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

A New Adaptive Steganographic Method Based on Gradient Adjacent Prediction and Side-Match Vector Quantization

Shiau-Rung Tsui[1], Cheng-Ta Huang[1]
Department of Computer Science and Information Engineering
National Central University, Taiwan
Software Research Center, National Central University, Taiwan
hjovulvul1223@gmail.com, ppp0936@gmail.com

Wei-Jen Wang[1,2]
[1]Department of Computer Science and Information Engineering
National Central University, Taiwan
[2]Software Research Center, National Central University, Taiwan
National Central University, Taiwan
wjwang@csie.ncu.edu.tw

Receied February, 2013; revised April, 2013

Abstract. *This study presents a new steganographic method that embeds secret data into a cover digital image using VQ encoding. The core concept of the proposed method uses the gradient adjacent prediction (GAP) algorithm, which enhances prediction accuracy of neighboring blocks in SMVQ encoding. To embed secret data into the cover image, the proposed method utilizes the features of GAP to decide the capacity of the secret data per pixel in a block. It then embeds the secret data accordingly. It also embeds an index value in each block to ensure that the secret data can be recovered back. The index value points to the closest codeword of a state codebook to the encoding block, where the state codebook is generated by GAP-based SMVQ. The result shows that the proposed method has better performance than a recent similar work proposed by Chen and Lin in 2010.*

Keywords: Steganography, Gradient adjacent prediction (GAP), VQ, SMVQ

1. Introduction.

Steganographic methods for digital images have been widely studied in the last decade. They employ various encoding techniques to embed secret data into the cover image, and to produce the stego-image carrying secret data. Those methods have been used in many types of applications, such as watermarking and secret communication. Encoding techniques used in those methods can be assorted to three kinds of technique domains - the spatial domain [1-5], the frequency domain [6-8], and the compression (substitution) domain [6], [7], [9-13]. Each technique domain has different features. For example, the spatial domain usually offers large embedding capacity for secret data and excellent visual quality for stego-images, but may not pass statistical steganalysis. On the other hand, the compression domain performs better in statistical steganalysis, but may produce less embedding capacity for secret data and lower visual quality for stego-images. As for the frequency domain, a method of the domain is usually used in watermarking applications since it is more robust against image distortion attacks.

This study focuses on developing a new steganographic method in the compression (substitution) domain. The most popular technique in this domain is to use an algorithm based on vector quantization (VQ) [14], to encode the cover image along with the secret data. The VQ uses a codebook to quantize an image to non-overlapping blocks of 4×4 pixels. In VQ, each block in the image is replaced by a short index value, which points to a codeword of the codebook and the codeword can best represent the block. Side match vector quantization (SMVQ) [15] is a VQ-based encoding approach that is also popular in image steganography. It is based on the observation that neighboring pixels are usually similar. The property can be utilized to construct state codebooks using smaller index size to improve compression rate, which can create larger capacity for embedding secret data. The gradient adjacent prediction (GAP) algorithm [17] can be used to improve SMVQ compression. It uses seven neighboring pixels to predict what the current encoding/decoding block should look like. Each successful prediction can potentially improve the compression rate of the SMVQ encoding method. This algorithm has the advantage of being able to detect the feature of the vertical side and the horizontal side precisely. This advantage can be used to improve the compression rate of SMVQ.

Most VQ-based steganographic methods produce VQ-based stego-images as the output; some VQ-based steganographic methods produce raw image as the output. This study focuses on the latter since the types of steganographic methods have much room for performance improvement. In 2006, Chang et al. proposed an SMVQ-based method that produces raw images as the output stego-images [11]. To encode a block, the method uses the two closest codewords to the block in the codebook, and constructs a substitute block by applying an adjustable weight ratio to the two closest codewords. A secret bit can be embedded in each block by adjusting different ratios. In 2010, Chen and Lin proposed two similar steganographic methods [13] that produce raw images as the output. It adopts the least-significant-bit embedding approach [16] and the concept of codeword radius from Chang et al.'s method. The methods, compared with prior similar studies, provide significantly larger embedding capacity for secret data. This is because the methods can embed multiple secret bits in each pixel of an image block.

Section 2.1 introduces Chen and Lin's methods and their limitations. Section 2.2 introduces GAP prediction and their spirits. Section 3 describes the proposed method. Section 4 summarizes the experimental results and makes comparisons to other methods. Section 5 draws the conclusions.

