1

An Image Data Hiding Scheme Based on Vector Quantization and Graph Coloring

Shuai Yue[1], Zhi-Hui Wang[2], and Chin-Chen Chang[3]

[1] Department of Software,
Dalian University of Technology,
DaLian, China,
peaceful1207@gmail.com
[2] Department of Software,
Dalian University of Technology,
DaLian, China,
wangzhihui1017@gmail.com
[3] Department of Information Engineering and Computer Science,
Feng Chia University,
Taichung, Taiwan
alan3c@gmail.com

Summary. Vector quantization, i.e., VQ, is an important image compression method. In the past, many data hiding schemes have been proposed based on VQ. However, none of them is graph coloring based, although graph coloring has been used in many applications. In this work, we propose a VQ-based data hiding scheme using graph coloring. The proposed scheme colored every codeword in the codebook in different colors, each of which represents some bits of secret messages. Then the scheme hidden data into the compression code through replacing one codeword in a color with another codeword in another color. The performance of the proposed scheme was evaluated on the basis of embedding capacity and imperceptibility, which proved the scheme's validity.[1]

1.1 Introduction

At the present time, computer science is making great advancements with the speed of computers and the bandwidth of the Internet seemingly increasing every day. This provides people more convenience in processing multi-media data, but, due to the great quantities of such data that are being transmitted, most of the image, sound, and video must be compressed before transmitting. Vector quantization is a popular image compression method that manipulates the fact that most blocks in an image are smooth and are similar to each other. VQ quantizes the image with a number of representative blocks, noted as codebook, so that the image can be represented

[1] Supported by the Fundamental Research Funds for the Central Universities.

J.-S. Pan et al. (Eds.): Recent Advances in Information Hiding and Applications, ISRL 40, pp. 1–17.
springerlink.com

as an index table of codewords in the codebook. Although VQ is simple, it can effectively reduce the size of the image while maintaining an acceptable quality of the compressed image.

Countless malicious attacks occur every day on the Internet, which is a public environment. Therefore, many techniques have been developed to protect data from illegal access. Data hiding in an image is one of those techniques that protect data by disguising the existence of the secret data by hiding data in the image with slight changes to the image on the condition that the modification cannot be perceived by the human eye [1, 2]. Data hiding in a VQ compressed image is more difficult than normal data hiding in the original image, since the size of the file that contains the VQ-compressed image is far smaller than the original image, and changes to the index table are usually more obvious.

In the past, many literatures related to VQ-based image data hiding have been published. Lin et al. [3] proposed a scheme in which the codebook, CB, is partitioned into two homogeneous parts, i.e., CB_1 and CB_2, where CB_1 and CB_2 are similar to each other. When '0' is to be embedded, the scheme finds the most similar codeword in CB_1, and when '1' is to be embedded, the scheme finds the most similar codeword in CB_2. Lin et al.'s scheme is similar to the least-significant-bit(LSB) substitution method. In 2005, Wu et al. [4] proposed a scheme that improved Lin et al.'s scheme by providing a codeword editing procedure to make CB_1 and CB_2 more similar to each other. However, both Lin et al's and Wu et al's methods can provide only limited embedding capacity. Later, Li et al. [5] proposed a scheme that divides the codebook into clusters according to the similarity between codewords. If the size of the cluster equals 2^k, then k bits of secret message can be hidden through the replacement of the codewords in the cluster. However, the embedding capacity is still limited because Li et al's scheme requires the size of the cluster to be equal to 2^k, which limited the number of clusters that can be used to embed data.

In the remainder of this work, we propose a scheme based on VQ and graph coloring. In Section 1.2, some works related to the proposed scheme are introduced. In Section 1.3, the proposed scheme is described. In Section 1.4, the experiment results are presented to show the validity of the proposed scheme. We conclude this chapter in Sec. 1.5.

1.2 Related Works

The proposed scheme makes use of vector quantization, graph coloring and particle swarm optimization, and these processes are described in this section.

1.2.1 Vector Quantization

Vector Quantization is a lossy data compression method that is used often [6]. It starts with the construction of a codebook that consists of codewords, each of which indicates a block of pixels. To compress an image, VQ scans the image in a Zig-zag manner, processing one block at a time with the block usually being 4×4 pixels

in size. Figure 1.1 shows the encoding and the decoding procedure of VQ. When VQ process one block, it finds the most similar codeword in the codebook, which is determined by Euclidian distance, as shown in Equation (1.1):

$$D(\text{block},\ \text{cw}) = \sqrt{\sum_{i=0}^{15} (\text{block}[i] - \text{cw}[i])^2}, \tag{1.1}$$

where block indicates the input block, and cw indicates the codeword in the codebook; block$[i]$ and cw$[i]$ indicate the i^{th} pixel value in the input block and in the codeword, respectively. Then, VQ outputs the index of the most similar codeword in the codebook to the index table as the compression code of the current input block. After processing every pixel block in the image, the compression code consists of two parts, i.e., the codebook and the index table. The decoder can recover the VQ indexed image by the codebook and he index table as Figure 1.1 shown.

