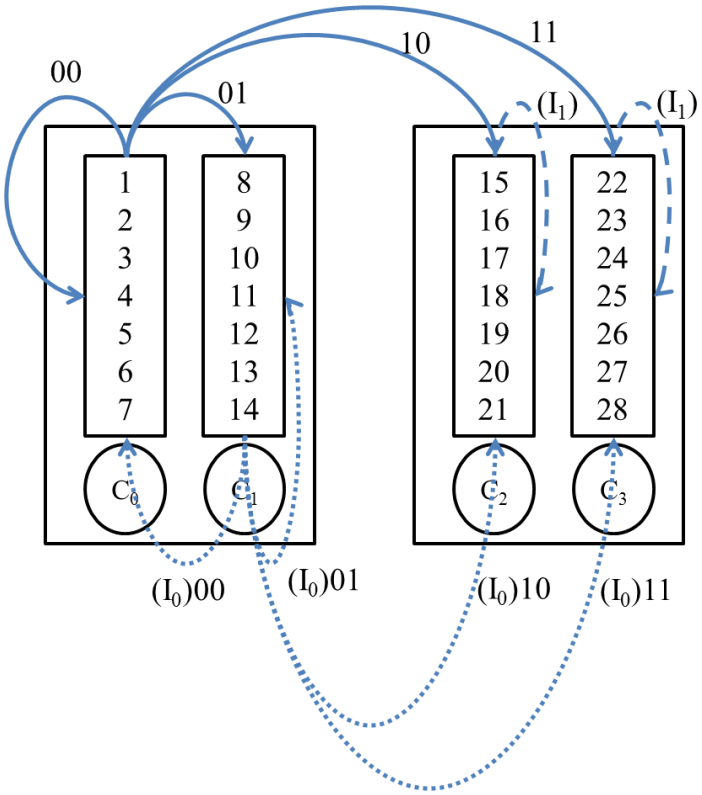

Figure 4. An example of Yang's method for embedding two secret bit in one index

2.5. Lee's method

Based on [9] and [15], Lee et al. proposed using the index histograms for further improvement [16]. Applying SMVQ can concentrate the histogram in the range of 0 to 15 and Lee exploits such a feature to hide the secret data. First, appropriately choose two thresholds n and m ($n<m<H$, H=256 is the size of the codebook) to divide the 256 indices into three portions. Specifically, indices which range in [0, n-1] are called Portion 1; the ones range in [n, m-1] are called Portion 2; and the ones range in [m, H-1] are called Portion 3. Indicator1 (0), indicator2 (10), and indicator3 (11) are used to mark the portion number.
Applying equation (1), indices in Portion 1 are modified for hiding secret information.

$$x_1' = \text{indicator1} \,\|\, x_1 \cdot 2^s + s_d \tag{1}$$

where x_1 and x_1' represent the original SMVQ index value and the modified SMVQ value respectively. The operators || and + mean cascading and arithmetic addition respectively. Basically, in addition to the 1-bit indicator, the s-bit secret message s_d just represents the LSBs of the resulting composite code x_1'. After embedding, each block in Portion 1 consumes (1 + E) bits, where E is the length of a coded SMVQ index.

Indices in Portion 2 are adopted for data compression. After coded by equation (2), each block in Portion 2 consumes $(2+\lceil \log_2(m-n) \rceil)$ bits, greatly reducing the length of the encoded data. Except for the portion indicator, indices in Portion 3 remain unchanged for data reconstruction. After coded by equation (3), each block in Portion 3 consumes (2+$\log_2 H$) bits.

$$x_2' = \text{indicator2} \,\|\, (x_2 - \text{n}) \tag{2}$$

$$x_3' = \text{indicator3} \,\|\, x_3 \tag{3}$$

Figure 5 shows the flow diagram of [16] where the last processing block named "Encoding" is described above by equation (1)-(3). Exploiting the merit of SMVQ, it improves both the compression ratio and embedding capacity compared with [9] and [15].