2. Related works.

In this section, we will introduce two related works, Chen and Lin's method, and gradient adjacent prediction (GAP).

2.1. Chen and Lin's steganographic method.

In section 2.1, we will briefly introduce the concepts of the two methods proposed by Chen and Lin [13]. Both the methods are very similar and employ the same concept, the distance radius concept. That is, their methods need to define a distance radius for each used codeword in the codebook. To encode a block R, the method retrieves the closest codeword X to R and the closest codeword Y to X from the codebook. The half of the distance between X and Y is the distance radius of X, namely t. Any block value within the distance radius can be mapped to X, which represents the same result of using SMVQ to map block R into a codeword. Given the integer representation of the secret bits I, the encoded value $X + I$ can be extracted if I is smaller than t. This is because $X + I$ is always mapped to X under SMVQ encoding. The steganographic method proposed by Chen and Lin is described below:

Input: Cover image G, Codebook C, Secret data S

Output: Stego-image G'
STEP 1: Divide the image G into non-overlapping blocks of 4×4 pixels.
STEP 2: For each block R, do Steps 3 to 6.
STEP 3: If R is on the first row/column, do VQ encoding. If not, go to Step 4.
STEP 4: Do SMVQ encoding to establish state codebook.
STEP 5: Find the closest codeword X to R and the closest codeword Y to X in the state codebook. Set $v = [\log_2(\|X - Y\|/2)]$ to be the largest embedding capacity for each pixel of R.
STEP 6: Retrieve v bits of secret data, namely I, and then replace the least significant v bits of X by the secret bits I. The block is saved in Z. Go back to Step 2 until all blocks are processed.
STEP 7: Output Stego-image Z.

Chen and Lin proposed another aggressive encoding method based on the above steganographic method. They pointed out that, for a 4×4 block, the constraint for embedding v bits per pixel is:

$$16(2^v - 1)^2 \le t^2 \tag{1}$$

Since the embedding is pixel-based, each pixel can embed different number of bits. As a result, they proposed an aggressive encoding method, which finds the largest v and then embeds one more bit ($v + 1$ bits) for the first k pixels of the block and v bits for the remaining pixels of the block, such that the following equation holds.:

$$k\left(2^{v+1} - 1\right)^2 + (16 - k)\left(2^v - 1\right)^2 \le t^2 \tag{2}$$

2.2. **Gradient Adjacent Prediction (GAP).** The GAP algorithm utilizes the seven known pixels to analyze the feature of a region of pixels, and then predicts what an unknown pixel should be. Figure 1 demonstrates the concept of the GAP algorithm, in which x is the pixel to be predicted and $a \sim g$ are decoded neighboring pixels. We use six types of features to define a region of pixels: “sharp horizontal,” “normal horizontal,” “weak horizontal,” “sharp vertical,” “normal vertical” and “weak vertical.” By applying GAP prediction, the value of x' can be calculated according to the following piece of code:

Input: Pixels a, b, c, d, e, f, g
Output: Predicted pixel x', Feature F

```
$d_h = |a - e| + |b - c| + |b - d|$            //view of the horizontal differences
$d_v = |a - c| + |b - f| + |d - g|$            //view of the vertical differences
if($d_v - d_h > 80$)                           {x' =a, $F$=“sharp horizontal”}
else if($d_v - d_h < -80$)                     {x' =b, $F$=“sharp vertical”}
else if($d_v - d_h > 32$)          {$x'$ =((a+b)/2+(d-c)/4+a)/2, $F$=“normal horizontal”}
else if($d_v - d_h < -32$)           {$x'$ =((a+b)/2+(d-c)/4+b)/2, $F$=“normal vertical”}
else if($d_v - d_h > 8$)          {$x'$ =(3((a+b)/2+(d-c)/4)+a)/4, $F$=“weak horizontal”}
else if($d_v - d_h < -8$)           {$x'$ =(3((a+b)/2+(d-c)/4)+b)/4, $F$=“weak vertical”}
return $x'$ and $F$
```