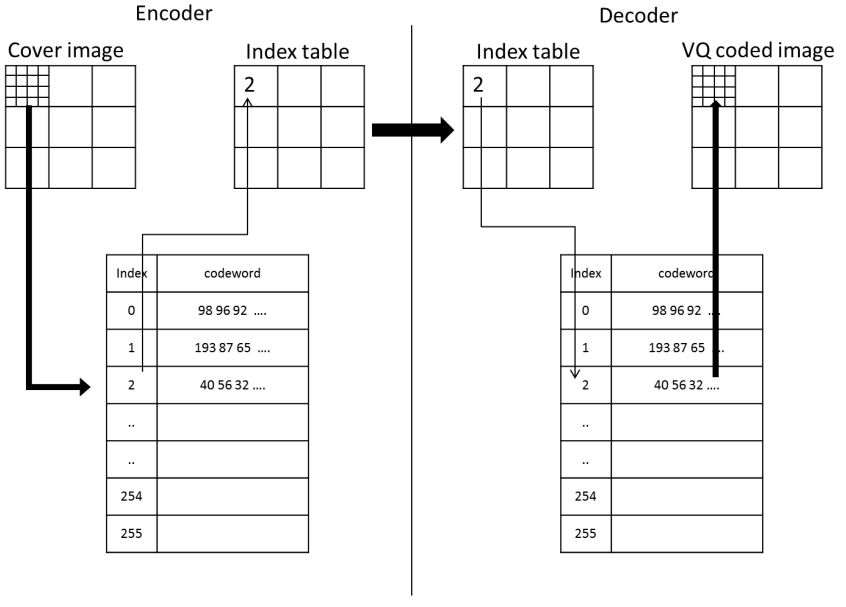


Fig. 1.1. The process of VQ compression.

1.2.2 Graph Coloring

Graph coloring includes vertex coloring, edge coloring, and face coloring [7]. In this work, graph coloring indicates vertex coloring, which assigns colors to the vertex of a graph so that none of the adjacent vertices has the same color. It is computationally difficult to color a graph. According to V. Guruswami et al.'s work [8], it is non-deterministic polynomial-time hard to color a three-colorable graph with four colors, meaning that it is impractical to color a graph that has a large number of vertices. So in this work, particle swarm optimization was selected to provide an approximate solution for a graph coloring problem [9].

1.2.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is an evolutionary algorithm [10]. It maintains a lot of particles, each of which represent a candidate solution to the problem. A particle noted as p has two properties,i.e., position, noted as x, and velocity, noted as v. PSO optimizes the solution to a problem by improving every candidate solution iteratively, as shown in Equation (1.2a) and (1.2b):

$$v(t+1) = w \times v(t) + R(c) \times (p(t) - x(t)) + R(c) \times (g(t) - x(t)), \quad (1.2a)$$

$$x(t+1) = x(t) + v(t+1), \quad (1.2b)$$

where, $x(t)$ and $v(t)$ indicate the position and the velocity of the particle at the t^{th} loop, respectively; $p(t)$ and $g(t)$ are the previous best position of the particle and the previous best position of all the particles in the t^{th} loop, respectively; w and c are constant coefficients ,and $R(c)$ is the uniform distribution on $[0,c]$. At the beginning of the algorithm, PSO initiates all the informants randomly through a pseudo-random number generator ,so that the results of different runs of PSO are different. So, in the implementation, a fixed number is selected to initiate the pseudo-random number generator. Due to the fact that the coloring result cannot be retrieved without this number, the fixed number can be treated as a key to protect the secret message, noted as KEY.

1.3 Proposed Scheme

In this section, the details of the proposed scheme are described.

Let I represent the cover image to be embedded in a binary message. The data hiding scheme begins after the construction of the codebook, noted as CB, by constructing a graph G. To construct G, the scheme adds a vertex, v, into graph G for every codeword in CB, so that every codeword has a corresponding vertex in G, noted as $v(CW)$. After adding every vertex into G, the scheme adds an edge into G for any two vertices v_1, v_2 such that the distance between their corresponding codewords, $D(cw_1, cw_2) \leq adj\_thresh$, where adj_thresh is a constant coefficient.

In the hiding scheme, every color can be denoted as an integer, so all the k colors, where k is the power of 2, can be represented as $C = \{0, 1, \cdots, k-1\}$, so that every color corresponds to $\log_2 k$ bits of secret message. For example, color 3 indicates '011' when k equals 8. After G is constructed, the scheme colors graph G with k different colors using the PSO algorithm, ensuring that any two adjacent vertices have different colors. The result of the coloring graph G is noted as $Color(v)$.

Next, graph G must be refined. The scheme checks every vertex, v in graph G, by iteratively testing whether a color can be found near v using a breadth-first search in a certain distance noted as bfs_thresh from v. Any vertex v that has colors that cannot be found around it will be deleted from G. This procedure is repeated until no vertex is deleted.

1.3.1 Embedding Phase

The detailed embedding procedure is illustrated in pseudo-code in this subsection.

Input: a cover image I and a secret message S_d
Output: codebook CB and a VQ compression code
Step 1. Generate codebook CB according to cover image I using *cell division*
Step 2. Construct a graph G with the following steps:
For every codeword CW in codebook CB,
add a vertex $v(CW)$ to graph G.
For any two vertices, v_1 and v_2 , in graph G,
add an edge between them if their corresponding codewords cw_1, cw_2 satisfying that $D(cw_1, cw_2) \leq adj\_thresh$.
Step 3. Color graph G with the PSO algorithm using a secret key KEY with k colors
Step 4. Refine graph G
For every vertex v in the graph G :
use a breadth-first search to determine whether, within a certain distance from the vertex v, there are vertices colored with all the k colors. If not, delete v from graph G.
Repeat this step until no vertex must be deleted.
Step 5. For a block CW_{input} in cover image I,
find its most similar codeword $CB_{most\_similar}$ word in CB.
If $v(CB_{most\_similar})$ exists in G
use a breadth-first search to find a vertex v' that meets the following requirements:

$$Color(v') = S_d,$$
$$D(v, v') \leq bfs\_thresh.$$

output the index of the corresponding codeword of v' as the compression code of the current input block.
go to process S_{d+1} with the next input block.
Else
out put the index of the current codeword $CW_{most\_similar}$ as the compression code of the current input block.
go to process S_d with the next input block.

