Hiding Information in a Reordered Codebook Using Pairwise Adjustments in Codewords

Chin-Chen Chang[1], Jui-Chuan Liu[1*], Ching-Chun Chang[2], and Yijie Lin[1]

[1]Department of Information Engineering and Computer Science, Feng Chia University, Taichung, 407, Taiwan
[2]Information and Communication Security Research Center, Feng Chia University, Taichung, 407, Taiwan
E-mail: alan3c@gmail.com, p1200318@o365.fcu.edu.tw, ccc@fcu.edu.tw, yjlin.phd@gmail.com

Abstract:
The fundamental data structure of our proposed scheme is a pixel pair. The pixel values are adjusted through simple yet well-designed equations depending on different conditions. The computation method provides an easy recovery of the original values for the pixel pair as well. For some applications, it's important to extract the hiding data as well as to recover the original media. In order to apply to a wider range of applications, more and more reversible data hiding (RDH) schemes are introduced or enhanced in the recent decades. Therefore, the two goals of the novel scheme proposed are to increase embedding capacity and to recover original media easily.

For vector quantization (VQ) compression, it's a well-known and efficient method to compress images. A well-trained codebook is crucial to maintain the visual quality of recovered images as good as possible. An index table is generated based on the codebook to full-fill the purpose of compression. Most of studies utilize index tables as the embedding carriers for VQ or VQ-variant compression methods. There are a few state-of-the-art schemes make use of codebooks to embed secret and our proposed scheme extends the study on the codebook embedding. The codewords in a codebook are divided into pixel pairs and secret bits are embedded by adjusting the pixel values of these split pairs to form a stego codebook. During extraction, secret bits are extracted first out from the stego codebook and a recovered codebook is obtained before recovering VQ compressed images.

Keywords: Reversible data hiding, vector quantization, codebook, pixel pairwise adjustment.

1 Introduction

Technologies are advanced faster than ever with artificial intelligence (AI) coupled. Huge amount of data flows back and forth through numerous channels no matter if they are public or private. Cloud servers become essential storages for various types of users with different kinds of purposes. Information security and privacy become more and more substantial in the recent decades. In order to keep confidential data and sensitive information in secure state during the data transmission, data hiding (DH) [1-4] is one of the technologies aiming to assist in protecting privacy. Many scholars continue to spend incredible effort in DH on either finding solutions to newly appear problems and issues or looking for improvement on performance or embedding capacity.

DH further can be divided into reversible and irreversible. Reversible data hiding (RDH) [5-10]

Fifth International Conference on Computer Vision and Computational Intelligence (CVCI 2024), edited by Chin-Chen Chang, Chutisant Kerdvibulvech , Proc. of SPIE Vol. 13169, 131690C · © 2024 SPIE · 0277-786X · doi: 10.1117/12.3023916

schemes later gained the popularity because they are not only keeping the visual quality of stego images but also assuring to recover the original cover images after data extraction. Barton patented an RDH method back in 1997 [11] which was based on spatial domains and embedded information into media directly. Demand of effective RDH schemes [12, 13] raised because of the fast development of cloud computing and IoT. Traditionally, data are hidden in cover images directly by data hiders. Later on, data hiding carriers can be encrypted media when a data hider embedded secret information. RDH in encrypted image schemes can offer securer capability and gain even more attraction. Exploring new data hiding carriers is also one of our study goals other than enhancing state-of-the-art schemes and finding new data embedding methods.