For example (see Figure 2), $a = 129, b = 155, c = 139, d = 153, e = 120, f = 82, g = 80$ and original value $x = 158$. After calculating the feature of the pixels, $d_h = 27$ and $d_v = 146$. From the branch condition of GAP algorithm, the feature of x is “sharp horizontal” ($146 - 27 = 119 > 80$) that is almost located at the horizontal line. Finally, the x' is 155 which is very close to the original value x. According to the neighboring pixels in first row and the second row, we can find out the difference directly. From the

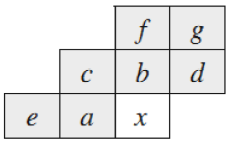

Figure 1. To predict pixel x using seven neighboring pixels $a, b, \ldots, g$, defined by the GAP algorithm.

pixels (82, and 80) in the first row and the pixels (139, 155, and 153) in the second row, we also can easily figure out that the relationship is Horizontal by the human visual system.

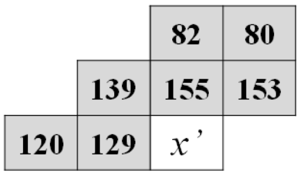

Figure 2. An example of the GAP prediction

3. **Proposed method.** Traditional VQ encoding uses a codebook to encode an image for the purpose of data compression. The SMVQ and GAP can be used to compress the VQ-encoded image to a smaller representation. The proposed method uses all of the techniques to encode the cover image along with the secret data. Since the proposed method produces a raw image as the output, it does not need to follow the steps of SMVQ and GAP encoding strictly. Our strategy is to leave the blocks in the first row/column as they are, and then to perform SMVQ and GAP on other blocks.

We adopted the gradient adjacent prediction (GAP) algorithm [17] that uses seven neighboring pixels to predict what a pixel should look like. Note that the GAP algorithm is not a VQ-based encoding scheme. It can be used to replace the side-match concept of the SMVQ.

Based on SMVQ and GAP, the proposed steganographic method can achieve higher PSNR than what VQ/SMVQ encoding can achieve. In the embedding procedure, the proposed method partitions the input image into blocks of 4×4 pixels. Then, it uses the LSB method [19] to embed an index value, which points to the most similar codeword to the block, into the first n pixels of each image block, where $n = 1 \sim 3$. The remaining pixels of the block are used to embed secret bits. To achieve this purpose effectively, it uses GAP to decide the number of bits to be embedded. The following steps show how the embedding procedure works:

Input: Cover image Z, codebook C, state codebook size 2^k and secret data S
Output: Stego-image Z'
STEP 1: Divide the image Z into non-overlapping blocks of 4×4 pixels.
STEP 2: For each block R, do Steps 3 to 7.
STEP 3: If R is on the first row/column, save R in Z' and go to bsck to Step 2. Otherwise, go to Step 4.
STEP 4: Use GAP to predict what R should be and get R'. Generate a state codebook SC by R' and codebook C, and then use the state codebook SC to get an index value pointing to the most similar codeword g.
STEP 5: Partition the index value (*i.e.* the size is k bits) into $1 \sim 3$ three-bit segments. Use LSB to embed the segments into the first n pixels except the last segment. The last

segment may not contain 3 bits because k is not dividable by 3. Assume the size of the last segment is m. Embed m bits by LSB into the corresponding pixel.
STEP 6: The GAP algorithm labels the uppermost pixels and the leftmost pixels with their property: "sharp", "weak", or "normal."" Count the total number of each label appeared in the block. If "sharp" dominates, retrieve four bits from the secret data and embed them into each of the rest of the pixels. If "normal" dominates, retrieve and embed three bits. Otherwise, retrieve and embed one bit. For each remaining pixel that corresponds to the position p in the block, the embedding approach adds or subtracts the value of the secret bits to the g_p, depending on which operation results in better visual quality.
STEP 7: Combine the pixels generated in Steps 5 and 6 to create a new stego-image block. Save the block in Z'. Go back to Step 2 until all blocks are processed.
STEP 8: Output stego-image Z'.