1.3.2 Extracting Phase

The extracting phase is just the inverse procedure of the embedding phase. The scheme constructs graph G according to the received codebook CB. Then, the scheme uses PSO to color graph G with the same KEY to initiate the pseudo-random generator. The coloring result will be exactly same as the result in the embedding phase. When the scheme meets a VQ-compression code, *index*, the scheme checks whether there is vertex v in graph G that corresponds the $index^{th}$ codeword in the codebook. If there is, $Color(v)$ is just the information hidden in the VQ-compression code. The corresponding block of the decompressed image is just the

$index^{th}$ codeword in the codebook which is different from most of the other VQ-based data hiding schemes whose index table cannot be decompressed by the normal VQ-decompression procedure.

1.3.3 Embedding and Extracting Example

In this subsection, a detailed example of the proposed scheme is provided for illustration purposes.

Assume that we are using a codebook as Table 1.1 shows. The codebook, *CB*, contains 14 codewords, which is far less than the usual size of the codebook, 256,

Table 1.1. Codebook of the embedding and extracting example.

Index	Codeword
0	{183,185, 185, 184, 184, 185, 185, 184, 184, 185, 185, 184, 184, 185, 185, 183}
1	{179, 181, 181, 180, 180, 181, 181, 180, 180, 181, 181, 180, 180, 181, 181, 179}
2	{59, 57, 57 59, 57, 55, 55, 58, 57, 55, 54, 56, 60, 58, 58, 60}
3	{168, 169, 169, 168, 169, 170, 171, 169, 169, 171, 171, 169, 169, 170, 170, 167}
4	{164, 165, 165, 164, 165, 166, 167, 165, 165, 167, 167, 165, 165, 166, 166, 163}
5	{153, 155, 154, 153, 153, 155, 156, 154, 154, 156, 156, 154, 153, 154, 155, 153}
6	{149, 151, 150, 149, 149, 151, 152, 150, 150, 152, 152, 150, 149, 150, 151, 149}
7	{140, 141, 140, 138, 140, 141, 141, 140, 140, 142, 142, 141, 138, 140, 140, 139}
8	{136, 137, 136, 134, 136, 137, 137, 136, 136, 138, 138, 137, 134, 136, 136, 135}
9	{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
10	{126, 125, 125, 126, 127, 126, 126, 127, 128, 127, 127, 128, 127, 127, 127, 127}
11	{122, 121, 121, 122, 123, 122, 122, 123, 124, 123, 123, 124, 123, 123, 123, 123}
12	{102, 101, 102, 103, 101, 100, 100, 101, 102, 100, 99, 101, 102, 101, 101, 102}
13	{98, 97, 98, 99, 97, 96, 96, 97, 98, 96, 95, 97, 98, 97, 97, 98}

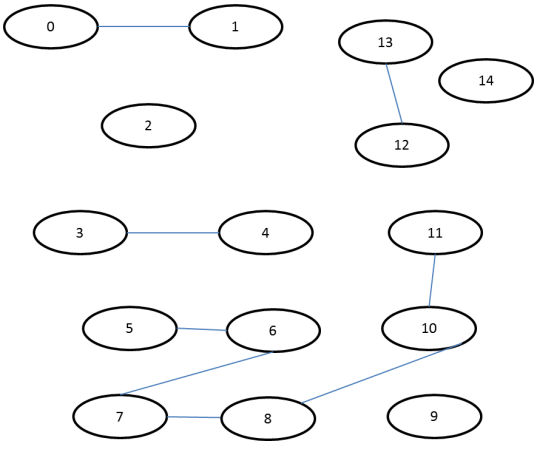


Fig. 1.2. Constructed graph G.

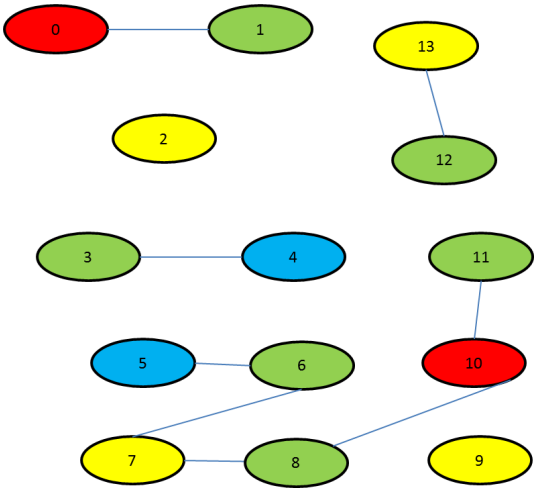


Fig. 1.3. Colored graph G.

but enough to illustrate our scheme. In the example, the adj_thresh is set at 41 while the bfs_thresh is set as 150. As in the first step, Graph G is constructed by adding a vertex for every codeword, so that every codeword has a corresponding vertex. The edges are then added to graph G. The proposed scheme iteratively tests whether the distance between two vertices is less than the adj_thresh. If it is, the scheme adds an edge between the two vertices. For example, $D(cw_0, cw_1)$=16 is less than adj_thresh, 41, so there is an edge between them where cw_i indicates the codeword whose index is i. The constructed graph G is as Figure 1.2 shows.

