Journal of Information Hiding and Multimedia Signal Processing ©2013 ISSN 2073-4212
Ubiquitous International Volume **4**, Number **4**, October **2013**

An Indicator Elimination Method for Side-match Vector Quantization

Chin-Chen Chang[1,2]

[1]Department of Information Engineering and Computer Science,
Feng Chia University
No. 100, Wenhwa Road, Taichung, 40724, Taiwan
alan3c@gmail.com

Yung-Chen Chou[2] and Chih-Yang Lin[2]

[2]Department of Information Engineering and Computer Science,
Feng Chia University
No. 500, Lioufeng Road, Taichung, 41354, Taiwan
yungchen@asia.edu.tw; andrewlin@asia.edu.tw

Received March, 2013; revised May, 2013

Abstract. *The vector quantization (VQ) concept is widely used in many applications. Side-match vector quantization (SMVQ) is a VQ-based image compression method that offers significantly im- proved performance of compression rate while maintaining the image quality of decompressed images. To eliminate distortion propagation, SMVQ requires one extra bit to serve as an indicator identifying which blocks are encoded by SMVQ or VQ, and to make sure all image blocks can be successfully recon- structed. To eliminate the indicators generated by SMVQ, a reversible data hiding method is adopted to conceal indicator into the compression code. From experimental results, the proposed method success- fully conceals indicators into compression code with similar visual quality performance to SMVQ. In addition, experimental results confirm that the proposed method significantly improves the performance of compression rate.*
Keywords: Data hiding, declustering, indicator elimination, Side-match vector quantization (SMVQ), vector quantization (VQ)

1. **Introduction.** Because of advanced development of information techniques, creating digital images or pictures is easier than before. Many compression techniques have been presented to significantly reduce the size of digital images saving storage space and transmission bandwidth. Image compression techniques can be briefly classified into two types: 1) spatial domain compression methods [1, 2, 3], and 2) frequency domain compression methods [4, 5, 6]. The target of spatial domain compression method is pixels of an image. On the other hand, the target of frequency domain compression method is coefficients of a transformed image. Spatial domain compression method is more efficient than frequency domain compression method in terms of computation cost because spatial domain compression method needs not to transform into frequency domain. In other words, spatial domain compression method is more suitable for low power environment (e.g., wireless networks, PDAs). In addition, spatial domain compression methods are usually easily implemented in both software and hardware [7, 8].

Vector quantization (VQ) [2] is one of the most powerful spatial domain image compression methods. VQ is a lossy compression method that uses codeword indices to represent images, where the codewords are taken from a common codebook that contains the proper number of codewords. When an image is to be compressed by the VQ system, the image is first partitioned into non-overlapping blocks and each block is mapped to the closest codeword in the codebook. The compression bit rate (often abbreviated to "compression rate") of VQ is determined by the size of the codeword and the codebook. For example, if an image sized 512×512 and a codebook consists of 256 codewords and each codeword contains 16 pixels, then the total bits of compressed image is $128 \times 128 \times 8 = 131,072$ (i.e., the total bits of original image is 2,097,152.)

VQ compression method is a powerful and simple image compression method. However, the perfor- mance of compression rate can be further improved. Therefore, many improved versions that consider the relationships among neighboring blocks have been proposed; for example, search-order coding (SOC) [3], locally adaptive scheme (LAS) [9], side-match vector quantization (SMVQ) [10], variable-rate SMVQ with a block classifier (CSMVQ) [11], and pattern-based SMVQ (PSMVQ) [12]. The SOC method uses previous index values to predict the current index value. The LAS scheme adopts the Lempel-Ziv [13] approach to make predictions more efficient. The SMVQ method predicts an input block by using the adjacent values from its upper and left blocks to form a temporary codeword. The temporary codeword is used to select a part of codewords from the main codebook to form a state codebook. Normally, the size of state codebook is smaller than the main codebook. Thus, using state codebook to encode a block can further reduce the size of compression code. In order to solve the ambiguity for encoding a block, an indicator is required to identify the source of encoding codeword. In other words, if the indicator is set as '0', then the block is encoded by using a codeword from the main codebook; otherwise (i.e., indicator is set as '1'), the block is encoded by using a codeword from the state codebook.

The CSMVQ and PSMVQ methods are the variants of SMVQ. Although all of these schemes use different techniques, they all require an indicator in each input block to indicate whether the block can be correctly predicted. These indicators are not a part of the data; instead, they are a part of the compressed codes and thus impair the compression rate. Eliminating these indicators will further improve the compression rate. For example, a grayscale image sized 256×256 pixels is divided into 4×4 non-overlapping blocks and encoded by SMVQ then 4096 ($= 64 \times 64$) blocks require 3969 ($= 63 \times 63$) bits to serve as indicators. From experimental results, the proposed method can successfully eliminate 3906 bits.

Data hiding technique is to conceal secret data into a cover image for achieving the goal of secret data delivery [14, 15, 16, 17, 18, 19]. We inspired with Chang and Lin's method [20] to discover that the data hiding technique not only can be used to delivery secret data but also can be applied to improve the performance of image compression method in terms of compression rate. In this paper, an indicator eliminating method is presented to improve the performance of SMVQ in terms of compression rate. The proposed indicator elimination method is based on the concept of declustering, which gathers dissimilar data as a pair. The declustering concept is different from clustering because the clustering concept is to gather similar data (small distance between two data points) as a pair but declustering didn't. The proposed indicator elimination method embeds the indicator of each block into its preceding block's index. Thus, the extra bits for recording the indicators are eliminated. From the experimental results, the compression rate of compressed image generated by VQ, SMVQ, and the proposed indicator elimination method are 0.0625,

0.01105, and 0.00468, respectively. The proposed method has better performance of compression rate than VQ and SMVQ.

The rest sections are organized as follows. First, the related works are briefly reviewed in Section 2. Section 3 presents the proposed indicator elimination method in detail. Experimental results are demonstrated in Section 4. Finally, some concluding remarks are stated in Section 5.

2. **Related Works.**

2.1. **Vector Quantization.** Vector quantization (VQ) [2] is a well-known lossy image compression scheme in spatial domain that is valued for its simple yet efficient encoding and decoding processes. The VQ compression method needs a common codebook before compression can be applied. Assume that a codebook contains M codewords and each codeword has $k \times k$ dimensions. When an image I is to be encoded, it is first partitioned into non-overlapping blocks sized $k \times k$ pixels (i.e., the same dimensions as the codeword). Then, every block is encoded by a closest codeword selected from the code book. The closest codeword represents that the codeword has minimal distortion value in comparison with the current encoding block. The closest codeword cwmin can be found by using Eq. 1.

$$cw_{\min} = \min \left\{ d\left(B_i, cw_x | x = 1, 2, \ldots, M\right) \right\}, \tag{1}$$

$$d\left(B_i, cw_x\right) = \sqrt{\sum_{j=1}^{k \times k} \left(B_i\left(j\right) - cw_x\left(j\right)\right)^2}, \tag{2}$$

where B_i represents current encoding block of I, cw_x is the x-th codeword in the codebook, and $B_i(j)$ and $cw_x(j)$ are the j-th pixel values of B_i and cw_x, respectively. M represents the total number of codewords of main codebook. $d(B_i, cw_x)$ represents the distance function that measures the distortion between block B_i and codeword cw_x. When a codeword is the smallest distance from B_i, it is the closest codeword for B_i. After the closest codeword cw_{min} has been found, the index of cw_{min} is used to encode B_i. Therefore, through the VQ encoding process, the original image is eventually represented by the indices of these closest codewords.

During decoding, the VQ decoder uses the same codebook as the encoder. For each incoming index, the decoder requires only a table lookup operation to acquire the corresponding codeword from the codebook in order to reconstruct the image block. Therefore, the VQ decoding process is much more efficient than the VQ encoding process, and the VQ-compressed image quality is determined by the quality of the codebook.

