# Reversible Data Hiding for VQ-Compressed Images Based on an Index Replacement Technique

Peyman Rahmani and Gholamhossein Dastghaibyfard

***Abstract*—In this paper a reversible data hiding scheme based on vector quantization (VQ) image compression technique is proposed. Reversibility property allows VQ compressed code to be completely recovered after extraction of the secret data. Positions in the sorted codebook are divided into some intervals. A mapping method is proposed to embed data. To each position with high hit rate in the sorted codebook, an interval of the positions is assigned. According to the secret bits each index which located in the positions with high hit rates of the sorted codebook is re-encoded by one of the indices in the interval which has been assigned to that position. Experimental results demonstrate that proposed scheme outperforms other existing schemes.**

***Index Terms*—Date hiding, reversible data hiding, vector quantization, Side match vector quantization**

## I. Introduction

One of the great challenges in today's fast developing internet is how to transfer secret data. Data hiding is a protective technique for secret data transmission. Secret data can be embedded into digital images, audios, videos… in such a way that it is undetectable to unauthorized interceptors. This paper considers data hiding in digital images, where images before and after data hiding are called cover images and stego-images, respectively.

Data hiding techniques are either reversible or irreversible. In reversible or so called lossless data hiding techniques original image can be recovered after extraction of the embedded data. Irreversible data hiding techniques distort cover image in an unrecoverable manner. However, any distortion may not be acceptable in some applications.

Most reversible data hiding schemes perform in the spatial domain of images, and two well known methods are: difference expansion [1] and histogram shifting [2].

Due to limitation of storage and bandwidth, typically images are stored and transmitted in compressed formats. Vector Quantization (VQ) [3] is a widely used image compression technique with properties of simple structure and fast decoding process.

VQ-based image data hiding schemes are proposed both irreversible and reversible techniques [4]-[8] in the literature. In contract, a reversible VQ-based image data hiding technique must be able to recover original VQ index table after extracting secret data.

Manuscript received April 10, 2012; revised May 18, 2012.

The authors are with the Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran (e-mail: rahmani@cse.shirazu.ac.ir, dstghaib@shirazu.ac.ir).

Some reversible VQ-based data hiding schemes [5]-[7] exploit the concept of side matching to embed data. VQ does not consider correlation between neighbor blocks; therefore VQ index table includes some redundant data. Side-Match VQ (SMVQ) [9] improves compression performance of VQ by removing such redundancies. Data hiding can be performed by replacing redundant data with secret bits.

The rest of this paper is organized as follows. In Section II, first VQ and SMVQ are explained and related works are given. Our proposed scheme is presented in Section III. Section IV includes experimental results and discussions.

## II. Background

In this section first VQ and SMVQ are explained and then related works will be given.

### A. Vector Quantization

VQ is a lossy data compression technique, widely used for image compression. VQ has three phases: training codebook, encoding and decoding. Most popular algorithm for training a codebook is LBG [10]. In the encoding procedure, to be encoded image is partitioned into non-overlapping blocks of $h \times h$ pixels. Then for each block $X = (x_{0,0}, x_{0.1}, \ldots, x_{0,h-1}, x_{1,0}, x_{1.1}, \ldots, x_{1,h-1}, \ldots, x_{h-1,0}, x_{h-1.1}, \ldots, x_{h-1,h-1})$ the closest codeword is found from codebook. Similarity between input block $X$ and codeword $Y = (y_{0,0}, y_{0.1}, \ldots, y_{0,h-1}, y_{1,0}, y_{1.1}, \ldots, y_{1,h-1}, \ldots, y_{h-1,0}, y_{h-1.1}, \ldots, y_{h-1,h-1})$ is measured by the following squared Euclidean distance and then current block $X$ is encoded by the index value of the closest codeword in the codebook.

$$D(X,Y) = \sum_{i=0}^{h-1} \sum_{j=0}^{h-1} (x_{i,j} - y_{i,j})^2$$

In decoding procedure, each block of the image can be easily reconstructed by its corresponding codeword in the codebook.