Then graph G is then colored with k different colors, so that no adjacent two vertices have the same color. Here, due to the small size of CB, many methods can

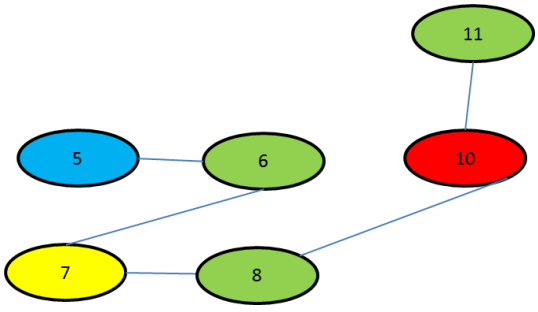


Fig. 1.4. Refined graph G.

be applied to resolve the graph coloring issue. However, the PSO can provide good security, so it is still adopted in this work. The PSO colored graph G is shown as Figure 1.3.

The next step in the process is to refine graph G. The proposed scheme deletes vertex, v, that there are colors that cannot be found using breadth-first search in a certain distance from v, such as vertex 0 and vertex 1. The result is as Figure 1.4.

After having been refined, graph G can be used to hide data. The proposed scheme scans the cover image in a Zig-zag manner, processing one block by one block. In the image 'baboon', the first block, cw_{input_1}, is $\{46, 46, 46, 63, 46, 46, 46, 63, 51, 51, 51, 51, 92, 92, 92, 53\}$, due to $D(cw_{input\_1}, cw_2)$ being the smallest one. The most similar codeword for cw_{input_1} is cw_2, that has no corresponding vertex in refined graph G. Therefore, the scheme outputs 2 as the compression code. In the image 'baboon', the sixth block, cw_{input_2} is $\{115, 167, 169, 103, 115, 167, 169, 103, 96, 164, 197, 166, 71, 154, 196, 179\}$, its most similar codeword is cw_6, which exists in graph G. Assuming the secret data is '00' which is color 0, the scheme found a vertex around cw_6 colored with color 0 which is vertex 10 in graph G. As a result, the scheme outputs 10 instead of 2 as the compression code that is embedded with secret data '00'.

In the extracting phase, the scheme constructs, colors and refines graph G exactly in the same manner as what the scheme has done in embedding phase. So in extracting phase, the decoder can achieve a same graph G as Figure 1.4. For compression code 2, if there is no vertex in graph G corresponding to codeword 2, there is no secret data inside the compression code; while for compression code 10 , the scheme finds that vertex 10 exists, so color(v), i.e. '00' is the secret data and cw_{10} is the pixel block that should be output as the VQ-coded image.

1.4 Experimental Results

In this section, some experimental results are presented to prove the validity of the proposed scheme. The experiments were conducted on an Intel Core2 Duo P7450 computer at 2.13 GHz, and they were implemented in C using Intel Open CV 2.1.0 and Standard PSO in C [11]. Figure 1.5 shows the cover images used in the

experiment in which Lena and Tiffany are smooth images; Baboon is a complicate image; the others are normal images. The cover images are all 512×512 pixels in size.

In image steganography, embedding capacity indicates the number of secret bits that can be hidden in a cover image. Usually, peak signal-to-noise ratio (PSNR) is favored to evaluate the quality of the stego-images. The definition of PSNR is illustrated in Equation (1.3a).

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right), \tag{1.3a}$$

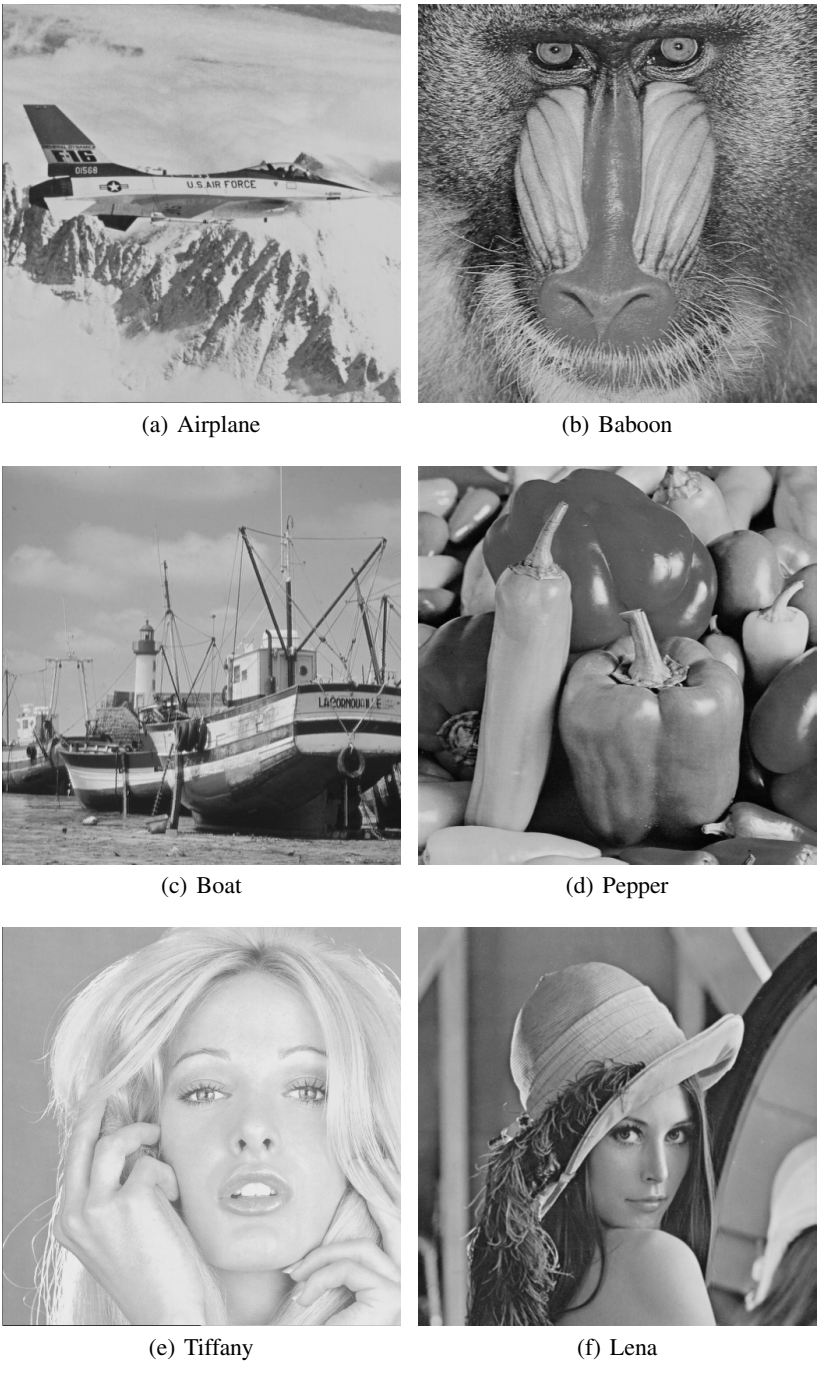
$$\text{MSE} = \frac{1}{W \times H}\sum_{x=0}^{H-1}\sum_{y=0}^{W-1}\left(I(x,y) - I'(x,y)\right)^2, \tag{1.3b}$$