2.2. **Side-match Vector Quantization.** The SMVQ method [10] is to improve the performance of VQ in terms of compression rate and to alleviate block effect. Because the characteristic of natural images (i.e., the smooth area has similar pixel distribution), SMVQ encodes a current block by a codeword selected from a state codebook. Here, the state codebook is constructed by selecting some codewords from the main codebook. Based on this concept, in SMVQ the first row and first column blocks are encoded using the traditional VQ process, and the residual blocks are encoded by VQ or SMVQ.

The first row and first column blocks must be encoded accurately because these blocks are used to predict the residual blocks. If an error occurs in the encoding phase, it will propagate throughout the entire image. In essence, the main codebook is generated by LBG algorithm [21] just as with traditional VQ, and the sender and receiver share the

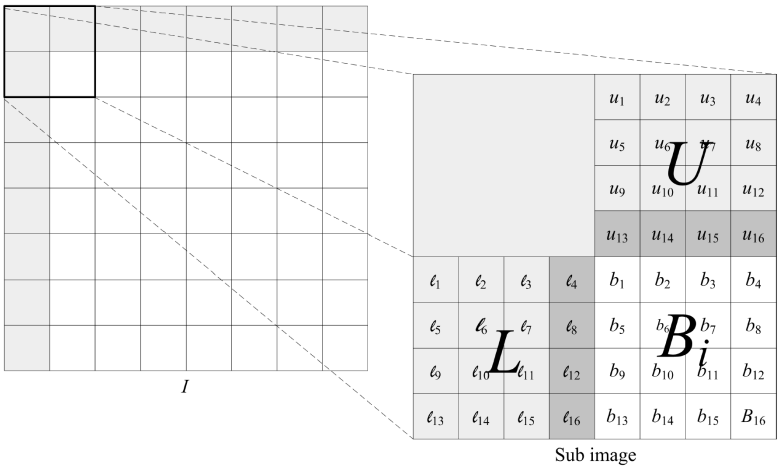


FIGURE 1. A diagram of SMVQ

same main codebook during the encoding and decoding phases. In case, if a block can be encoded by SMVQ, then a codeword with the smallest distance from the current encoding block is chosen from the state codebook to represent the current block. Obviously, the state codebook can be different for each block in an image. Because the sender and receiver have the same main codebook and the state codebook generating procedure, thus the decoding can successfully reconstruct the image.

Fig. 1 is a diagram for SMVQ. Bi is the current encoding block and U and L are two neighboring blocks of B_i. Because SMVQ encodes the image by raster scan order, blocks U and L are encoded before B_i. The boundary pixel values of blocks U and L (i.e., u_{13}, u_{14}, u_{15}, u_{16}, ℓ_4, ℓ_8, ℓ_{12}, and ℓ_{16}) are used to select S possible codewords from the main codebook to form a state codebook for the block B_i according to their side-match distortions (SMDs). Here, S represents the number of codewords in a state codebook. The SMD is defined as follows:

$$SMD(B_i, cw_j) = \begin{array}{l}((\frac{u_{k^2-k+1}+\ell_k}{2} - cw_j(1))^2 + \sum_{q=k^2+k-1}^{k^2}(u_q - cw_j(q))^2 \\ + \sum_{q=2}^{k}((\ell_{q\times k} - cw_j(q\times k))^2\end{array}, \tag{3}$$

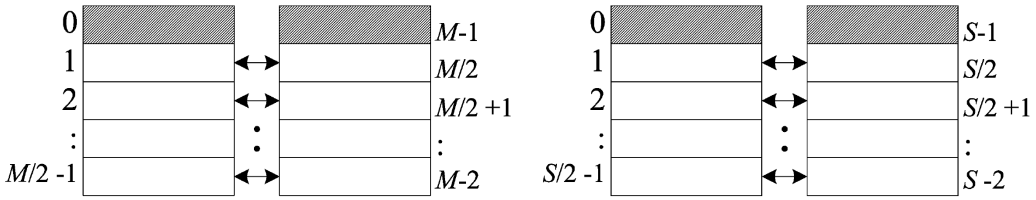
where cw_j is j-th codeword in the codebook.

Generally, the size of the state codebook is smaller than the size of the main codebook. The S possible codewords are having the smallest SMD values. If B_i can be encoded by applying $SMVQ$, then the length of compression code of B_i can be reduced from $[\log_2 M]$ to $[\log_2 S]$. Here, M and S represent the number of codewords in the main codebook and state codebook, respectively.

SMVQ has a better compression rate than VQ, but it may lead to poor visual quality owing to the derailment problem. Derailment may occur when the pixel distribution of an encoded block is very dissimilar from that of its neighboring blocks. A general solution to derailment is to use traditional VQ instead of SMVQ to encode these improperly predicted blocks. In this case, an extra bit is required to indicate that a block is encoded by traditional VQ or SMVQ to make sure all blocks except those located in the first row and column can be correctly decoded later.

3. **The Proposed Method.** As mentioned above, the indicator is used to indicate that a block is encoded by SMVQ or VQ. However, the performance of SMVQ in terms of

compression rate could be damaged on account of indicator. In this paper, an indicator elimination method is proposed to improve the compression rate of SMVQ by using reversible data hiding technique.



(a)Declustering for main codebook (b) Declustering for state codebook

FIGURE 2. A declustering diagram for main codebook and state codebook

The proposed method is inspired by Chang and Lin's method [20], which is designed to conceal secret data into a VQ compression code. The proposed method also applies Chang and Lin's declustering concept to achieve the goal of SMVQ compression rate improvement.

3.1. **Declustering.** The basic idea of the declustering is to classify two most dissimilar codewords into a pair. Let a main codebook $MCB = \{mcw_i | i = 0, 1, \ldots, M-1\}$ contains M codewords generated by LBG algorithm [21]. First, the mean value of mcw_i is computed by using Eq. 4.

$$\mu_i = \frac{\sum_{j=1}^{k^2} mcw_i(j)}{k^2}, \tag{4}$$

where μ_i represents the mean value of i-th codeword in MCB and $mcw_i(j)$ is the j-th component of code-word mcw_i. The dimension of a codeword is k^2. Then, M codewords in the MCB are sorted in ascending order by μ_i. In the sorted MCB, any to neighboring codewords are similar to each other [22, 23, 24]. Next, all of codewords in the MCB are paired except mcw_0 and mcw_{M-1}. Here, mcw_0 and mcw_{M-1} are reversed for handling special cases. The paring rule is (mcw_i,mcw_j) where $i = 1, 2, \ldots, \frac{M}{2} - 1$ and $j = \frac{M}{2}, \frac{M}{2} + 1, \ldots, M - 2$. That is, the paired codewords are $\{(mcw_1, mcw_{\frac{M}{2}}), (mcw_2, mcw_{\frac{M}{2}+1}), \ldots, (mcw_{\frac{M}{2}-1}, mcw_{M-2})\}$. According the pairing strategy, two very dissimilar codewords are paired in the same set.

Because declustering is the foundation of our proposed indicator elimination method; therefore, the declustering strategy also performs a state codebook (denoted as SCB). The only difference is that the SCB is sorted by SMD values of codeword in ascending order. That is, the declustering strategy is applied when a SCB has been generated. Assume that an SCB contains S codewords $\{scw_0, scw_1, \ldots, scw_{S-1}\}$. Again, the scw_0 and scw_{S-1} in the SCB are reserved for handling the special cases. Finally, the dissimilar codewords in a SCB are paired to form $\frac{S-2}{2}$ pairs: $(scw_1, scw_{\frac{S}{2}}), (scw_2, scw_{\frac{2}{2}+1}), \ldots$, and $(scw_{\frac{S}{2}-1}, scw_{S-2})$, respectively.