Vector quantization (VQ) compress [14] is an important and abiding technique for its high compression rate and popularity in various applications. A well-trained codebook is essential for VQ compression and there are a few studies using codebook as the embedding media. The VQ indices are classified into different categories according to how data are embedded: the best matching [15], codeword-order-cycle permutation [16], clustering [17–21], index set reconstruction [22-25] in the earlier research. The category of index set reconstruction operated the order of codeword indices in sorted codebooks to embed secrets. Our proposed scheme further improves embedding capacity in the sorted codebooks without disturbing the visual quality of images by manipulating pixel pairs. Codewords in a codebook are split into pixel pairs whose pixel values are adjusted depending on the bit value of 0 or 1 of a secret binary bit stream. Because there are only simple computations involved, both embedding and extracting processes are easy. The main contributions of this study are listed below:

- Explore a less-common data hiding carrier and further increase its embedding capacity.
- Apply simple mathematic computation during data embedding and data extraction.
- Stego codewords are fully reversible.

Related Works is in Section 2 to brief some key points of methods which are either allied with the proposed scheme or involved in our experiments. Section 3 explainis a fundamental idea of our scheme and Section 4 describes how the fundamental idea is applied to codebooks in the proposed scheme. Analyze experiment results in Section 5 and Section 6 concludes our paper.

2 Related Works

Our proposed data hiding carrier is a well-trained codebook to limit the loss of visual performance during compression. The steps of a VQ codebook generation is described in Subsection 2.1 and an earlier data hiding scheme based on VQ codeword index reordering is roughly explained in Subsection 2.2.

2.1 VQ codebook Generation

The vector quantization is a compression method which uses a limited sized of vectors to represent a blocked cover image to achieve the purpose of reducing data size and facilitating space for storage. Therefore, picking represented vectors as the codewords in a codebook is essential to constraint the visual quality loss of the original image.
Even there are public codebooks can be used for compression, Algorithm 1 describes how a codebook is trained when using a set of grayscale images. A flow diagram is displayed in Figure 1 as well.

Algorithm 1 – VQ codebook generation [14]	
Input	A collection of greyscale sample images CI.
Output	A VQ codebook CB.
Step 1	Set $cb = \{\}$
Step 2	FOR each image I in CI Divide the image I into $n \times n$ blocks and add blocks to cb END
Step 3	Random select blocks as centroids depending on the size of CB desired.
Step 4	Cluster a block by calculating Euclidean distance between the block and the centroids
Step 5	Recapture the centroid for each group.
Step 6	Repeat Step 4 and Step 5 until centroids reach a stable state.
Step 7	Collect centroids into CB.
Step 8	Export CB.

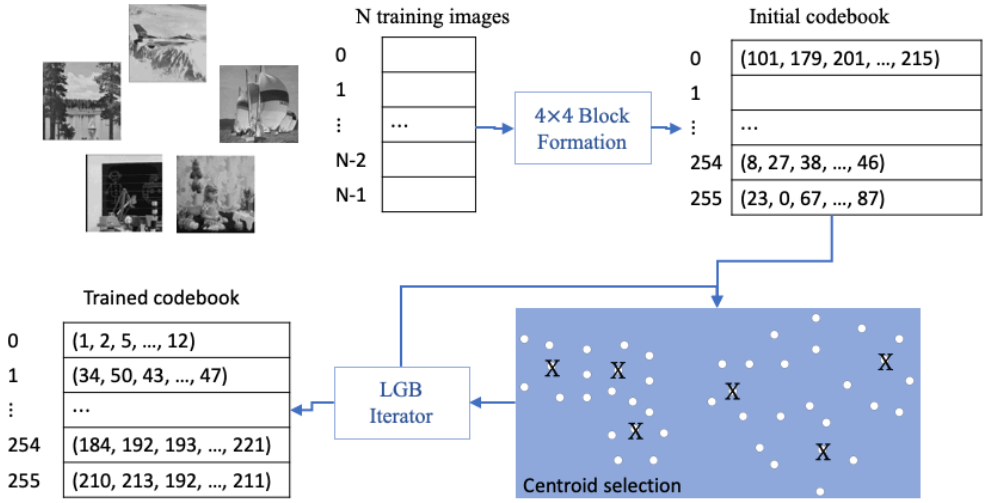

Figure 1: Flow of codebook training.