### B. Side-Match Vector Quantization

SMVQ is an image compression technique that exploits correlation between neighboring blocks, to further improve compression ratio of VQ method.

In SMVQ encoding procedure, blocks of the first row and first column of the image are encoded by VQ and for each residual block a state codebook is generated by using side match prediction.

For each block $X$ to be encoded, border pixel values of its upper and left adjacent block as shown in Fig. 1 are used to compute side match distortion (SMD) for all codewords in the super codebook and then $m$ codewords with minimized SMD are selected to form a state codebook for block $X$. Then codeword with minimum Euclidean distance is found in the

state codebook. Position of the closest codeword in the state codebook is used to encode block $X$ by $\lceil \log_2 m \rceil$ bits. SMD among upper block $U$, left block $L$ and codeword $Y$ is computed as follows:

$$SMD(Y) = \sum_{i=0}^{h-1} (u_{h-1,i} - y_{0,i})^2 + \sum_{i=0}^{h-1} (l_{i,h-1} - y_{i,0})^2$$

In decoding procedure, for each block the state codebook same as which was generated in encoding can be generated from super codebook, therefore the state codebooks need not to be stored.

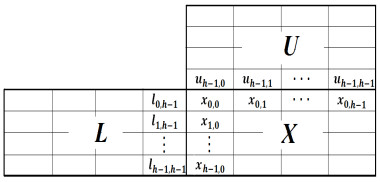


Fig. 1. Border pixels which is used in SMVQ

In SMVQ selection of $m$ has a great impact in compression ratio. If $m$ be equal to $n$, the resulted index table has the same size as VQ index table, but the histogram of the SMVQ indices has a concentrated distribution.

### C. Some VQ-based Reversible Data Hiding Schemes

In 2007 Chang *et al*. [4] proposed a reversible scheme that sorts codebook based on frequency of occurrences of indices in images, then the sorted codebook is divide into $2^{B-1}*3$ clusters, where $B$ is the number of bits to be embedded in each index. Only indices in the front one-third clusters are used to embed data. Such indices are transferred to the corresponding indices in the other clusters to embed secret bits. Other clusters are used for recovery purpose.

In 2007 Chang *et al*. [5] proposed another scheme which uses minimum spanning tree or shortest spanning path to put dissimilar codewords into same cluster. In embedding process if current index has minimum side match distortion among all codewords in its cluster, then this block is called exchangeable and according to the secret data, one of the indices in that cluster is selected to re-encode that block. Two indices are used as indicators to indicate if a block is non-exchangeable.

In 2010 Wang *et al*. [8] proposed a reversible data hiding scheme for VQ-compressed images. In a pre-processing step, codebook is sorted according to the similarity of the codewords. After sorting adjacent indices have closer values and each index is represented by its difference with its previous index which decrease number of needed bits to encode each index. Remained space is used to embed secret bits.

In fact, all VQ-based data hiding schemes explore redundancy in the VQ index table and replace redundant data with secret bits. By exploiting the concept of side matching, more capacity can be provided.

## III. Proposed Scheme

In this section the proposed scheme is described in details. It is assumed that input and output are in the form of VQ index table. In what follows, it is assumed that SMVQ state codebook has the same size as super codebook. Moreover position means position of the index in the sorted codebook.

SMVQ histogram has been considered in [5-7]. They found that, most indices are located in the first positions of the sorted codebook.

Assume $b = \lceil \log_2 n \rceil$ is the number of bits representing each index in the index table, where $n$ is the codebook size. To embed $c$ bits into each index, $n$ positions of the sorted codebook are divided into $2^d$ intervals with the same sizes, where $d = b - c$. Each of these intervals (except last one) is associated with one of the first $r = 2^d - 1$ positions of the sorted codebook i.e., positions in the range of $[0, r)$. Last interval is used as indicator of the indices located in the positions with lower hit rates i.e., positions in the range of $[r, n)$.