Fig. 2 is an example to demonstrate the declustering procedure of the main codebook and state codebook. The codewords in gray means the codewords are reversed for handling special cases.

3.2. **Indicator Elimination.** The proposed indicator elimination method for SMVQ utilizes declustring strategy to conceal the indicators into compression code. For simplicity, cw_{min} and $\overline{cw_{\min}}$ represent the codeword closest to current encoding block and the

codeword paired with cw_{min}, respectively. Thus, if the closest codeword is taken from MCB, then $cw_{min} = mcw_{min}$ and $\overline{cw_{\min}} = mcw_{\min}$. Furthermore, if the closest codeword is found from the state codebook, SCB, then $cw_{min} = scw_{min}$ and $\overline{cw_{\min}} = \overline{scw_{\min}}$

Referring to the declustering results, $(cw_{\min}, \overline{cw_{\min}})$ is to pair two most dissimilar codewords in a pair. Because pixel distribution of a local area has similar distribution, the $SMD(B_i, cw_{min})$ is always smaller than the $SMD(B_i, \overline{cw_{\min}})$ calculated using Eq. 3. Based on this important property, the proposed indicator elimination process is carried out as follows.

First, an image I is divided into non-overlapping blocks sized $k \times k$ pixels and represented as $I = \{B_i | i = 1, 2, \ldots, N\}$, where N is the total number of blocks of I, then the blocks stay in the first column and first row are the seed blocks and encoded them by using VQ. Next, each residual block B_i is encoded by using SMVQ. For a current encoding block B_i, apply SMVQ to the source (i.e., the main codebook or state codebook) of encoding codeword. As mentioned above, if B_i can be encoded by using a codeword from the state codebook than B_i's indicator is '1', else, B_i's indicator is '0'.

In the proposed method, a residual block is one of following three types. **Type I:** B_i is located in the second column of the image I. Because B_i's preceding block is a seed block, it served as a reference role for the following encoding and cannot be modified; therefore, an extra bit is required to record the indicator of B_i. **Type II:** B_i belongs to the last column of the image I. B_i is encoded by mcw_{min} or scw_{min} because no block is succeeded by B_i. **Type III:** B_i does not belong to **Type I** and **II**, then B_i is encoded by using one of the following eight cases. The notation $IND(B_i)$ is to represent the indicator of B_i.

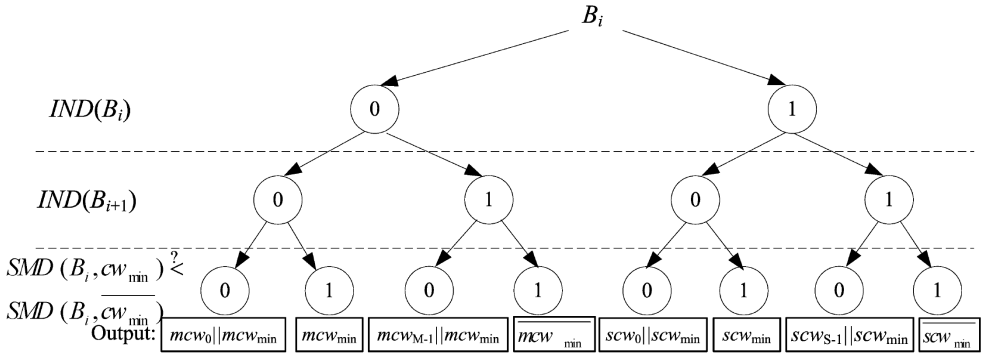


FIGURE 3. Diagram for eight cases for encoding B_i

Case 000: If $IND(B_i) = IND(B_{i+1}) =$ '0', and $SMD(B_i, mcw_{\min}) \geq SMD(B_i, \overline{mcw_{\min}})$, then B_i is encoded by $mcw_0||mcw_{\min}$, where "||" denotes the concatenation operation.

Case 001: If $IND(B_i) = IND(B_{i+1}) =$ '0', and $SMD(B_i, mcw_{\min}) < SMD(B_i, \overline{mcw_{\min}})$, then B_i is encoded by mcw_{min}.

Case 010: If $IND(B_i) =$ '0', $IND(B_{i+1}) =$ '1', and $SMD(B_i, mcw_{\min}) \geq SMD(B_i, \overline{mcw_{\min}})$, then B_i is encoded by $mcw_{M-1}||mcw_{\min}$.

Case 011: If $IND(B_i) =$ '0', $IND(B_{i+1}) =$ '1', and $SMD(B_i, mcw_{\min}) < SMD(B_i, \overline{mcw_{\min}})$, then B_i is encoded by $\overline{mcw_{\min}}$.

Case 100: If $IND(B_i) =$ '1', $IND(B_{i+1}) =$ '0', and $SMD(B_i, scw_{\min}) \geq SMD(B_i, \overline{scw_{\min}})$, then B_i is encoded by $scw_0||scw_{\min}$.

Case 101: If $IND(B_i) =$ '1', $IND(B_{i+1}) =$ '0', and $SMD(B_i, scw_{\min}) < SMD(B_i, \overline{scw_{\min}})$, then B_i is encoded by scw_{min}.

Case 110: If $IND(B_i) = IND(B_{i+1}) =$ '1', and $SMD(B_i, scw_{\min}) \geq SMD(B_i, \overline{scw_{\min}})$, then B_i is encoded by $scw_{M-1}||scw_{\min}$.

Case 111: If $IND(B_i) = IND(B_{i+1}) =$ '1', and $SMD(B_i, scw_{\min}) < SMD(B_i, \overline{scw_{\min}})$, then B_i is encoded by $\overline{scw_{\min}}$.

Fig. 3 is a diagram summarizing these eight cases. Three digits are used to label each case. The first bit corresponds to B_i's indicator; the second relates to B_{i+1}'s indicator and the third bit is used to indicate the values of its SMD relationships, which means that if $SMD(B_i, cw_{\min}) < SMD(B_i, \overline{cw_{\min}})$, then the third bit is set as '1'; otherwise, the third bit is set as '0'.

Example: Indicator Elimination Fig. 4 shows an example of the proposed indicator elimination. The SCBs in this example are generated by SMVQ. Here, the state codebook SCB size is 8 and the threshold TH for $SMVQ$ is 40. Fig. 4(a) is an input image divided into non-overlapping blocks, as shown in Fig. 4(b). For convenience, the seed blocks of image are shaded with gray in Fig. 4(b). Figs. 4(c) to 4(j) illustrate the corresponding SCBs, SMD values, and output compression codes for blocks, respectively. For instance, Fig. 4(c) shows the SCB of block B_7 and its $scw_{min} = scw_1$ and $\overline{scw_{\min}} = scw_4$ by Eq. 3. Block B7 is in the second column (i.e., $B_7 \in$ **Type I**) and its $d(B_7, scw_1) \leq TH$, therefore, an extra bit '1' is required to serve as B_7's indicator. Block B_8 is compressed by VQ (i.e., block B_8's indicator is '0') and $SMD(B_7, scw_1) = 29.93 < SMD(B_7, scw_4) = 56.53$, which satisfies Case 101; therefore, B_7 is encoded by "001" ($= scw_{min}$) and to imply block B_8's indicator as '0'. Fig. 4(f) shows an instance of **Type II**. Because its $d(B_{10}, scw_2) \leq TH$ (where $scw_{min} = scw_2$); therefore B_{10} is encoded by "010". In Fig. 4(i), $IND(B_{14}) = IND(B_{15}) =$ '0', and B_{14}'s SMD relationship $SMD(B_{14}, mcw_{30}) = 207.86 \geq SMD(B_{14}, mcw_{15}) = 96.64$ satisfies Case 000; therefore, B_{14} is encoded by "00000 11110" $= (mcw_0||mcw_{30})$ and to imply $IND(B_{15}) =$ '0'.