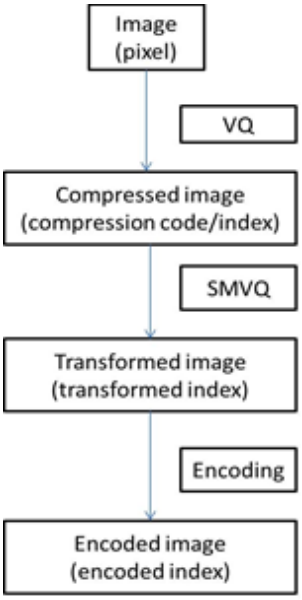

Figure 5. The flow diagram of Lee's approach

3. The proposed adaptive system

Figure 6 is the flow chart of the proposed adaptive system. The original image is compressed by VQ and the resulting indices are transformed by SMVQ. At this stage, two embedding schemes, the data transform and the direct embedding, are provided. According to an individual user's demands on embedding capacity and compression ratio, the system adaptively selects the appropriate one for data hiding. The following subsections describe the proposed schemes in details.

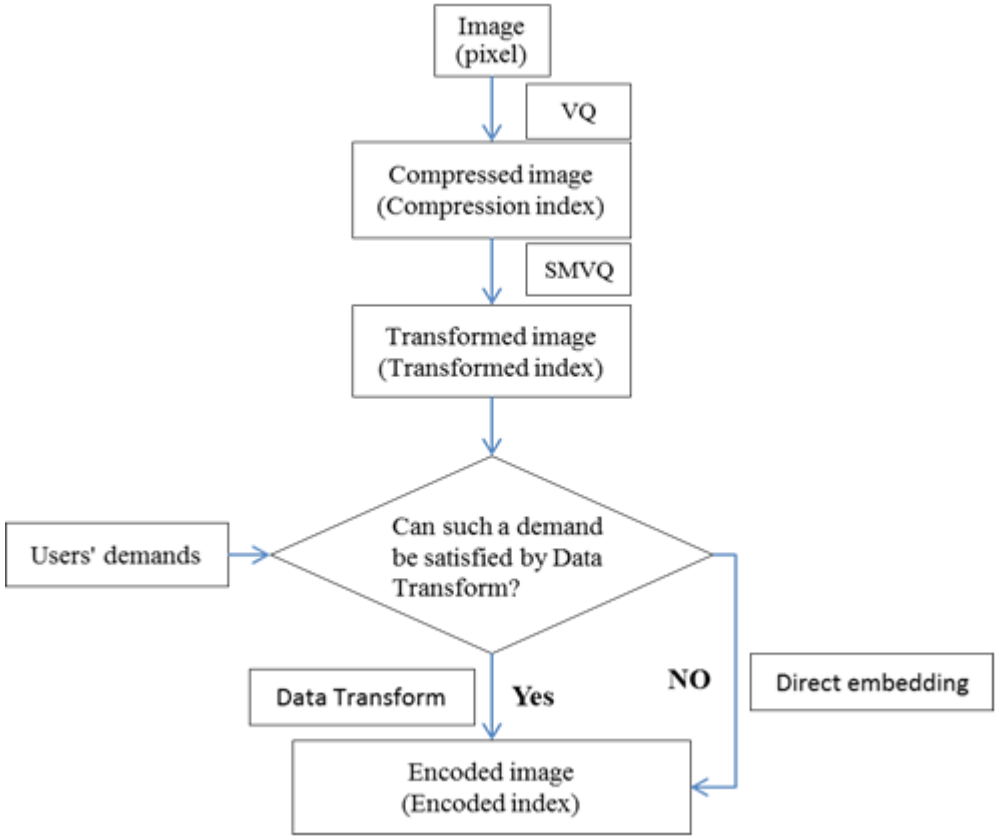

Figure 6. The flow chart of the proposed adaptive system

3.1. The data transform scheme

First, appropriately choose two thresholds n and m ($n<m<H$, H=256 is the size of the codebook) to divide the 256 indices into three portions. Specifically, indices which range in [0, n-1] are called Portion 1; the ones range in [n, m-1] are called Portion 2; and the ones range in [m, H-1] are called Portion 3. Indicator1 (0), indicator2 (10), and indicator3 (11) are used to mark the portion number. Indices in Portion 1 are adopted to hide information, applying equation (1) mentioned in subsection 2.5. In this stage, it is identical to Lee's method [16].