The secret data extraction procedure uses GAP to generate a state codebook for each image block, and uses LSB to restores an index value from the first to the third pixels. Then we use GAP to calculate the number of secret bits we had embedded in each of the remaining pixels. The secret data extraction procedure is described below:

Input: Stego-image Z', codebook C and state codebook size 2^k
Output: Secret data S
STEP 1: Divide the stego-image Z' into non-overlapping blocks of 4×4 pixels.
STEP 2: For each block R, do Steps 3 to 7.
STEP 3: If R is on the first row/column, do nothing and return to Step 2. Otherwise, go to Step 4.
STEP 4: Use GAP to predict what R should be and get R'. Use R' and codebook C to generate a state codebook SC.
STEP 5: Retrieve $[k/3]$ three-bit segments from the least significant three bits of the first $[k/3]$ pixels except the last segment. Read only $k mod$ 3 bits from the corresponding pixel that embeds the last segment. Recover the index value and obtain the codeword g of the block.
STEP 6: Use GAP to see which property ("sharp", "weak", or "normal") of block R' dominates. If "sharp" dominates, retrieve three secret bits s from each of the rest of the pixels. If "normal" dominates, retrieve two bits. Otherwise, retrieve one bit. For each remaining pixel h that corresponds to the position p in the block, the retrieval procedure calculates the absolute difference between the g_p and h, and then gets the secret bits s.
STEP 7: Append s to S. Go back to Step 2 until all block are processed.
STEP 8: Output secret data S.

4. Experimental results.

We have conducted some experiments to evaluate the performance of the proposed method, and show the results in this section. We used six 512×512 gray-level images for performance evaluation -"Lena," "Baboon," "Sailboat" "Pepper," "F16," and "Boat," as shown in Figure 3. In the experiments, we used two different codebooks, one with 256 codewords and the other with 512 codewords, respectively. The codebooks are generated by the LBG algorithm [18], given many images for training.

Table 1 and Table 2 show the visual quality of the testing images, encoded by VQ, SMVQ, and GAP-based SMVQ respectively. The results of Table 1 and Table 2 are generated by using a codebook of 256 codewords and a codebook of 512 codewords, respectively. Note that the visual quality of an SMVQ-encoded image is always worse than the visual quality of a VQ-encoded image. However, a GAP-based-SMVQ-encoded image can potentially achieve better visual quality than a VQ-encoded image because we use raw image blocks in the first row/column. Generally, a GAP-based-SMVQ-encoded image has better visual quality than an SMVQ-encoded image in terms of PSNR. The improvement comes from the accurate prediction scheme of GAP. From Table 1 and Table 2, we found that GAP-based SMVQ could enhance SMVQ image quality.

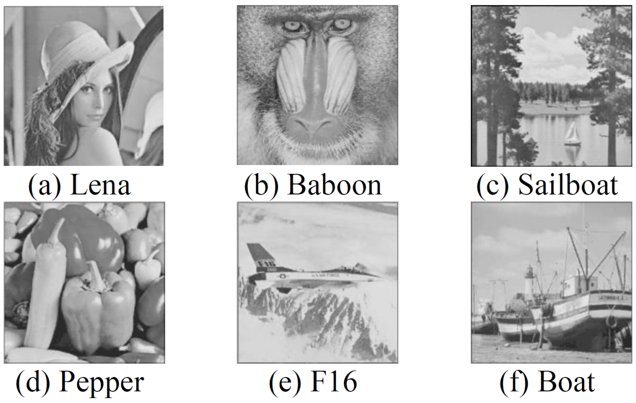

FIGURE 3. The six test cover images, each of which has a size of 512×512 pixels

TABLE 1. Image quality in terms of PSNR for different compression strategies, given different sizes of state codebooks and a codebook of 256 codewords.

Image	VQ	16-Codeword State Codebook		64-Codeword State Codebook		128-Codeword State Codebook	
		SMVQ	GAP	SMVQ	GAP	SMVQ	GAP
Lena	31.373	26.731	27.278	30.088	30.169	31.049	31.107
Baboon	24.452	22.267	22.375	23.642	23.842	24.180	24.213
Sailboat	28.622	25.359	25.585	27.792	27.956	28.435	28.684
Pepper	30.728	26.714	27.291	29.787	29.878	30.543	30.562
F16	30.582	26.227	26.957	29.389	29.511	30.228	30.411
Boat	29.387	25.863	25.840	28.321	28.348	29.067	29.146

TABLE 2. Image quality in terms of PSNR for different compression strategies, given different sizes of state codebooks and a codebook of 512 codewords.