3.3. **Indicator Extraction and Image Reconstruction.** Indicator extraction is the reverse of the proposed indicator elimination process. Initially, the indices mapped to the blocks located in the first column and first row of the image are reconstructed by traditional VQ decoding with the MCB. Then, each residual block B_i is not only used to reconstruct the block but also used to extract the B_{i+1}'s indicator.

Generally, residual block B_i can be classified into three types. **Type I:** if B_i stays in second column, then its indicator can be directly extracted from compressed data. According to SMVQ, if $IND(B_i) =$ '0', then B_i was encoded by the MCB; otherwise, B_i was encoded by the SCB. If $IND(B_i) =$ '0', then next $[\log_2 M]$ bits from compressed data are set as temporary mcw_{min}; otherwise, the next $[\log_2 S]$ bits from the compressed data are set as the temporary scw_{min}. Then, B_i's restoration and B_{i+1}'s indicator extraction can be done by following rules.

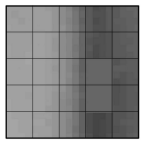
Rule 1: If $IND(B_i) =$ '0' and $mcw_{min} = mcw_0$, then $IND(B_{i+1}) =$ '0' and take next $[\log_2 M]$ bits to form mcw_{min}.

Rule 2: If $IND(B_i) =$ '1' and $scw_{min} = scw_0$, then $IND(B_{i+1}) =$ '0' and take next $[\log_2 S]$ bits to form scw_{min}.

Rule 3: If $IND(B_i) =$ '0' and $mcw_{min} = mcw_{M-1}$, then $IND(B_{i+1}) =$ '1' and take next $[\log_2 M]$ bits to form mcw_{min}.

Rule 4: If $IND(B_i) =$ '1' and $scw_{min} = scw_{S-1}$, then $IND(B_{i+1}) =$ '1' and take next $[\log_2 S]$ bits to form scw_{min}.

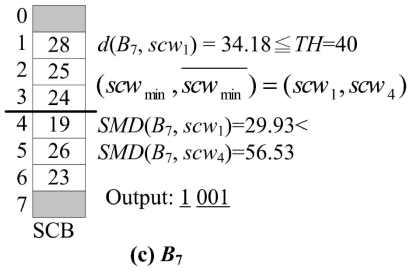
Rule 5: If $IND(B_i) =$ '0' and $mcw_0 < mcw_{\min} < mcw_{M-1}$, then : if $SMD(B_i, mcw_{\min}) < SMD(B_i, \overline{mcw_{\min}})$, then $IND(B_{i+1}) =$ '0', Else $IND(B_{i+1}) =$ '1' and swap the values of mcw_{min} and $\overline{mcw_{\min}}$.



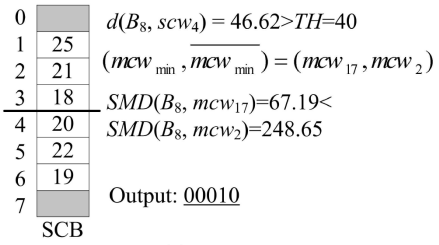
(a) Image *I*

28	25	17	6	12
28	B_7	B_8	B_9	B_{10}
28	B_{12}	B_{13}	B_{14}	B_{15}
28	B_{17}	B_{18}	B_{19}	B_{20}
28	B_{22}	B_{23}	B_{24}	B_{25}

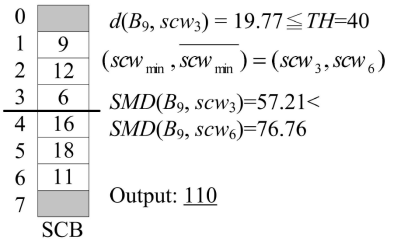
(b) Index table



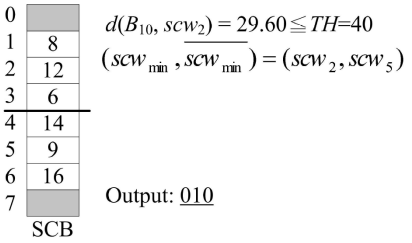
(c) B_7



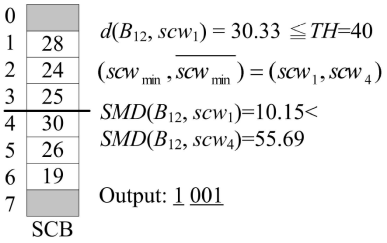
(d) B_8



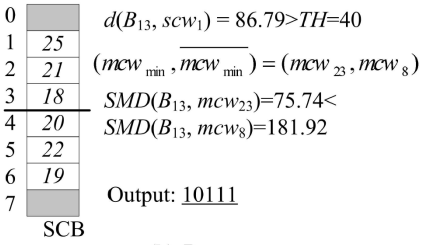
(e) B_9



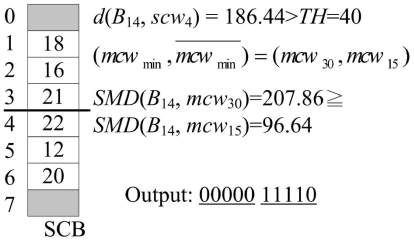
(f) B_{10}



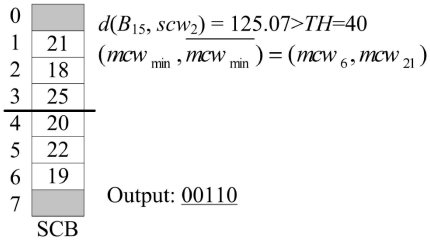
(g) B_{12}



(h) B_{13}



(i) B_{14}



(j) B_{15}

FIGURE 4. Example of the indicator elimination process

Rule 6: If $IND(B_i)$ = '1' and $scw_0 < scw_{\min} < scw_{S-1}$, then : if $SMD(B_i, scw_{\min}) < SMD(B_i, \overline{scw_{\min}})$, then $IND(B_{i+1})$ = '0', Else $IND(B_{i+1})$ = '1' and swap the values of scw_{min} and $\overline{scw_{\min}}$.

After B_{i+1}'s indicator has been extracted, B_i can be restored by using codeword mcw_{min} or scw_{min} according to B_i's indicator.

Type II, if B_i stays at the last block for each row, then B_i is reconstructed by using the codeword from MCB or SCB according to B_i's indicator.

Type III, if B_i does not belong to both **Types I** and **II**, then B_i's indicator is extracted from $B + i - 1$ and then apply the indicator extraction rules mentioned above. After the indicator has been extracted, the current reconstructing block can be correctly reconstructed.

Example: Indicator Extraction and Image Reconstruction. Fig. 5 shows an example for the indicator extraction and image reconstruction. Fig. 5(a) is part of the compressed code and Fig. 5(b) shows that the blocks located in the first column and first row have been decoded by using VQ with the MCB. As Fig. 5(c) shows, the corresponding bit of B_7 is '1'; therefore, B_7 is judged as it was compressed by SMVQ. Then, the next $[\log_2 S]$ bits are extracted from the compressed code to be scw_{min}. According to the proposed indicator extracting procedure, B_8's indicator is '0' according to $SMD(B_7, scw_1) < SMD(B_7, scw_4)$. Because B_7 implies that B_8's indicator is '0', the next $[\log_2 M]$ bits are extracted to form mcw_{min} and used to compute the SMD values. Because $SMD(B_8, mcw_2) \geq SMD(B_8, mcw_{17})$, B_9's indicator is '1'. Fig. 5(i) is an example of an exception. Based on the decompression in Fig. 5(h), B_{13}'s indicator is '0' and getting the next $[\log_2 M]$ bits to be $mcw_{min}(= mcw_0)$. Because its mcw_{min} equals mcw_0, B_{14}'s indicator can be judged as '0' and it must get the next $[\log_2 M]$ bits to form its mcw_{min}, which will also be used to reconstruct B_{13}.