In [16], portion 2 is not used for data embedding. To improve the embedding capacity, the data transform scheme adopts the indices in Portion 2 for data hiding as well. The detailed processes are explained as follows. Subtract n from all indices in Portion 2 and then embed one bit in each index in the top third of Portion 2. Figure 7 shows an example of the embedding procedure for H=256. As shown in Figure 7, because n=4 and m=16 in this example, indices 4-15 are

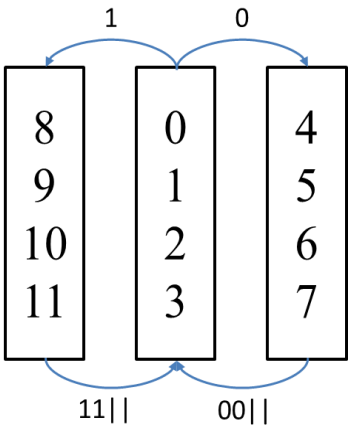

Figure 7. An example of embedding a one-bit secret data in Portion 2

transform scheme adopts the indices in Portion 2 for data hiding as well. The detailed processes are explained as follows. Subtract n from all indices in Portion 2 and then embed one bit in each index in the top third of Portion 2. Figure 7 shows an example of the embedding procedure for H=256. As shown in Figure 7, because n=4 and m=16 in this example, indices 4-15 are transformed into 0-11, respectively. Among these three clusters, Indices 0-3 are used for data embedding. If the embedded bit is 0, indices 0-3 will be transformed to 4-7, respectively. If the embedded bit is 1, indices 0-3 will be transformed to 8-11, respectively. Because the transformed result ranges from 4 to 11, four bits are adopted to record it. For indices 4-11, no secret data is embedded. Indices 4-7 are transformed to 0-3, respectively. Cascading "00" in front of the 2-bit result, it totally consumes four bits as well. Indices 8-11 are also transformed to 0-3, respectively. Cascading "11" in front of the 2-bit result, it totally consumes four bits as well.

For codebooks of various sizes, the configuration of data transformation is similar to the one shown in Figure 7. Figure 8 demonstrates the data transformation maps of four commonly used codebook sizes: 128, 256, 512, and 1024. According to [10], the optimal thresholds for these four cases are (n, m) = (2, 8), (4, 16), (8, 32), and (16, 64), respectively. In this paper, these threshold values are adopted for simulations. Indices in Portion 3 are encoded based on the identical process defined by equation (3).

3.2. The direct embedding scheme

Indices in Portion 1 are adopted to hide information, using equation (1). Thus far it is exactly the same as the data transform scheme. In order to reach the best balance between embedded capacity and compression ratio, we allow another alternative for data embedding in Portion 2, by using equation (4).

$$x_2' = \text{indicator2} \,\|\, x_2 \cdot 2^{s2} + s_d \tag{4}$$

where x_2 and x_2' represent the original SMVQ index value and the modified SMVQ value respectively. The operators || and + mean cascading and arithmetic addition respectively. Basically, in addition to the 2-bit indicator, the s2-bit secret message s_d just represents the LSBs of the resulting composite code x_2'. After embedding, each block in Portion 2 consumes (2 +E_2) bits, where E_2 is the length of a coded SMVQ index.