Image	VQ	16-Codeword State Codebook		64-Codeword State Codebook		256-Codeword State Codebook	
		SMVQ	GAP	SMVQ	GAP	SMVQ	GAP
Lena	32.248	25.898	26.836	29.776	29.910	32.115	32.079
Baboon	24.703	21.947	22.176	23.349	23.398	24.602	24.648
Sailboat	29.251	24.569	25.072	27.433	27.635	29.141	29.222
Pepper	31.408	26.060	26.662	29.565	29.736	31.306	31.342
F16	31.578	25.705	26.166	29.330	29.350	31.412	31.561
Boat	30.163	25.044	25.639	28.072	28.106	30.063	30.657

In Table 3 and Table 4, we can see that all the embedding capacity values are close, and so are the visual quality values, given different sizes of state codebook and different codebooks. Although the difference is small, we still can find that the visual quality goes up as the size of the state codebook increases if we are given the same codebook. On the other hand, the embedding capacity decreases as the size of the state codebook increases. Note that the embedding capacity is the same for state codebook sizes 16 and 32. This is because our method decides the number of pixels for embedding an index value by $[k/3]$, given the state codebook size 2^k. For example, a state codebook with 2^8 codewords should be partitioned into $[(\log_2 8)/3]$ segments, implying the use of the first pixel to embed the index value. When the size of the state codebook doubles, we need $[(\log_2 16)/3]$ segments, implying the use of the first two pixels to embed the index value.

Table 3. The visual quality in PSNR and the embedding capacity (EC) in bits of the stegoimages produced by the proposed method, given different sizes of state codebooks and a codebook of 256 codewords.

Image	State codebook size							
	8		16		32		128	
	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC
Lena	26.339	518910	28.044	529018	29.781	527240	31.877	489099
Baboon	22.561	614813	23.111	596904	23.644	599844	24.953	554320
Sailboat	25.268	551855	26.191	547064	28.126	546070	29.825	504556
Pepper	26.274	528911	28.048	542434	29.778	532574	31.641	492089
F16	25.840	469413	27.693	477172	29.025	486612	31.345	449644
Boat	24.923	508001	26.597	508550	28.293	507640	30.162	469313

Table 4. The visual quality in PSNR and the embedding capacity (EC) in bits of the stegoimages produced by the proposed method, given different sizes of state codebooks and a codebook of 512 codewords.

Image	State codebook size							
	8		16		32		256	
	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC
Lena	26.254	510894	27.642	523334	28.937	521136	32.906	482404
Baboon	22.219	622877	23.147	608146	23.426	612220	25.266	566761
Sailboat	24.523	537813	25.835	534562	27.174	532798	30.311	490555
Pepper	25.634	525712	27.400	529256	28.802	526764	32.293	484757
F16	25.261	441601	26.882	460292	28.587	459480	32.311	424541
Boat	24.536	505854	25.982	518490	27.578	518112	31.155	480025

Table 5. Comparison of the proposed method and Chen and Lin's method [13], given the state codebook size of 16 codewords and a codebook of 512 codewords.

Image	Proposed method		Chen and Lin	
	PSNR	EC(bit)	PSNR	EC(bit)
Lena	27.642	523334	26.512	355952
Baboon	23.147	608146	22.155	456286
Sailboat	25.835	534562	25.274	424587
Pepper	27.400	529256	26.804	374148
F16	26.882	460292	26.386	296154
Boat	25.982	518490	25.786	375021
average	26.148	528897	25.486	380358

Table 6. The visual quality in PSNR and the embedding capacity (EC) in bits of the stego-image Lena using different embedding capacity for "Sharp", "Normal", and "Weak" blocks, given the state codebook size of 16 codewords and a codebook of 512 codewords.

GAP Setting: Sharp/Normal/Weak	Lena PSNR	Lena EC(bit)
5/4/1	27.073	270858
5/4/2	27.251	481824
5/4/3	27.681	692790
5/4/4	27.398	903756
5/3/1	26.957	256550
5/3/2	27.302	467516
5/3/3	27.698	678482
4/3/1	27.068	256108
4/3/2	27.247	466984
4/3/3	27.023	677950
4/2/1	26.971	241710
4/2/2	27.318	452676
3/2/2	27.115	452144
3/2/1	27.039	241778

To show how good the proposed steganographic method is, we compare the proposed method with Chen and Lin's first steganographic method, which is proposed in [13]. We do not compare ours with Chen and Lin's aggressive embedding method (the second method) in [13] because it is not applicable to all images, as explained in Section 2. The comparison results are shown in Table 5. The proposed method achieves better visual quality for all testing images. The average improvement rate is 2.5% better. As for the embedding capacity, the proposed method achieves significantly larger embedding capacity. The embedding capacity provided by the proposed method is 28% better. The results indicate that the proposed method is a much better steganographic method in the compression domain.