4. **Experimental Results and Discussions.** To evaluate the performance of the proposed indicator elimination method, the compression methods of VQ, SMVQ and the proposed method are implemented by using MATLAB 7.0 software, which works on a Pentium-IV 1.5GHz CPU and 512MB RAM hardware platform. The three experimental codebooks, which respectively contain 128, 256, and 512 codewords, are trained by the LBG algorithm [10]. Each codeword in a codebook is 4×4 dimensions. Fig. 6 shows the test images sized 256×256 pixels.

Compression rate and image quality are two important factors for evaluating the performance of compression method. The compression rate (CR) is defined as follows:

$$CR = \frac{\|I'\|}{\|I\|}, \tag{5}$$

where $\|I\|$ and $\|I'\|$ represent the total bits of the original image I and compressed image I'. A smaller CR value indicates that a compression method has better compression performance. For visual quality factor, $PSNR$ (peak signal-to-noise ratio) is used to measure the visual quality of decompressed image generated by VQ, $SMVQ$, and the proposed method. $PSNR$ is defined as:

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE} (\text{dB}), \tag{6}$$

where MSE (mean-square error) is used to measure the difference between images I and I' with $H \times W$ pixels and defined as $MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (I_{ij} - I'_{ij})^2$, where I_{ij} and I'_{ij} represent the pixel values at locations (i, j) of I and I', respectively. A larger $PSNR$ value means that a decompressed image has visual quality better than a decompressed image with small $PSNR$ value.

4.1. **Experimental Results.** Fig. 7 shows the results of compression rates comparison. From experi- ment, VQ demonstrates a fixed compression rate ($CR = 0.0625$) in all test images, while SMVQ improves the VQ compression rate except in the case of complex images such as Baboon and Jet(F14). This is because a complex block usually has a low

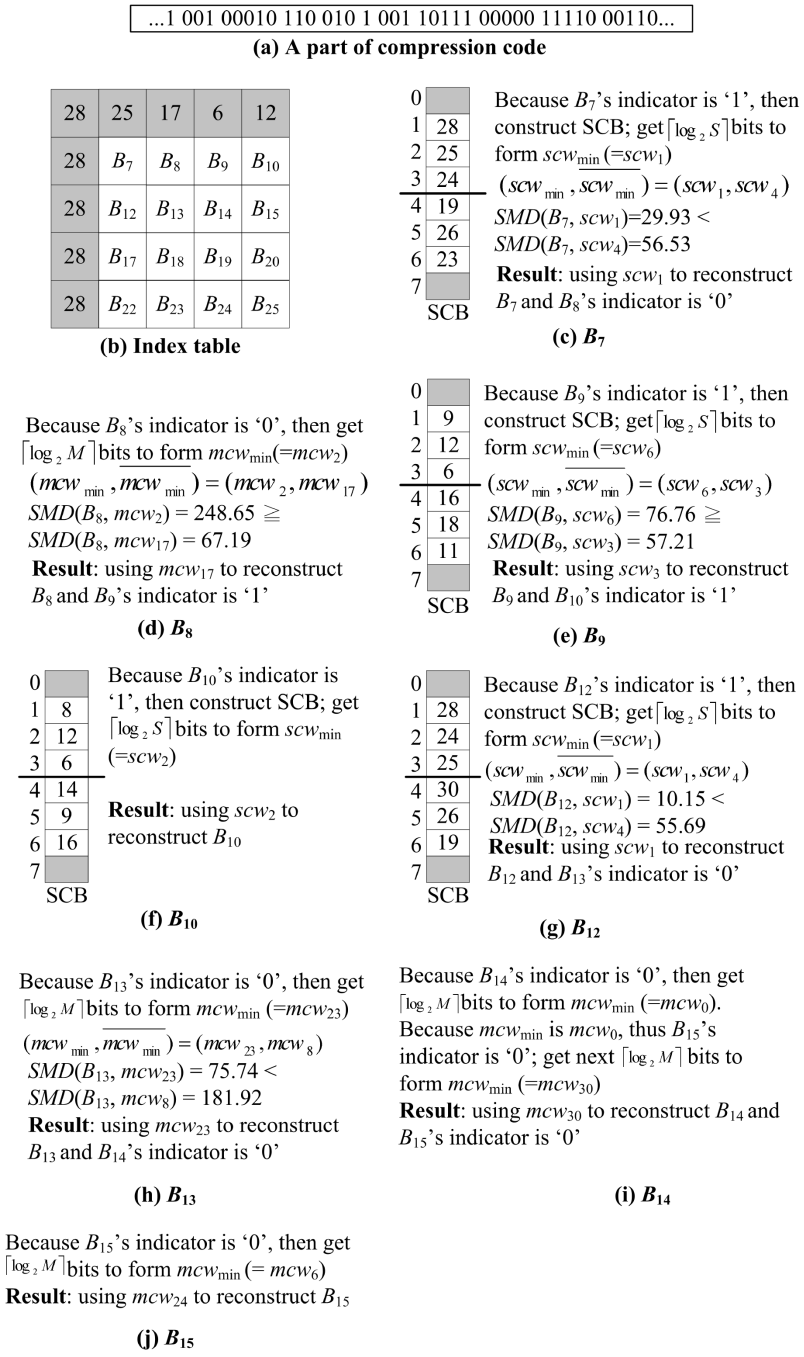


FIGURE 5. Example of indicator extraction and image reconstruction

probability of being replaced with a suitable codeword from a state codebook. In Fig. 7, the proposed method is superior in terms of compression rate. On average, the proposed method improves 17.68% against to the VQ in terms of compression rate. Also on average, our proposed method improves 8.31% against to the SMVQ in terms of compression rate.

Fig. 8 shows the results of the image quality comparison for VQ, SMVQ, and the proposed method. Since, mcw_0, mcw_{M-1}, scw_0, and scw_{S-1}, are preserved for handling the exception, so the $PSNR$ of the proposed method slightly lower than SMVQ. However, as Fig. 8 shows, reserving these pseudo-codewords had no significant effect on the image quality of most of the test images.

4.2. **Discussions.**

4.2.1. *The Size of Codebooks.* The size of the main codebook affects the performance in terms of image quality and compression rate. A large main codebook contains more codewords, which means a greater chance to find the most suitable codeword for compressing