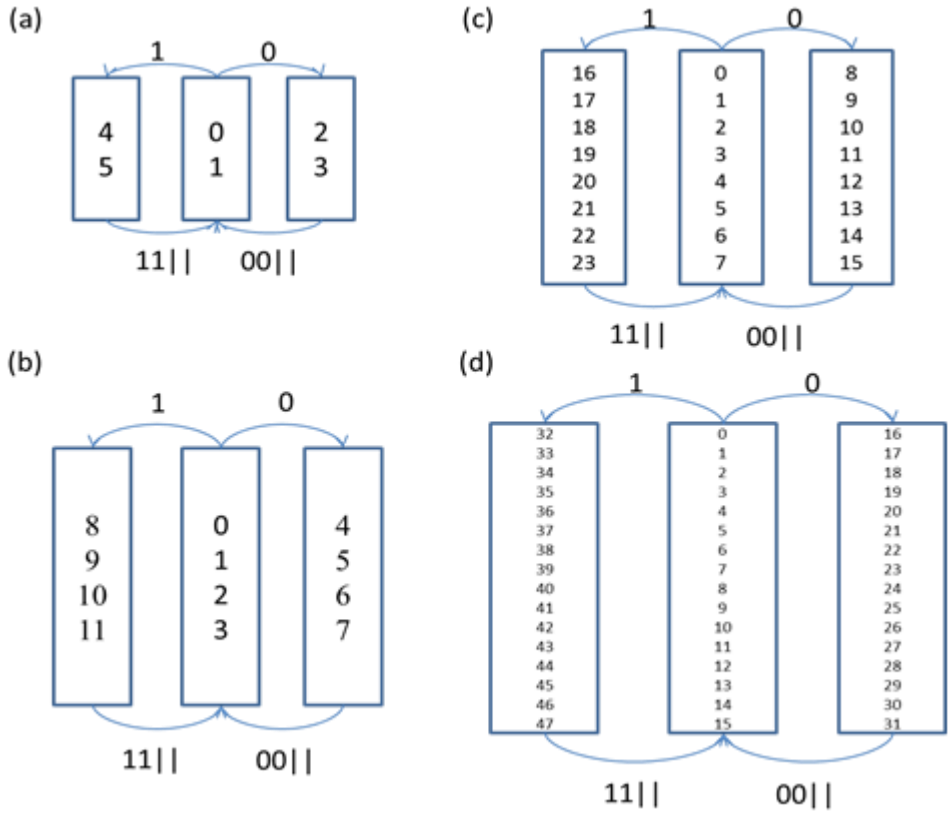

Figure 8. Data transformation for various codebook sizes: (a) H= 128; (b) H = 256; (c) H= 512; (d) H = 1024

However, because embedding data in Portion 2 induces some loss on the compression ratio, certain constraint must be set in advance. Assuming that there are α indices less than n, and β indices less than m but greater than or equal to n, s2 should be any positive integer less than or equal to α / β. This constraint is set to limit the code size and maintain a reasonable compression ratio. Indices in Portion 3 are coded using equation (3), the same procedure adopted in the data transform scheme.

3.3. A numerical example

Figure 9 illustrates an encoding example of using the data transform scheme, with H=256, n=4, m=16, s_1=3 (τ=5, in other words), and s_2=1. As shown in Figure 9(b), (d), and (e), the block indices in Portion 1 are dark-shaded and allow three bits to be embedded whereas the indices in Portion 2 are light-shaded and allow one bit to be embedded.

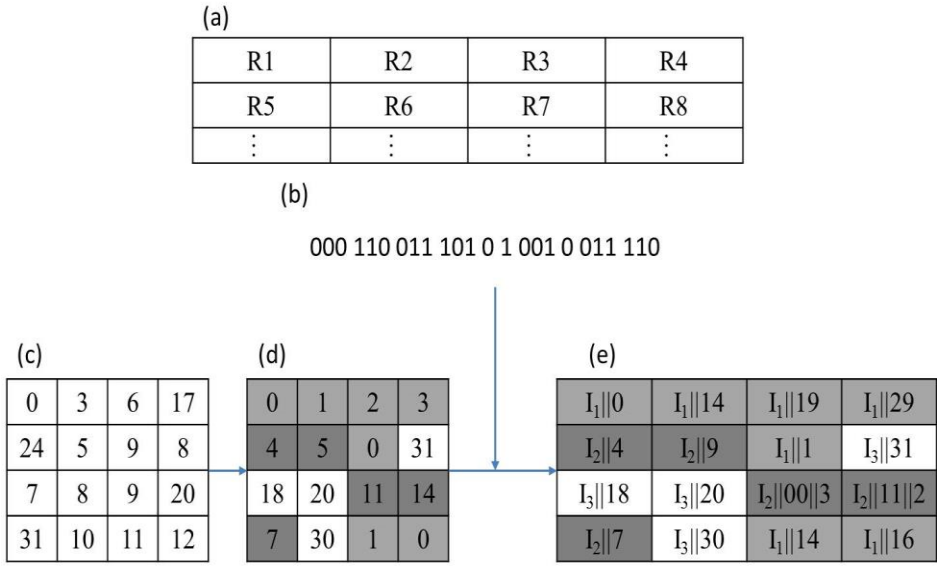

Figure 9. An example of the encoding process: (a) Image blocks; (b) Secret data; (c) VQ Compressed image; (d) SMVQ Transformed image; (e) Encoded image

For de-quantization of Portion 1, the original indices and the embedded data can be extracted using equations (5) and (6), respectively.

$$rx_1 = \left\lfloor \frac{x_1'}{2^{s1}} \right\rfloor \tag{5}$$

$$rsd_1 = x_1' \bmod 2^{s1} \tag{6}$$

where rx_1 and rsd_1 denote the decoded index and extracted data, respectively. The de-quantization for Portion 2 depends on which scheme is applied for data embedding. In this example, since the data transform scheme is applied, the decoding process described in Figure 7 should be performed. Figure 10 demonstrates the decoding results for this example. However, if the direct embedding scheme is applied for data embedding, the original indices and the embedded data can be extracted using equations (7) and (8), respectively.

$$rx_2 = \left\lfloor \frac{x_2'}{2^{s2}} \right\rfloor \tag{7}$$

$$rsd_2 = x_2' \bmod 2^{s2} \tag{8}$$

where rx_2 and rsd_2 denote the decoded index and extracted data, respectively.