To optimize the image quality and embedding capacity, we can adjust the embedding capacity strategy according to the property of a block, defined by our GAP algorithm. We found that, the embedding capacity for different types of blocks should range from 5 bits to 1 bit. This is because, when the embedding capacity per pixel is larger than 5 bits, the image quality of the stego-image is unexpectedly worse than the image quality of the corresponding SMVQ-GAP image. Table 6 shows the results of embedding secret data into image Lena, given the state codebook size of 16 codewords and a codebook of 512 codewords. The results show that, we can adopt an aggressive approach to embed more secret data into a cover image since the GAP setting does not affect the stego-image quality significantly. In addition, we can increase the total embedding capacity of a cover image by increasing the embedding capacity per pixel for the three types of image blocks, defined by our GAP algorithm.

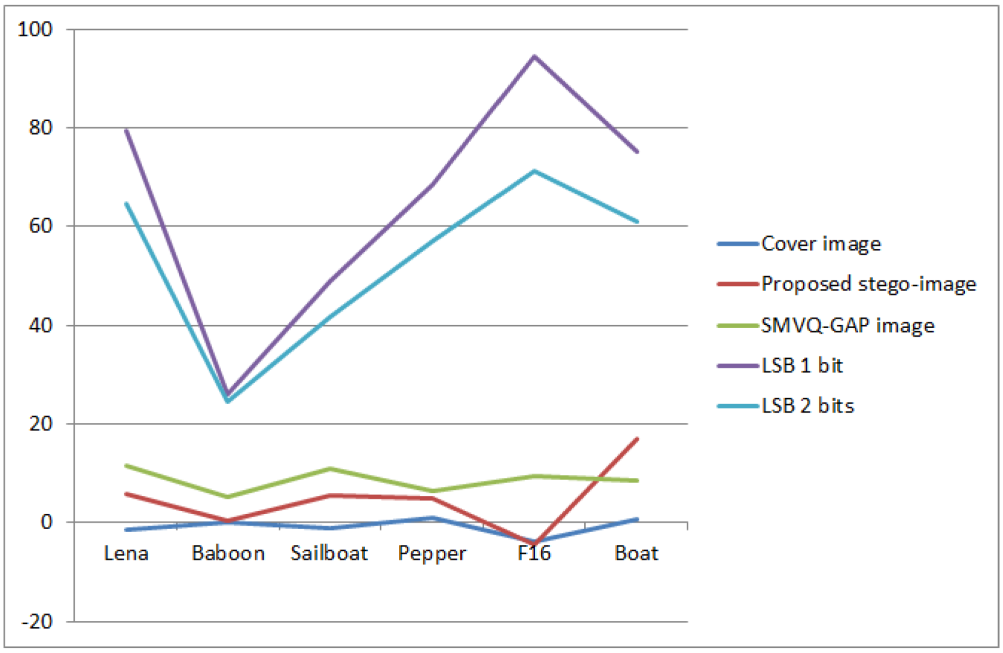

Figure 4. The AUMP LSB Detection values of the six test cover images, the corresponding SMVQ-GAP images, the corresponding stego-images of our proposed method, the corresponding 1-bit-LSB stego-images, and the corresponding 2-bits-LSB stego-images.

From the security perspective, we have tested our stego-images by utilizing AUMP LSB detector [20] which has better detection ability than Chi-square[19]. An adaptive Asymptotically Uniformly Most Powerful (AUMP) test is a statistical hypothesis test designed to decide if a natural image contains hidden secret data. This test maximizes the detection probability of hidden bits, independently of the image parameters and whatever the hiding rate.

In AUMP LSB detection policy, the difference of AUMP values between cover image and stego-image indicates whether the secret data is detected. In Figure 4, we can observe two advantages of our proposed method. First, the AUMP detection values of our method are very close to the AUMP values of the cover images when compared with the LSB stego-images. Second, the SMVQ-GAP images (without any hidden secret data) have larger AUMP values than our stego-images. This implies that the AUMP test cannot tell whether an image is either compressed by the SMVQ-GAP or encoded by the proposed method, or simply a nature image such as Baboon and F16.