Indices located in the first $r$ positions of the sorted codebook are called mappable. To embed data in such indices, the index located in the position $p$ ($0 \le p < r$), is re-encoded by one of the indices located in the interval $p$ according to $c$ secret bits.

In the next subsection data embedding phase is described. Data extraction and recovery phase is described in the second subsection.

### A. The Data Embedding Procedure

In the proposed scheme, the indices in the first row and first column of the index table, are called seed indices and will be kept unchanged and don't carry any secret data. The data embedding procedure embed data in residual indices as follows:

Input: A VQ index table, the secret bitstream, a codebook ($n$ = codebook size, $b = \lceil \log_2 n \rceil$), and parameter $c$ ($d = b - c$, $r = 2^d - 1$)
Output: The embedded index table
For each index $I_i$ do:

Step 1: Compute SMD for all codewords in the codebook and then sort the codebook based on SMD.

Step 2: Find $p_i$ which is the position of $I_i$ in the sorted codebook.

Step 3: Embedded index $I_i'$ is obtained based on $p_i$ according to the following two cases:

Case 1 ($p_i < r$): Retrieve next $c$ secret bits and compute its decimal value $s_k$. Then compute $p_i'$ which is the position of the embedded index as follows: $p_i' = p_i \times 2^c + s_k$. Then index $I_i'$ which is the index located in the position $p_i'$ of the sorted codebook is sent to the output as embedded index.

Case 2 ($p_i \ge r$): Retrieve next $c$ secret bits and compute its decimal value $s_k$. Then compute $p_i'$ which is the position of an index as indicator as follows: $p_i' = r \times 2^c + s_k$ and then send $I_i' \| I_i$ to the output, where $I_i'$ is the index located in the position $p_i'$ of the sorted codebook and $I_i$ is current index and $\|$ is the concatenation operation.

### B. The Data Extraction and Original VQ Index Table Recovery Procedure

Seed indices don't include secret data. The data extraction and recovery procedure performs on the residual indices as follows:

Input: An embedded index table, a codebook ($n$ =

codebook size, $b = \lceil \log_2 n \rceil$), and parameter $c$ ($d = b - c$, $r = 2^d - 1$)

Output: The original VQ index table and the secret bitstream

For each embedded index $I_i'$ do:

Step 1: Compute SMD for all codewords in the codebook and then sort the codebook based on SMD.

Step 2: Find $p_i'$ which is the position of the current embedded index $I_i'$ in the sorted codebook.

Step 3: Compute $p_i$ as follows: $p_i = \lfloor p_i' / 2^c \rfloor$, extract secret data $s_k$ by $s_k = p_i' - p_i \times 2^c$ and convert that to its binary format in $c$ bits and concatenate with previously extracted secret bits.

Step 5: Recover original index $I_i$ based on $p_i$ according to the following two cases:

Case 1 ($p_i < r$): Find original index $I_i$ in the position $p_i$ of the sorted codebook.

Case 2 ($p_i = r$): Retrieve next index from embedded index table as original index $I_i$.

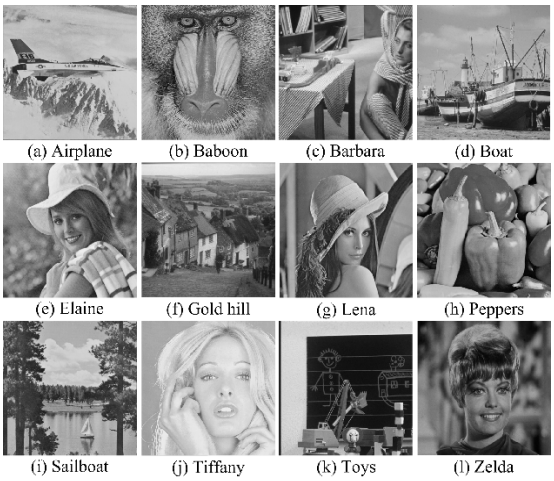


Fig. 2. 12 test images

## IV. Experimental Results

This section includes some experiments and discussions on experimental results to verify the effectiveness and feasibility of this newly proposed scheme. For experiments, the following 12 gray-level test images were used (Fig. 2). All images are of the size 512×512. For VQ encoding, each image was divided into 16384 blocks of size 4×4 pixels. Two codebooks include 256 and 512 16-dimensional codewords were trained by using the well-known Linde–Buzo–Gray (LBG) [10] algorithm.