where I and I' denote the cover image and the stego-image, respectively, both of which have the same size. $I(x,y)$ denotes the pixel value of the cover image at the x^{th} row and y^{th} column. W and H denote the width and the height of the image, respectively. PSNR is widely used in the evaluation of image quality. A higher PSNR indicates better quality of the stego-image.

In the proposed scheme, 4 parameters, i.e., k, adj_thresh, bfs_thresh and the size of the codebook, influence the experimental results. The symbol k controls the number of all colors used to color graph G; adj_thresh controls the distance between the vertices of graph G; bfs_thresh controls the radius of the breadth-first search, and ensures that the data hidden in the compression code is recoverable through the refine step; the size of the codebook influences the generation of codebook which is the base of graph G.

Figure 1.6 shows the influence of adj_thresh on the proposed scheme. In this experiment, the size of codebook is set as 512. It can be observed that as the adj_thresh increases, the embedding capacity also increases while the PSNR decreases either. However the increment of the embedding capacity is not stable, as such, sometimes the embedding capacity even decreases. This is a result of when adj_thresh increases, more edges can be added to graph G leading to a more complicated graph G which is harder for the PSO to color.

Figure 1.7 illustrates the variation of the embedding capacity and the PSNR when bfs_thresh varies. When $k = 8$, the variation is minimal. It is normal that the embedding capacity only minimally increases, while the PSNR also minimally decreases, because an increase in bfs_thresh means that the radius of the breadth-first search is larger, so that the difference between the original index and the stego-index may be larger, which will finally lead to a decrease in PSNR. When k is set to 32, the variation is significant. In this case, when bfs_thresh is in the range of 20 to 40, the embedding capacity is 0. However, when the bfs_thresh is larger than 40, the embedding capacity increases stably. The PSNR also varies significantly, due to bfs_thresh being less than 40. Because of this, there are not enough vertices around a vertex to be colored with 32 different colors, so that there is no data embedded and the PSNR is the same as the VQ-coded image's PSNR. This means that when k increases, the bfs_thresh should also increase.