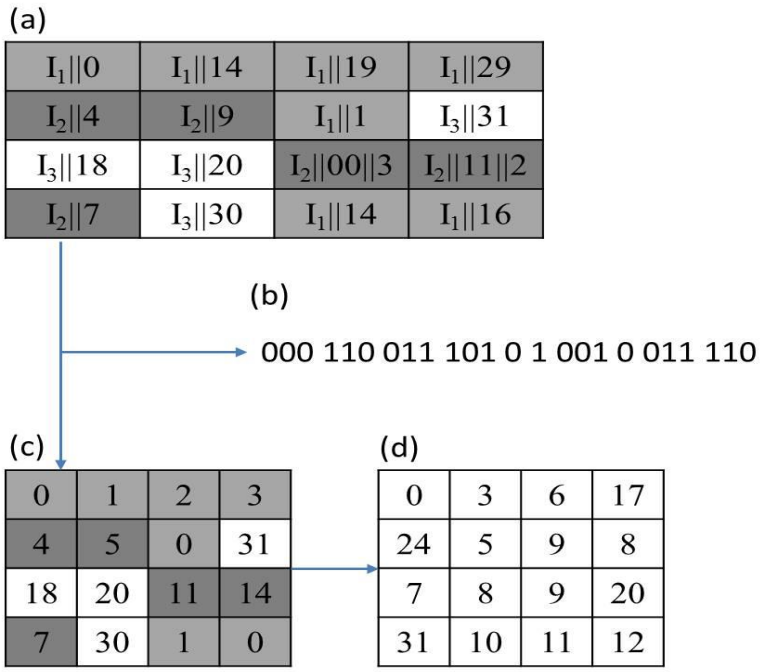

Figure 10. An example of decoding process: (a) Encoded image; (b) Secret data; (c) Transformed image; (d) Compressed image;

3.4. Parameters and Decision Strategy

Suppose there are α indices in Portion 1 and β indices in Portion 2. Among those β indices in Portion 2, γ of them possess an index value of the top third values. If the user demands an embedding amount of ε bits, the system calculates two parameters using equation (9) and (10).

$$s_1 = \left\lfloor \frac{\alpha}{\varepsilon} \right\rfloor \tag{9}$$

$$\phi = \varepsilon - \alpha s_1 \tag{10}$$

If ϕ is less than or equal to γ, the system selects the data transform scheme to embed S_1 bits in each index of those α ones in Portion 1 and embed one bit in each index of those γ ones in Portion 2. On the other hand, if ϕ is greater than γ, the direct embedding scheme is chosen to embed S_1 bits in each index in Portion 1 and embed S_2 bits in each index in Portion 2. In the latter case, S_2 can be calculated using equation (11).

$$s_2 = \left\lceil \frac{\phi}{\beta} \right\rceil \tag{11}$$

4. Experimental results

In this study, the five sample images shown in Figure 11 are adopted for codebook training. Four more sample images, Tiffany, House, Elaine, and Baboon, are adopted as four of the nine cover images.

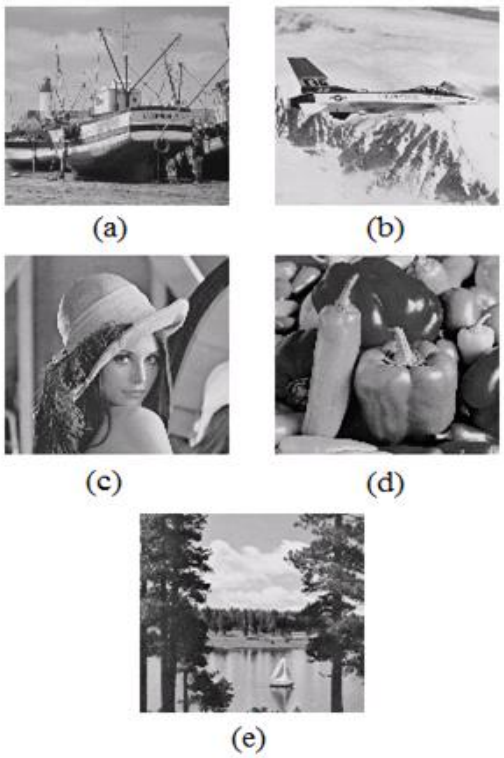

Figure 11. Samples for codebook training: (a) Boat; (b) Airplane; (c) Lena; (d) Pepper; (e) Sailboat