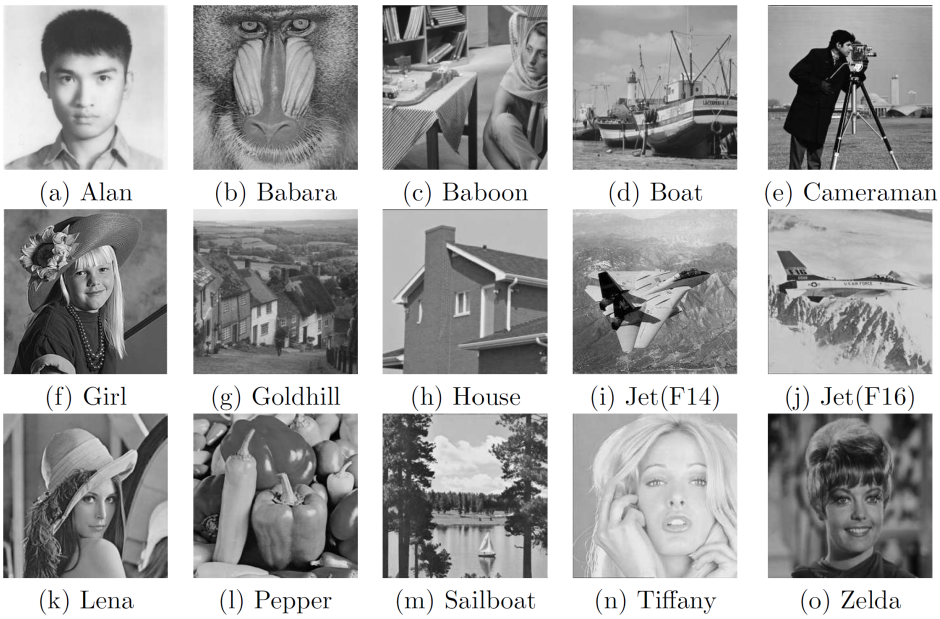


FIGURE 6. Fifteen images for the experiment

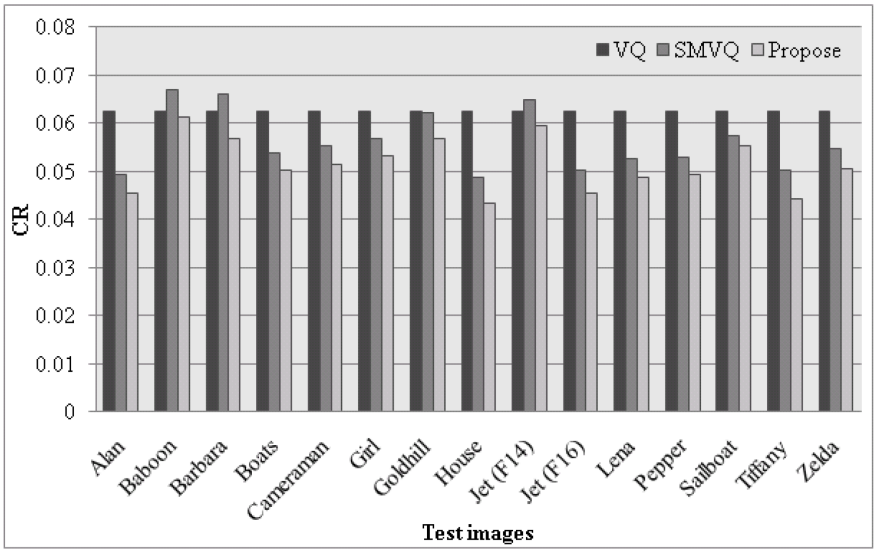


FIGURE 7. Comparison of compression rates ($M = 256$, $S = 8$, and $TH = 20$)

an image. Unfortunately, a larger main codebook affects the performance of compression rate. Figs. 9 and 10 show these relationships in the proposed method. As the two figures show, the largest main codebook provides the best image quality, but has the worst compression rate. We suggest that a main codebook containing 256 codewords is a suitable choice for producing an acceptable tradeoff between image quality and compression rate.

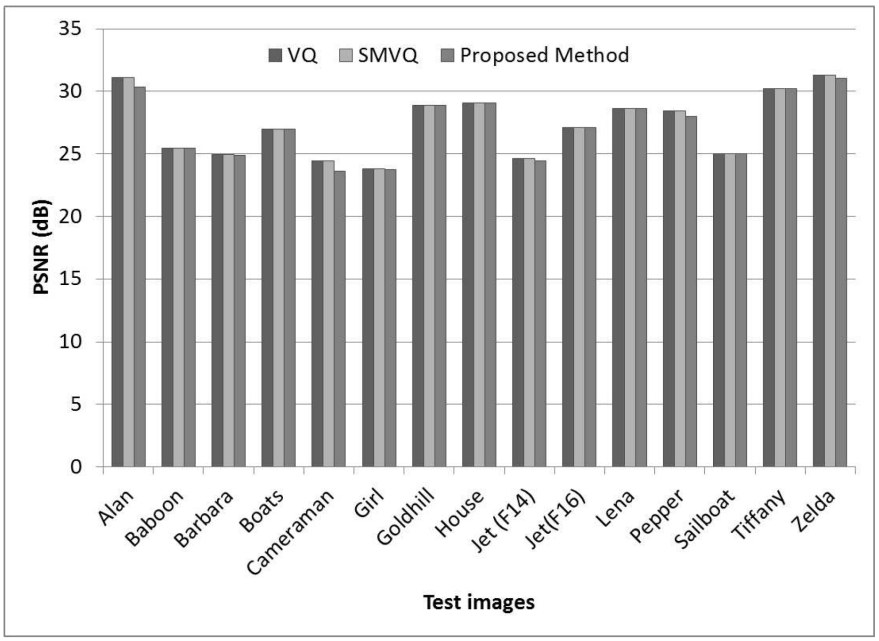


FIGURE 8. Comparison of image quality ($M = 256$, $S = 8$, and $TH = 20$)

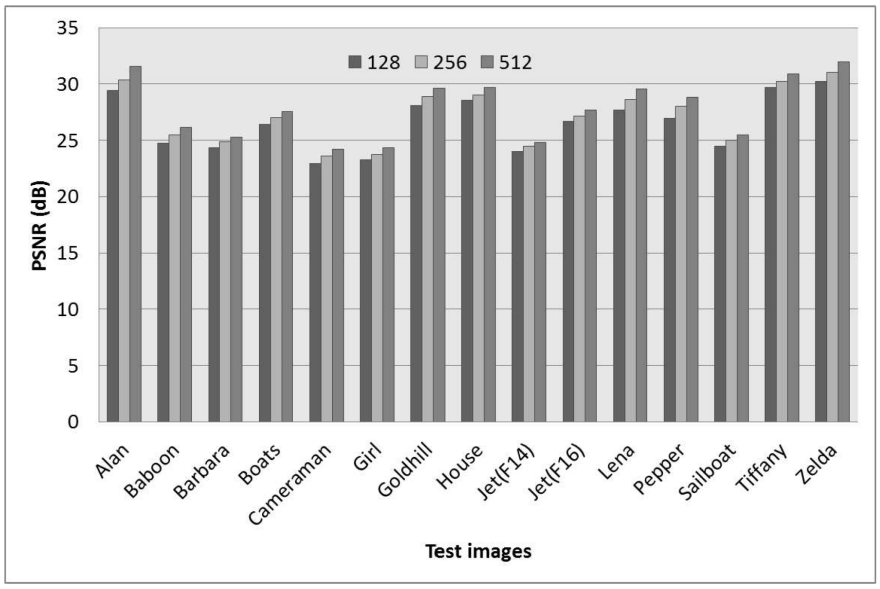


FIGURE 9. The image quality results for difference size of MCB ($S = 8$ and $TH = 20$)

The size of the state codebook also affects the performance of image compression. Therefore, higher compression rate performance can be achieved by increasing number of blocks compressed by SMVQ. Certainly, enlarging the size of a state codebook is an easy way to make sure that it contains more possible codewords. Unfortunately, a large state codebook requires more bits to represent the codeword index. Figs. 11 and 12 show the results of compression rate and image quality using different sizes of state codebooks. In this experiment, the size of the main codebook is set to 256. As the figures show, the size of a state codebook has insignificant effect on image quality, but it adversely affects

compression rate. Therefore, based on the experimental results shown in Figs. 11 and 12, we suggest that a suitable codebook size of state codebook is set to 4 or 8.