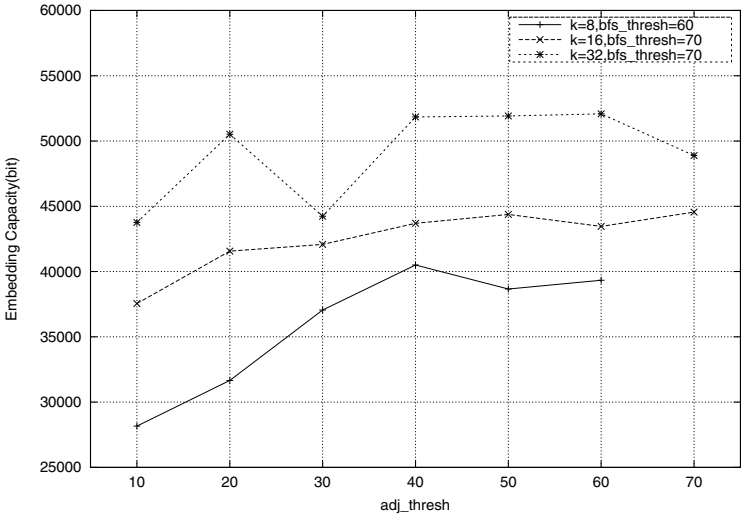


(a) Airplane (b) Baboon

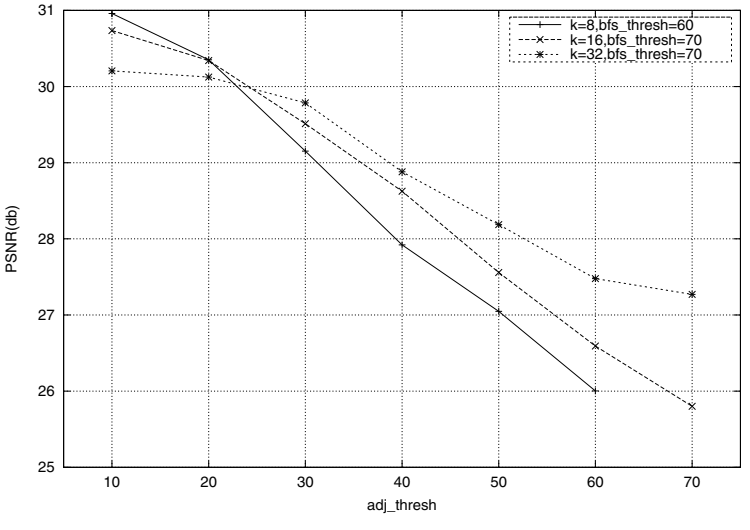
(c) Boat (d) Pepper

(e) Tiffany (f) Lena

Fig. 1.5. Images used to test the proposed scheme.

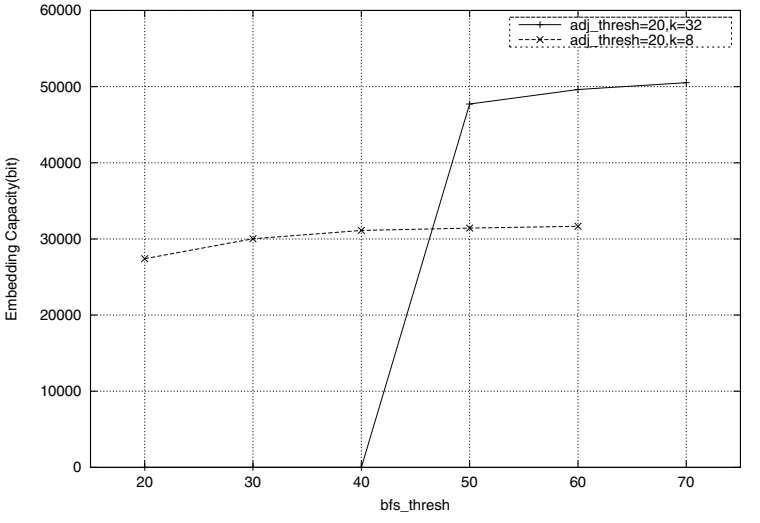


(a) Embedding capacity

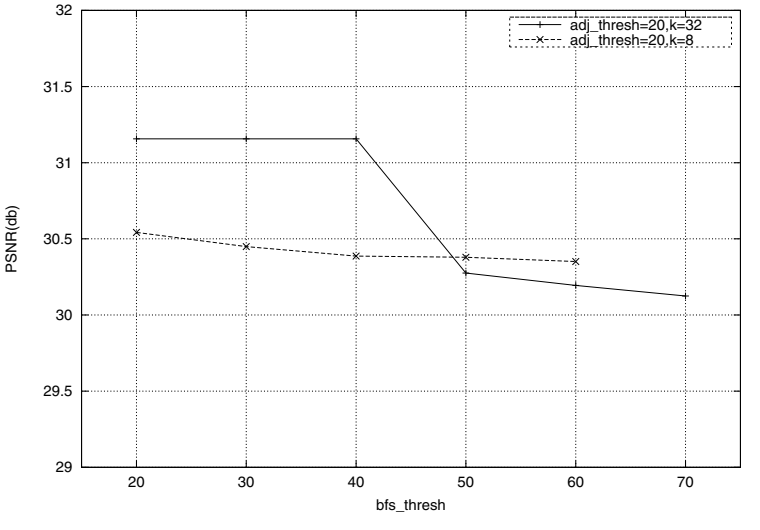


(b) PSNR

Fig. 1.6. Influence of adj_thresh for image "Airplane".