Because the data hiding schemes are reversible, Peak Signal Noise Ratio (PSNR) is not an issue herein. Two parameters, Embedded Capacity (EC, in bit) and Bit Rate (BR, in bit per pixel), are regarded for performance evaluation. The latter one is defined by equation (12).

$$\mathrm{BR} = \frac{N_b}{N_p} \tag{12}$$

where N_b and N_p represent the bit number of the encoded image and the pixel number of the original cover image. A superior embedding scheme provides higher EC and lower BR than other ones. Table 1 compares the performance of the data transform scheme with that of [16], setting codebook size H=256 and threshold (n, m) = (4, 16). By efficiently manipulating indices in Portion 2 for data embedding, the data transform scheme provides higher EC while maintaining the same BR.

Integrating the direct embedding scheme and applying the decision strategy explained in subsection 3.4, the adaptive system provides more alternatives for meeting the users' various embedding demands. As shown in Tables 2 and 3, competitive options are available and an individual user can select anyone of them, with great flexibility. For example, if Lena is adopted and the required EC is somewhere between 32322 bits (E=5) and 43096 bits (E=6), Lee's scheme [16] has no alternative but set E=6 at the expense of increasing BR from 0.408 to 0.449. On the other hand, for such a specific demand, the adaptive system provides five possible solutions with a BR smaller than or equal to 0.449. Based on the required EC and BR, a superior solution can be determined automatically.

Table 1. Performance provided by [16] and the data transform scheme

Image	Lee et al.'s[16]		The data transform scheme	
	EC (in bits)	BR (in bpp)	EC (in bits)	BR (in bpp)
Airplane	68676	0.539	70270	0.539
Lena	64644	0.531	66677	0.531
Peppers	67314	0.528	69479	0.528
Boat	56850	0.533	58770	0.533
Sailboat	55368	0.529	57567	0.529
Tiffany	77208	0.541	78543	0.541
House	52692	0.531	54628	0.531
Elaine	65112	0.520	67750	0.520
Baboon	64398	0.510	66702	0.510

Table 2. Performance provided by [16] and the adaptive system (Cover image=Lena)

Embedding schemes	BR	EC
Lee et al.'s (E =5)	0.408	32322
Data transform scheme (E_1 =5)	0.408	34355
Direct embedding scheme (E_1 =5; E_2 =5)	0.421	35788
Direct embedding scheme (E_1 =5; E_2=6)	0.434	39254
Direct embedding scheme (E_1 =5; E_2=7)	0.447	42720
Lee et al.'s (E=6)	0.449	43096
Data transform scheme (E_1=6)	0.449	45129
Direct embedding scheme (E_1=6; E_2=5)	0.462	46562
Direct embedding scheme (E_1=6; E_2=6)	0.475	50028
Direct embedding scheme (E_1=6; E_2=7)	0.488	53494

Table 3. Performance provided by [16] and the adaptive system (Cover image=Baboon)

Embedding schemes	BR	EC
Lee et al.'s (E =5)	0.387	32199
Data transform scheme (E_1=5)	0.387	34503
Direct embedding scheme (E_1=5; E_2=5)	0.406	37059
Direct embedding scheme (E_1=5; E_2=6)	0.424	41919
Lee et al.'s (E =6)	0.428	42932
Data transform scheme (E_1=6)	0.428	45236
Direct embedding scheme (E_1=6; E_2=5)	0.447	47792
Direct embedding scheme (E_1=6; E_2=6)	0.465	52652

Instead of giving an embedding demand ε and allowing the system to automatically select a feasible solution, the users can also choose a balanced (EC, BR) pair on their own. Setting E_1 from 5 to 6, Figure 12 shows the available solutions provided by the system, using Elaine as the cover image. In the figure, the data transform scheme and direct embedding scheme are denoted as scheme I and scheme II, respectively. Applying the proposed schemes, a user can choose a specific combination of (BR, EC) which meets his/her demand and set appropriate.