4.2.2. *Comparisons of Compression Rate.* To explore the CR performance of the proposed scheme, we conducted an experiment to observe CRs with near $PSNR$s. Typically, this

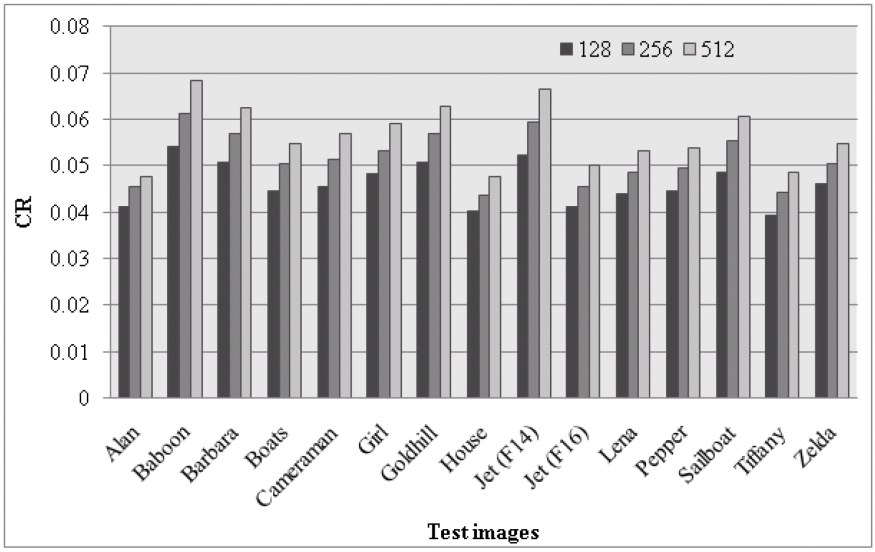


FIGURE 10. The results of compression rate for different size of MCB ($S = 8$ and $TH = 20$)

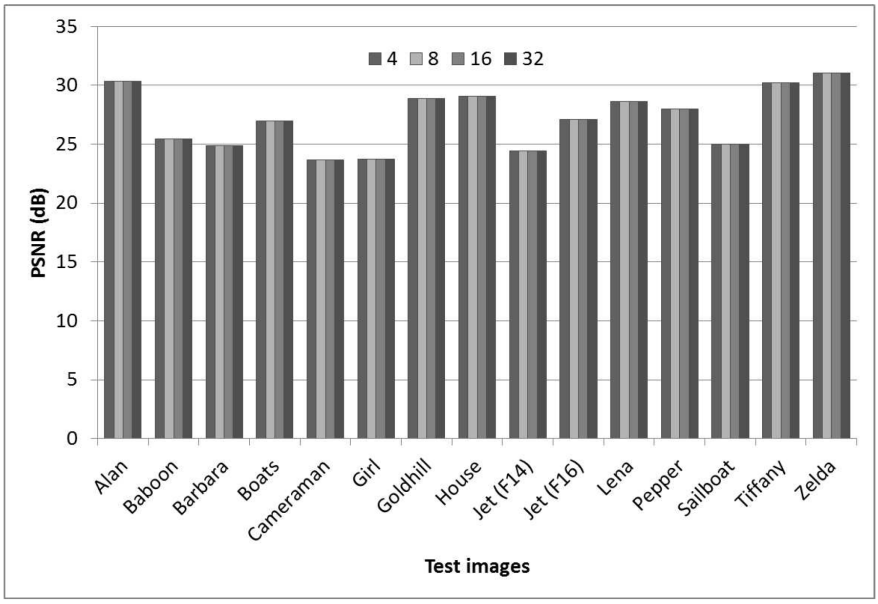


FIGURE 11. The results of image quality with different sizes of SCB ($M = 256$ and $TH = 20$)

kind of experiment involves observing CRs with a fixed $PSNR$. The major reason we used "near $PSNR$s" instead of "a fixed $PNSR$" is that two codewords in the main codebook have been reserved; therefore, we cannot offer exactly the same $PSNR$s as VQ and $SMVQ$ do. Compare the proposed method with VQ and $SMVQ$, Table 1 shows the

comparison results in terms of image quality and compression rate. In Table 1, for image quality, the proposed method is slightly lower than VQ and $SMVQ$s outcome. However, the compression rates offered by our proposed method are better than those achieved by VQ and $SMVQ$.

4.2.3. *Computation Cost.* Encoding a block using VQ compression requires the add operation to be performed 31 times (here, subtraction can be seen as the add operation), the multiplication operation 16 times and the square operation once to compute the distance between a block and a codeword in order to determine the most similar codeword in the main codebook. SMVQ requires even more computation cost to construct a state codebook during the encoding/decoding period because the purpose of SMVQ is to increase the compression rate.

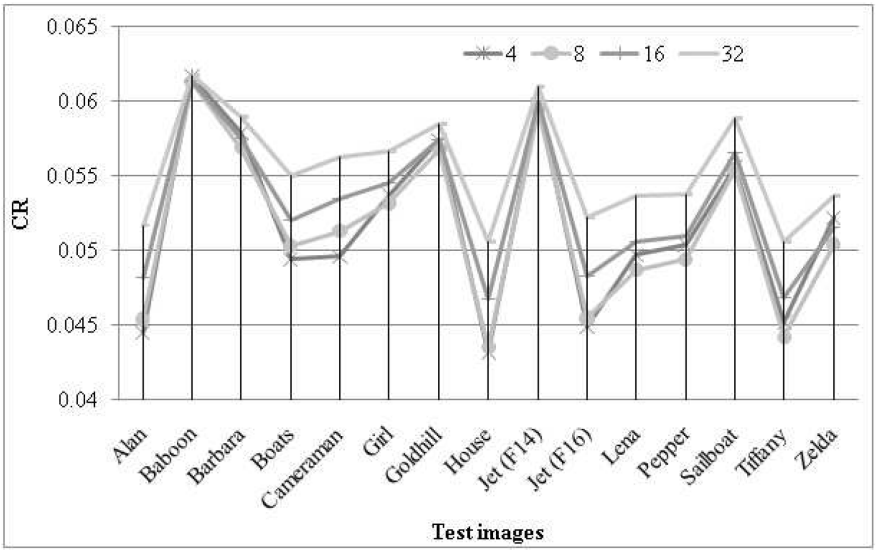


FIGURE 12. The results of compression rate with different sizes of SCB ($M = 256$ and $TH = 20$)

TABLE 1. Comparison results in near $PSNR$s ($M = 256$, $S = 8$, and $TH = 20$)

	VQ		SMVQ		Proposed	
Images	$PSNR$	CR (%)	$PSNR$	CR (%)	$PSNR$	CR (%)
Alan	31.14	6.25	31.14	4.93	30.35	4.54
Baboon	25.46	6.25	25.46	6.70	25.46	6.13
Barbara	24.93	6.25	24.92	6.60	24.90	5.69
Boats	27.00	6.25	27.00	5.39	26.99	5.03
Cameraman	24.43	6.25	24.43	5.52	23.64	5.13
Girl	23.8	6.25	23.8	5.69	23.74	5.32
Goldhill	28.91	6.25	28.91	6.23	28.90	5.68
House	29.05	6.25	29.05	4.87	29.05	4.35
Jet (F14)	24.61	6.25	24.61	6.50	24.45	5.95
Jet(F16)	27.13	6.25	27.13	5.01	27.13	4.55
Lena	28.65	6.25	28.65	5.26	28.65	4.87
Pepper	28.45	6.25	28.45	5.28	28.00	4.94
Sailboat	25.01	6.25	25.01	5.74	25.00	5.54
Tiffany	30.24	6.25	30.24	5.01	30.23	4.42
Zelda	31.31	6.25	31.31	5.47	31.04	5.04

Although the proposed method requires that declustering be performed on the main codebook and on each state codebook, declustering for the main codebook can be performed offline in advance, while a state codebook requires only 254 operations to compute all codeword distortions from the current blocks of the main codebook. In other words, our method requires S - 2 operations to form a state codebook than SMVQ requires.

Because the decoding process is the reverse of the encoding process, Table 2 shows the results of computation costs incurred during the compression processes for the VQ, SMVQ and proposed methods. In Table 2, SMVQ and the proposed method have a slightly higher computation cost than VQ. However, the computation cost of the proposed method is less than SMVQ. Combining the experimental results in Tables 1 and 2 makes obvious that our proposed method not only successfully improves the compression rate but also requires less computation cost than SMVQ does.