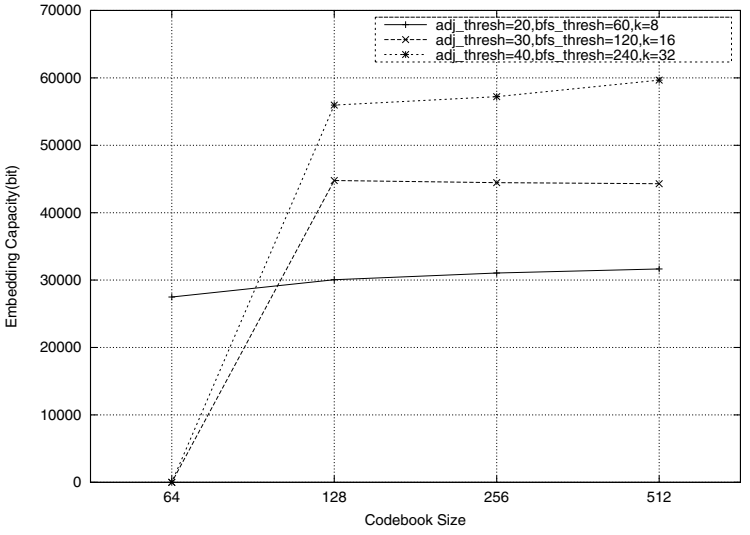


(a) Embedding capacity

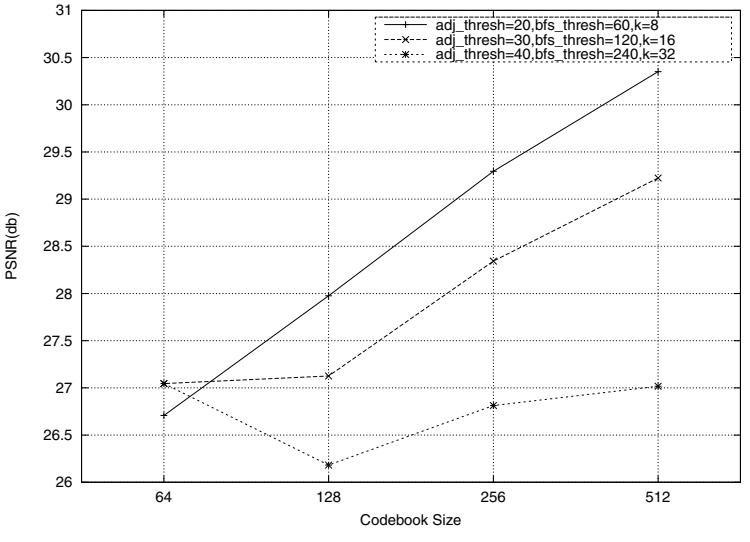


(b) PSNR

Fig. 1.7. Influence of bfs_thresh for image "Airplane".

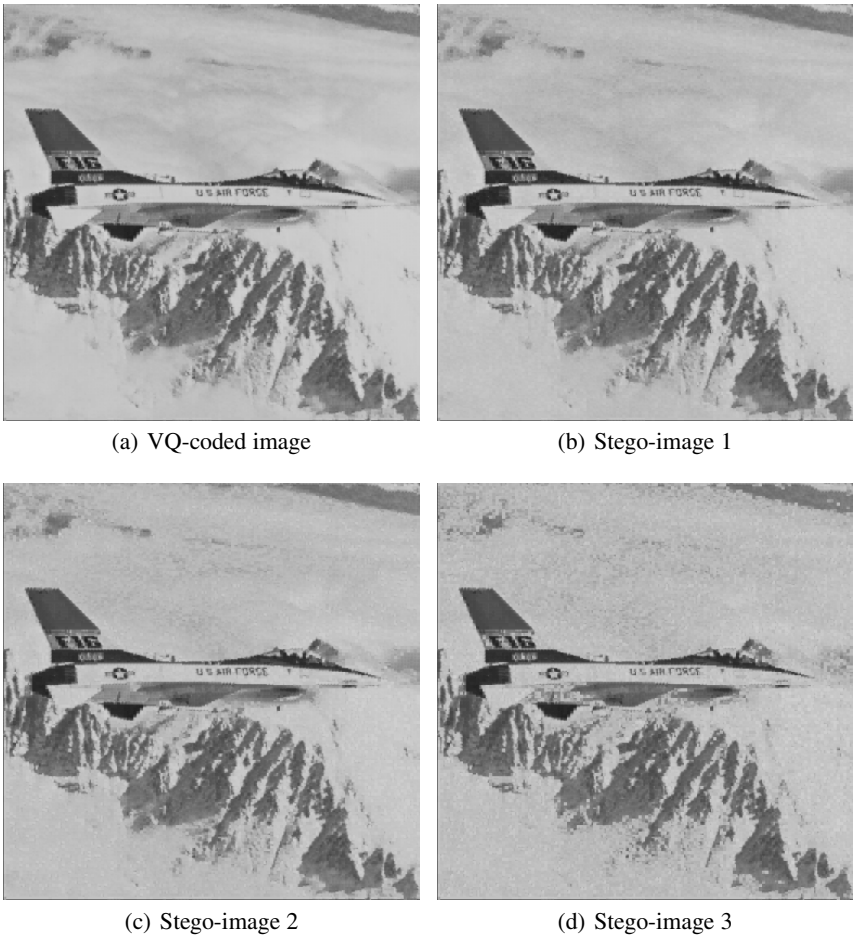


(a) Embedding capacity



(b) PSNR

Fig. 1.8. Influence of the size of codebook for image “Airplane”.



(a) VQ-coded image (b) Stego-image 1

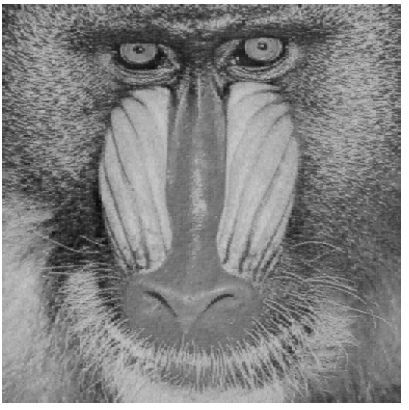
(c) Stego-image 2 (d) Stego-image 3

Fig. 1.9. Image "Airplane" and its stego-images.

The influence of the size of the codebook is shown in Figure 1.8. It can be observed that as the size of the codebook increases, the embedding capacity and the PSNR increases simultaneously. This result is different from the influence of bfs_thresh. It occurs because the size of the codebook can improve the quality of the VQ-coded image and the improvement is usually larger than the negative influence caused by the embedding of the secret data. However, sometimes the embedding of secret data counterbalances the improvement brought about by the increase of codebook size which can depicted in Figure 1.8 when adj_thresh, bfs_thresh, k and codebook size is 40, 240, 32, and 128, respectively.