5. **Conclusion.** In this paper, we proposed a steganographic method based on SMVQ and GAP. The proposed method can embed huge amount of secret data, and yet maintain good visual quality for the stego-images. The key idea of the proposed method is to utilize the first to the third pixels of an image block, depending on the state codebook size, to embed a GAP-adjusted SMVQ index. The index is used to calculate and to recover the embedded secret bits in the remaining pixels of the block. The experimental results show that the proposed method has significantly larger embedding capacity for secret data and better image quality than a prior method proposed by Chen and Lin [13] in 2010.

Acknowledgment. This work was partially supported by the National Science Council, Taiwan, under Grant No. 101-2218-E-008-003-.

REFERENCES

[1] H. Xu, J. Wang, and H. J. Kim, Near-optimal solution to pair-wise LSB matching via an immune programming strategy, *Journal of Information Sciences*, vol. 180, no. 8 pp. 1201-1217, 2010.

[2] F. Pang, X. Li, and B. Yang, Adaptive reversible data hiding scheme based on integer transform, *Journal Signal Processing*, vol. 92, no. 1, pp. 54-62, 2012.

[3] C. C. Lin, W. L. Tai, and C. C. Chang, Multilevel reversible data hiding based on histogram modification of difference images, *Journal of Pattern Recognition*, vol. 41, no. 12, pp. 3582-3591, 2008.

[4] X. Liao, Q. Y. Wen, and J. Zhang, A steganographic method for digital images with four-pixel differencing and modified LSB substitution, *Journal of Visual Communication and Image Representation*, vol. 22, no. 1, pp. 1-8, 2007.

[5] D. C. Lou, and C. H. Hu, LSB steganographic method based on reversible histogram transformation function for resisting statistical steganalysis, *Journal of Information Sciences*, vol. 188, no. 1, pp. 346-358, 2012.

[6] K. L. Chung, C. H. Shen, and L. C. Chang, A novel SVD- and VQ-based image hiding scheme, *Journal of Pattern Recognition Letters*, vol. 22, no. 9, pp. 1051-1058, 2001.

[7] C. C. Chang, C. C. Lin, C. S. Tseng, and W. L. Tai, Reversible hiding in DCT-based compressed images, *Journal of Information Sciences*, vol. 177, no. 13, pp. 2768-2786, 2007.

[8] H. Noda, M. Niimi, and E. Kawaguchi, High-performance JPEG steganography using quantization index modulation in DCT domain, *Journal of Pattern Recognition Letters*, vol. 27, no. 5, pp. 455-461, 2006.

[9] C. C. Chang, T. S. Nguyen, and C. C. Lin, A reversible data hiding scheme for VQ indices using locally adaptive coding, *Journal of Visual Communication and Image Representation*, vol. 22, no. 7, pp. 664-672, 2011.

[10] 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.

[11] C. C. Chang, W. L. Tai, and C. C. Lin, A reversible data hiding scheme based on side match vector quantization, *IEEE Trans. Circuits and Systems for Video Technology*, vol. 16, no. 10, pp. 1301-1308, 2006.

[12] C. H. Yang, W. J. Wang, C. T. Huang, and S. J. Wang, Reversible steganography based on side match and hit pattern for VQ-compressed images, *Journal of Information Sciences*, vol. 181, no. 11, pp. 2218-2230, 2011.

[13] L. S. T. Chen, and J. C. Lin, Steganography scheme based on side match vector quantization, *Journal of Optical Engineering*, vol. 49, no. 3, pp. 037008-037008-7, 2010.

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

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

[16] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, Techniques for data hiding, *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996.

[17] M. Fallahpour, D. Megias, and M. Ghanbari, Subjectively adapted high capacity lossless image data hiding based on prediction errors, *Journal of Multimedia Tools and Applications*, vol. 52, no. 2-3, pp. 513-527, 2011.

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

[19] S. Guillermito, *Chi-square steganography test program*, 2004, available at http://www.guillermito2.net/stegano/tools/index.html

[20] L. Fillatre, Adaptive steganalysis of least significant bit replacement in grayscale natural images, *IEEE Trans. Signal Processing*, vol. 60, no. 2, pp. 556-569, 2012.