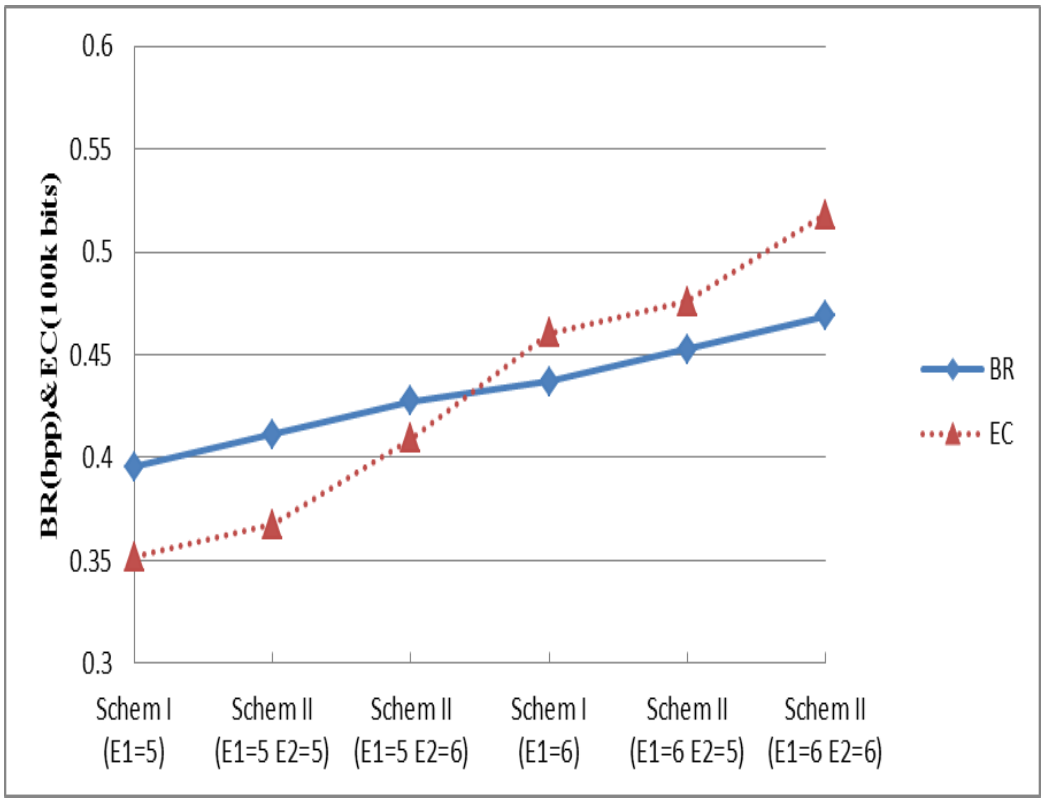

Figure 12. Performance of proposed schemes for image "Elaine"

Table 4 summarizes the simulation results for the four most commonly used codebook sizes. It ensures that the proposed system can be applied to various codebook sizes, providing improved (EC, BR) pairs. The improvements over [16] are more significant when the codebook size is 128 or 256. The performance declines when the codebook size is 512 or 1024 but is still slightly superior to that of [16]. If the size of state codebook for SMVQ is 16, all the VQ indices are supposed to concentrate within 0 to 15. However, with greater codebook sizes like 512 or 1024, there is no guarantee that the SMVQ indices are concentrated in either Portion 1 or 2.

5. Conclusions

In this paper, by efficiently manipulating the SMVQ indices in Portion 2, the authors propose two data embedding schemes. The data transform scheme maintains the same compression ratio and additionally increases the EC by embedding one bit in some indices of Portion 2. The direct embedding scheme further increases the EC at the expense of minor loss on the compression ratio. Integrating these two schemes, an adaptive system with a simple decision strategy is proposed to provide full flexibility in balancing the EC and BR requirements. According to the simulation results, the system outperforms the scheme proposed by Lee et al., especially for smaller codebook sizes. Because the computational complexity is low, for both parameter calculations and decision making, the system can be conveniently employed in many real time applications.

Table 4. Performance provided by the adaptive system for various codebook sizes

Cover Images	Codebook size H			
	128	256	512	1024
	Threshold (n, m)			
	(2,8)	(4,16)	(8,32)	(16,64)
	EC(in bits) / BR(in bpp)			
Airplane	33192	35932	38874	38745
	0.353	0.408	0.478	0.552
Lena	31520	34355	36453	36009
	0.35	0.407	0.485	0.566
Peppers	32606	35822	38351	38178
	0.345	0.399	0.472	0.552
Boat	28423	30345	32804	32220
	0.368	0.424	0.504	0.585
Sailboat	27704	29883	31443	30609
	0.367	0.423	0.508	0.594
Tiffany	32347	39939	45051	44829
	0.367	0.393	0.452	0.521
House	26805	28282	31485	30519
	0.373	0.43	0.51	0.594
Elaine	31772	35194	38521	37116
	0.338	0.395	0.471	0.561

6. References

[1] Lin, C.C. and Zhang, X.B. 2012. A High Capacity Reversible Data Hiding Scheme Based on SMVQ, *International Conference on Genetic and Evolutionary Computing*, 169-172.