Since the proposed scheme is reversible and can completely recover original VQ index table, the quality of the image is not considered. We consider two major aspects of performance i.e., hiding capacity and bit rate of the embedded index table. Bit rate is used to evaluate the compression performance, is the ratio of the size of the output codestream to the number of pixels in the image. Bit rate is defined as follows: bit_rate = $|CS| / (H \times W)$ where $|CS|$, $H$, $W$ are the bit number of the output codestream, height and width of the image, respectively. Bit-rate is represented by bits per pixel (bpp).

Tables I and II illustrate hiding capacity and bit rate of the proposed scheme for various values of $c$ for codebook size 256 and 512 respectively. It can be observed that by selecting larger capacities, size of the output codestream will increase. In fact there is a trade off between hiding capacity and bit rate.

Table III shows comparison of the proposed scheme and the schemes proposed by Chang *et al*. [4], Chang *et al*. [5] and Wang *et al*. [8]. In all cases the hiding capacity of our proposed scheme outperforms theirs, whereas lower bit rates by small values of $c$ are achievable.

TABLE I. Performance of the Proposed Scheme for Various Values of $C$ (Codebook Size 256)

| Images | $c = 3$ | | $c = 4$ | | $c = 5$ | |
|---|---|---|---|---|---|---|
| | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) |
| Airplane | 48,387 | 0.577 | 64,516 | 0.613 | 80,645 | 0.656 |
| Baboon | 48,387 | 0.662 | 64,516 | 0.756 | 80,645 | 0.832 |
| Barbara | 48,387 | 0.634 | 64,516 | 0.699 | 80,645 | 0.753 |
| Boat | 48,387 | 0.590 | 64,516 | 0.640 | 80,645 | 0.686 |
| Elaine | 48,387 | 0.541 | 64,516 | 0.583 | 80,645 | 0.645 |
| Gold hill | 48,387 | 0.563 | 64,516 | 0.630 | 80,645 | 0.711 |
| Lena | 48,387 | 0.564 | 64,516 | 0.610 | 80,645 | 0.662 |
| Pepper | 48,387 | 0.566 | 64,516 | 0.608 | 80,645 | 0.657 |
| Sailboat | 48,387 | 0.596 | 64,516 | 0.650 | 80,645 | 0.700 |
| Tiffany | 48,387 | 0.529 | 64,516 | 0.567 | 80,645 | 0.621 |
| Toys | 48,387 | 0.593 | 64,516 | 0.629 | 80,645 | 0.667 |
| Zelda | 48,387 | 0.536 | 64,516 | 0.583 | 80,645 | 0.647 |

TABLE II. PERFORMANCE OF THE PROPOSED SCHEME FOR VARIOUS VALUES OF C (CODEBOOK SIZE 512)

| Images | $c = 3$ | | $c = 4$ | | $c = 5$ | |
|---|---|---|---|---|---|---|
| | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) |
| Airplane | 48,387 | 0.645 | 64,516 | 0.685 | 80,645 | 0.731 |
| Baboon | 48,387 | 0.709 | 64,516 | 0.846 | 80,645 | 0.935 |
| Barbara | 48,387 | 0.708 | 64,516 | 0.781 | 80,645 | 0.847 |
| Boat | 48,387 | 0.660 | 64,516 | 0.715 | 80,645 | 0.770 |
| Elaine | 48,387 | 0.604 | 64,516 | 0.649 | 80,645 | 0.721 |
| Gold hill | 48,387 | 0.631 | 64,516 | 0.699 | 80,645 | 0.793 |
| Lena | 48,387 | 0.627 | 64,516 | 0.677 | 80,645 | 0.733 |
| Pepper | 48,387 | 0.632 | 64,516 | 0.677 | 80,645 | 0.732 |
| Sailboat | 48,387 | 0.667 | 64,516 | 0.724 | 80,645 | 0.784 |
| Tiffany | 48,387 | 0.591 | 64,516 | 0.633 | 80,645 | 0.704 |
| Toys | 48,387 | 0.662 | 64,516 | 0.708 | 80,645 | 0.755 |
| Zelda | 48,387 | 0.598 | 64,516 | 0.647 | 80,645 | 0.713 |