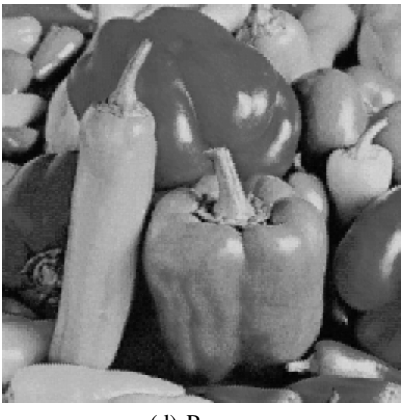
In Figure 1.9, the resulting images of the proposed scheme with different parameters are presented. The sizes of the codebook of different image are all set as 512. Figure 1.9(a) shows the original VQ-coded image; stego-images 1,2 and 3 are the

(a) Airplane

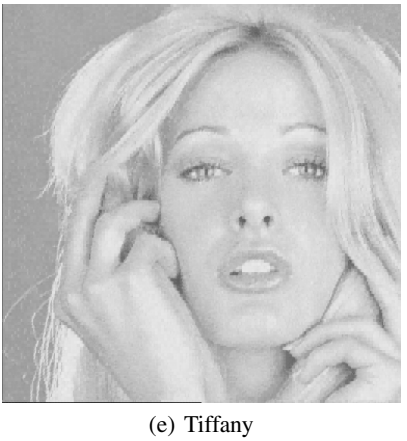


(b) Baboon

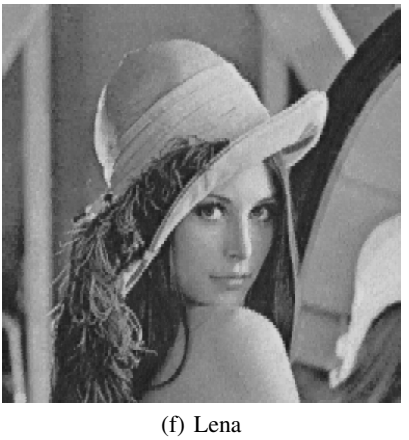
(c) Boat



(d) Pepper



(e) Tiffany



(f) Lena

Fig. 1.10. Stego-images.

results of the proposed scheme when adj_thresh is 20, 30, 40; bfs_thresh is 60, 120, 240; k is 8, 16, 32, respectively. Compared with the original VQ-coded image, the modification in stego-image 1 is totally imperceptible and the modification in stego-image 2 is still acceptable. However, the modification in stego-image 4 is too obvious. Figure 1.10 illustrates the stego-images of the proposed scheme with the same parameter as the stego-image 1 in Figure 1.9. It can be observed that our proposed scheme performs well on all the cover images. The embedding capacity and the PSNR of the images in Figure 1.10 are shown in Table 1.2.

Table 1.2. Performance of the proposed scheme.

Picture	$adj\_thresh = 20$, $bfs\_thresh = 60$, $k = 8$		$adj\_thresh = 30$, $bfs\_thresh = 120$, $k = 16$		$adj\_thresh = 40$, $bfs\_thresh = 240$, $k = 32$	
	bits	PSNR	bits	PSNR	bits	PSNR
Airplane	31659	30.351415	44304	29.224346	59660	27.017768
Baboon	11772	24.373809	20332	23.95553	0	24.540745
Boat	33492	28.447689	51116	27.212622	59660	25.451332
Pepper	29829	29.387094	51744	27.496079	44845	25.822918
Tiffany	44499	31.341663	61480	29.638875	79550	27.532017
Lena	39786	29.526763	46384	28.46273	78240	24.821300

1.5 Conclusions

In this work, we proposed a scheme based on VQ and graph coloring that can provide both good quality stego-images and good embedding capacity stego-images, which means our scheme is very flexible. The stego-index table that our proposed scheme outputs can be decompressed into a meaningful picture, while most of the other VQ-based data hiding schemes cannot accomplish this, which means our proposed scheme is valid and useful to steganography. Also, the proposed scheme manipulates PSO as a coloring method, which provides better security since the number to initiate the pseudo-random number generator is, in fact, a key. In addition, graph coloring was successfully applied in the image data hiding based on vector quantization. Therefore, our scheme provides flexibility, security, and innovation.

References

1. Duric, Z., Jacobs, M., Jajodia, S.: Information hiding: Steganography and steganalysis. Handbook of Statistics 24, 171–187 (2005)
2. Tefas, A., Nikolaidis, N., Pitas, I.: Image watermarking: Techniques and applications. The Essential Guide to Image Processing 22, 597–648 (2009)

3. Lin, Y.C., Wang, C.C.: Digital images watermarking by vector quantization. Optical Engineering 3, 76–78 (1999)
4. Wu, H.C., Wu, N.I., Tsai, C.S., Hwang, M.S.: Image steganographic scheme based on pixel-value differencing and lsb replacement methods. Vision, Image and Signal Processing 152, 611–615 (2005)
5. Li, Y., Li, C.T.: Steganographic scheme for vq compressed images using progressive exponential clustering. In: Proc. IEEE Int'l. Conf. Video and Signal Based Surveillance, p. 85 (2006)
6. Cosman, P.C., Oehler, K.L., Riskin, E.A., Gray, R.M.: Using vector quantization for image processing. Proceedings of the IEEE 81, 1326–1341 (1993)
7. Yanez, J., Ramirez, J.: The robust coloring problem. European Journal of Operational Research 148, 546–558 (2003)
8. Guruswami, V., Khanna, S.: On the hardness of 4-coloring a 3-collorable graph. In: Proc. 15th Annual IEEE Conf. Computational Complexity, pp. 188–197 (2000)
9. Cui, G., Qi, L., Liu, S., Wang, Y., Zhang, X., Cao, X.: Modified PSO algorithm for solving planar graph coloring problem. Progress in Natural Science 18, 353–357 (2008)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Int'l Conf. Neural Networks, vol. 4, pp. 1942–1948 (1995)
11. Standard PSO (2007), `http://www.particleswarm.info/Programs.html`