[2] Chen, L.S.T., Chang, W.T., Lee, S.S. and Lin, J.C. 2010. Data Hiding Based on Side Match Vector Quantization and Modulus Function, *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 66-69.

[3] Rahmani, P., Dastghaibyfard, G. and Rahmani, E 2011. A Reversible Data Embedding Scheme Based on Search Order Coding for VQ Index Tables, *International ISC Conference on Information Security and Cryptology*, *79-82.*

[4] Lee, C.F., Chen, H.L., and Lai, S.H. 2010. An Adaptive Data Hiding Scheme with High Embedding Capacity and Visual Image Quality Based on SMVQ Prediction through Classification Codebooks, *Image and Vision Computing*, 8: 1293-1302.

[5] Yang, C.H., Wang, W.J., Huang, C.T. and Wang, S.J. 2011. Reversible Steganography Based on Side Match and Hit Pattern for VQ-compressed Images, *Information Sciences*, 181, 11: 2218-2230.

[6] Shie, S.C. and Jiang, J.H. 2012. Reversible and High-payload Image Steganographic Scheme Based on Side-match Vector Quantization, *Signal Processing*, 92, 9: 2332-2338.

[7] Chang, C.C., Chen, G.M. and Lin, M.H. 2004. Information Hiding Based on Search-order Coding for VQ Indices, *Pattern Recognition Letter* 25, 11: 1253-1261.

[8] Chang, C.C., Hsieh, Y.P., and Lin, C.Y. 2007. Lossless Data Embedding with High Embedding Capacity Based on Declustering for VQ-compressed Codes, *IEEE Transaction on Information and Forensics Security*, 2, 3: 341-349.

[9] Chang, C.C., Wu, W.C., and Hu, Y.C. 2007. Lossless Recovery of a VQ Index Table with Embedded Secret Data, *Journal of Visual Communication and Image Representation*, 18, 3: 207-216.
[10] Chang, C.C., Kieu, T.D., and Chou, Y.C. 2009. Reversible Information Hiding for VQ Indices Based on Locally Adaptive Coding, *Journal of Visual Communication and Image Representation*, 20, 1: 57-64.
[11] Tian, J. 2003. Reversible Data Embedding Using a Difference Expansion, *IEEE Transaction on Circuits and Systems for Video Technology*, 13, 8: 890-896.
[12] Shie, S.C., Lin, S.D.,and Fang, C.M. 2006. Adaptive Data Hiding Based on SMVQ Prediction, *IEICE Transaction on Information System*, E89-D, 1: 358-362.
[13] Wu, M.N., Lin, C.C. and Chang, C.C. 2008. An Embedding Technique Based upon Block Prediction, *Journal of Systems and Software*, 81, 9: 1505-1516.
[14] Du, W.C. and Hsu, W.J. 2003. Adaptive Data Hiding Based on VQ Compressed Images, *Proc. Inst. Elect. Eng., Vision, Image and Signal Processing*, 150, 4: 233-238.
[15] Yang, C.H. and Lin, Y.C. 2009. Reversible Data Hiding of a VQ Index Table Based on Referred Counts, *Journal of Visual Communication and Image Representation*, 20, 6: 399-407.
[16] Lee, J.D., Chiou, Y.H., and Guo, J.M. 2010. Reversible Data Hiding Based on Histogram Modification of SMVQ Indices, *IEEE Transaction on Information and Forensics Security*, 5, 4:638-648
[17] Kim, T. 1992. Side Match and Overlap Match Vector Quantizers for Images, *IEEE Transaction on Image Processing*, 1, 2: 170-185.
[18] Chang, C.C., Tai, W.L. and Lin, C.C. 2006. A Reversible Data Hiding Scheme Based on Side Match Vector Quantization, *IEEE Transaction on Circuits and Systems for Video Technology*, 16, 10: 1301-1308.
[19] Tsai, P. 2009. Histogram-based reversible data hiding for vector quantization-compressed images, *IET Image Processing*, 3, 2: 100-114.
[20] Linde, Y., Buzo, A., and Gray, R. 1980. An Algorithm for Vector Quantizer Design, *IEEE Transaction on Communications*, 28, 1: 84-95.