4.2.4. *Threshold Setting.* Threshold setting is used to adjust the performance of the proposed method. A suitable threshold setting helps user to get the desired compression image outcome. Then, the problem becomes one of finding a tradeoff between image quality and compression rate (i.e., a larger TH setting gives a better compression rate but worse image quality in the decompressed image, and vice versa). Table 3 shows comparison results for different threshold settings when the sizes of the main codebook and the state codebook are 256 and 8, respectively. In Table 3, the $PSNR$s of the proposed method are higher than the others and are very close when $TH = 10$ or 20. A user may use a larger threshold to get better compression rate. Contrary, the user may set a small threshold to get the better image quality of decompressed image.

5. **Conclusions.** In SMVQ, indicators are very important for distinguishing between blocks that have been encoded by SMVQ instead of VQ. Experiments prove that SMVQ offers significantly reduced compression rates in comparison with VQ. Experimental results show that the proposed indicator elimination method indeed improves the compression rate with minor extra computation cost, while maintaining almost the same image quality as can be achieved with SMVQ. In addition, we have observed that for the proposed method, the best sizes for the main codebook and the state codebook are 256 and 8, respectively, under a threshold of 20 for codeword selection from the main codebook.

TABLE 2. Results for execution time (sec.) comparison incurred during the encoding processes ($M = 256$, $S = 8$ and $TH = 20$)

Images	VQ	SMVQ	Proposed
Alan	12.313	78.750	73.426
Baboon	12.016	77.375	73.535
Barbara	12.016	75.141	68.796
Boats	13.188	75.312	72.973
Cameraman	12.219	69.531	68.283
Girl	12.547	64.141	68.657
Goldhill	12.718	67.563	66.599
House	13.110	73.687	67.067
Jet(F14)	12.062	82.969	74.392
Jet(F16)	12.281	80.938	74.876
Lena	12.235	67.187	67.799
Pepper	12.421	65.718	69.981
Sailboat	12.172	67.547	66.708
Tiffany	12.360	85.141	78.148
Zelda	12.609	63.156	60.972

TABLE 3. PSNR comparison results for different TH settings)

Images	VQ	SMVQ				Proposed method			
		TH=10	TH=20	TH=30	TH=40	TH=10	TH=20	TH=30	TH=40
Alan	31.14	31.14	31.14	31.12	31.07	30.35	30.35	30.32	30.26
Baboon	25.46	25.46	25.46	25.45	25.43	25.46	25.46	25.45	25.42
Barbara	24.93	24.93	24.92	24.92	24.90	24.90	24.90	24.89	24.87
Boats	27.00	27.00	27.00	26.99	26.97	26.99	26.99	26.98	26.95
Cameraman	24.43	24.43	24.43	24.43	24.41	23.64	23.64	23.64	23.62
Girl	23.80	23.80	23.80	23.79	23.78	23.75	23.74	23.74	23.72
Goldhill	28.91	28.91	28.91	28.88	28.80	28.90	28.90	28.87	28.76
House	29.05	29.05	29.05	29.04	29.00	29.05	29.05	29.03	28.98
Jet (F14)	24.61	24.61	24.61	24.60	24.60	24.45	24.45	24.45	24.44
Jet(F16)	27.13	27.13	27.13	27.13	27.10	27.13	27.13	27.13	27.10
Lena	28.65	28.65	28.65	28.66	28.59	28.65	28.65	28.63	28.57
Pepper	28.45	28.45	28.45	28.42	28.37	28.00	28.00	27.96	27.90
Sailboat	25.01	25.01	25.01	25.00	24.99	25.01	25.00	25.00	24.98
Tiffany	30.24	30.24	30.24	30.23	30.18	30.23	30.23	30.21	30.15
Zelda	31.31	31.31	31.31	31.23	30.99	31.05	31.04	30.94	30.69

REFERENCES

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

[2] R. M. Gray, Vector quantization, *IEEE ASSP Magzine*, vol. 1, no. 2, pp. 4-29, 1984.

[3] C. H. Hsieh, and J. C. Tsai, Lossless compression of vq index with search-order coding, *IEEE Trans. Image Processing*, vol. 5, no. 11, pp. 1579-1582, 1996.

[4] C. Christopoulos, A. Skodras, and T. Ebrahimi, The JPEG2000 still image coding system: an overview, *IEEE Trans. Consumer Electronics*, vol. 46, no. 4, pp. 1103-1127, 2000.

[5] Y. H. Han, and J. J. Leou, Detection and correction of transmission errors in jpeg images, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 2, pp. 221-231, 1998.

[6] G. K. Wallace, The jpeg still picture compression standard, *IEEE Trans. Consumer Electronics*, vol. 38, no. 1, pp. xviii-xxxiv, 1992.

[7] Z. N. Li, and M. S. Drew, Fundamentals of Multimedia, *Prentice Hall*, Upper Saddle River, New Jersey, USA, 2003.

[8] K. Sayood, Introduction to Data Compression, *Morgan Kaufmann*, San Francisco, CA, USA, 2000.

[9] C. C. Chang, C. H. Sung, and T. S. Chen, A locally adaptive scheme for image index compression, *Proc. of The 10th Conference on Computer Vision, Graphics, and Image Processing*, pp. 93-99, 1997.

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

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

[12] C. C. Chang, F. C. Shine, T. S. Chen, Pattern-based side match vector quantization for image compression, *Imaging Science Journal*, vol. 48, no. 2, pp. 63-76, 2000.

[13] J. Ziv, and A. Lempel, Compression of individual sequences via variable-rate coding, *Journal of Optical Engineering*, vol. 24, no. 5, pp. 530-536, 1978.

[14] C. C. Chang, T. D. Kieu, and W. C. Wu, A lossless data embedding technique by joint neighboring coding, *Journal of Pattern Recognition*, vol. 42, no. 7, pp. 1597-1603, 2009.

[15] C. C. Chang, P. Y. Pai, C. M. Yeh, Y. K. Chan, A high payload frequency-based reversible image hiding method, *Journal of Information Sciences*, vol. 180, no. 11, pp. 2286-2298, 2010.

[16] T. D. Kieu, and C. C. Chang, A high stego-image quality steganographic scheme with reversibility and high payload using multiple embedding strategy, *Journal of Systems and Software*, vol. 82, no. 10, pp. 1743-1752, 2009.

[17] L. Huang, L. Tseng, and M. Hwang, The study of data hiding in medical images, *International Journal of Network Security*, vol. 14, no. 6, pp. 301-309, 2012.

[18] C. Y. Yang, C. H. Lin, and W. C. Hu, Reversible data hiding for high-quality images based on integer wavelet transform, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 3, no. 2, pp. 142-150, 2012.

[19] A. Latif, and F. Rashidi, A watermarking scheme based on the parametric slant-hadamard transform, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 2, no. 4, pp. 377-386, 2011.

[20] C. C. Chang, and C. Y. Lin, Reversible steganography for vq-compressed images using side matching and relocation, *IEEE Trans. Information Forensics and Security*, vol. 1, no. 4, pp. 493-501, 2006.

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

[22] L. Guan, and M. Kamel, Equal-average hyperplane partitioning method for vector quantization of image data, *Journal of Pattern Recognition Letters*, vol. 13, no. 10, pp. 693-699, 1992.

[23] Z. M. Lu, and S. H. Sun, Equal-average equal-variance equal-norm nearest neighbor search algorithm for vector quantization, *Journal of IEICE transactions on information and systems*, vol. E86-D, no. 3, pp. 660-663, 2003.

[24] Z. Pan, K. Kotani, and T. Ohmi, Fast search method for image vector quantization based on equal-average equal-variance and partial sum concept, *Proc. of IEEE International Conference on Multimedia and Expo*, pp. 1440-1443, 2005.