TABLE III. COMPARISON BETWEEN THE PROPOSED SCHEME AND OTHER SCHEMES (CODEBOOK SIZE 512)

| Images | Scheme [4] ($B = 2$) | | Scheme [5] ( *number of clusters* = 51) | | Scheme [8] | | The proposed scheme ($c = 4$) | |
|---|---|---|---|---|---|---|---|---|
| | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) | Capacity (bits) | Bit rate (bpp) |
| Airplane | 24,608 | 0.76 | 50,522 | 0.64 | 49,149 | 0.69 | 64,516 | 0.69 |
| Baboon | 13,138 | 0.73 | 37,166 | 0.83 | 49,149 | 0.86 | 64,516 | 0.85 |
| Boat | 20,626 | 0.75 | 48,186 | 0.67 | 49,149 | 0.70 | 64,516 | 0.72 |
| Gold hill | 16,548 | 0.75 | 47,281 | 0.68 | 49,149 | 0.73 | 64,516 | 0.70 |
| Lena | 22,626 | 0.75 | 48,987 | 0.64 | 49,149 | 0.74 | 64,516 | 0.68 |
| Peppers | 22,592 | 0.74 | 50,131 | 0.64 | 49,149 | 0.72 | 64,516 | 0.68 |

## REFERENCES


[1] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, 2003.

[1] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354-362, March 2006.

[2] R. M. Gray, "Vector quantization," *IEEE Acoustics, Speech and Signal Processing Magazine*, vol. 1, no. 2, pp. 4-29, April 1984.

[3] C. C. Chang, W. C. Wu, and Y. C. Hu, "Lossless recovery of a VQ index table with embedded secret data," *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 207–216, 2007.

[4] C. C. Chang, Y. P. Hsieh, and C. Y. Lin, "Lossless data embedding with high embedding capacity based on declustering for VQ-compressed codes," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 341-349, September 2007.

[5] C. C. Chang and C. Y. Lin, "Reversible steganography for VQ-compressed images using side matching and relocation," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 4, pp. 493-501, December 2006.

[6] P. Tsai, "Histogram-based reversible data hiding for vector quantisation-compressed images," *IET Image Processing*, vol. 3, no. 2, pp. 100–114, 2009.

[7] Z. H. Wang, C. C. Chang, K. N. Chen, and M. C. Li, "An encoding method for both image compression and data lossless information hiding," *The Journal of Systems and Software*, vol. 83, no. 11, pp. 2073-2082, 2010.

[8] T. Kim, "Side match and overlap match vector quantizers for images," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 170-185, April 1992.

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


**Peyman Rahmani** received his B.Sc. degree in computer engineering from Department of Computer Engineering, College of Engineering, Tarbiat Moalem University, Tehran, Iran, in 2008 and his M.Sc. degree in computer engineering from Department of Computer Science and Engineering, College of Electrical & Computer Engineering, Shiraz University, Shiraz, Iran, in 2011. His current research interests include data hiding and image compression.

**Gholamhossein Dastghaibyfard** received his M.Sc. and Ph.D. degree in computer science from Electrical Engineering and Computer Science Department, College of Engineering, University of Oklahoma, Norman Oklahoma USA, in 1979 and 1990, respectively. He is currently an assistant professor of computer science in Department of Computer Science and Engineering, College of Electrical & Computer Engineering, Shiraz University, Shiraz, Iran. His current research interests include parallel algorithms, grid computing and information technology.