2.2 Codeword Index Reordering Scheme

There are fewer studies focus on using codebook as a data embedding carrier, but it's still an interesting area to explore. After getting a fundamental codebook CB, the scheme sorted the codebook and embedded secret data by using the codeword index reordering technique [24]. Figure 2 illustrates the flow of the scheme.

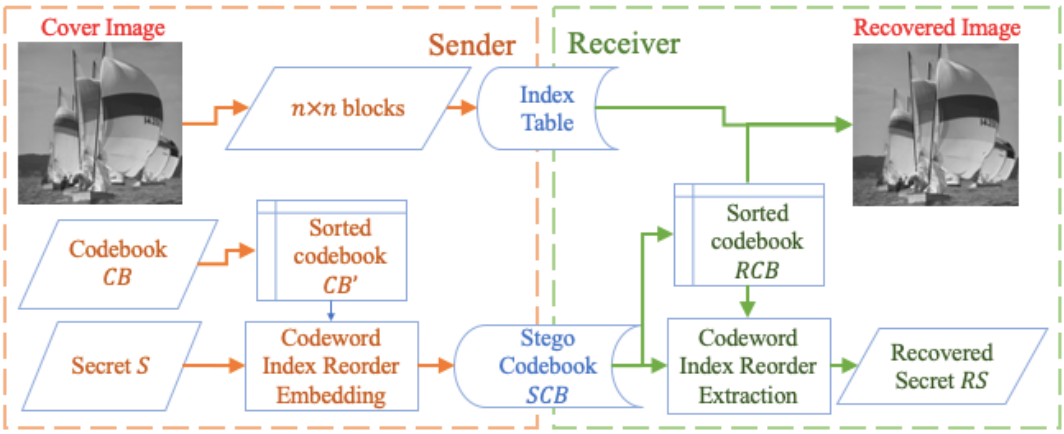

Figure 2: Flow of codeword index reordering scheme.

Algorithm 2 walks through the steps of embedding data by using the indices of codewords in a sorted codebook.

Algorithm 2 – Codeword Index Reorder Embedding	
Input	A VQ codebook CB, a projecting line L, secret data $S = \{s_0, s_1, \cdots, s_k\}, s_k = \{0, 1\}$.
Output	A stego VQ codebook SCB.
Step 1	Project codewords in codebook CB on to the line L to get a set of projected values.
Step 2	Sort codebook CB by the projected values to gain CB'.
Step 3	Set $SCB = \{\}$
Step 4	WHILE $CB' \neq \{\}$ Calculate the embeddable bit count $$bc = \lfloor log2(length(CB')) \rfloor. \quad (1)$$ Convert $e = s_0 + s_1 + \cdots + s_{bc-1}$ to decimal de. Move the codeword of $CB'[de]$ out from CB' and add to SCB. END
Step 5	Export SCB.

To understand the concept of embedding data using index reordering, Figure 3 uses a simple stego set to explain the steps of an embedding process. Let sorted number set $SN = \{0, 1, 2, 3, 4, 5, 6, 7\}$ and secret message $S = (1001011001011)_2$, the number of embeddable bits nb for 8 numbers is 3 based on Eq. (1). The first 3 bits of secret is $(100)_2$ which equals to $(4)_{10}$. The decimal number 4 is used as an index to move the content number 4 from the sorted set SN to a stego set SN'. After the first move, there are seven numbers left in SN, one number in SN', and the next embeddable bit count $bc = \lfloor log_2 7 \rfloor = 2$. Therefore, the next 2 secret bits are $(10)_2$ which means that the next content to be moved is the number indexed by $(2)_{10}$. The number 2 pointed by index 2 is moved from the sorted set SN to the stego set SN'. Repeat the steps of calculating bc, fetch index by convert secret bits to decimal and move content indexed by the fetched index until there is no secret bit left and move the rest of contents into stego set based on their sorted orders.

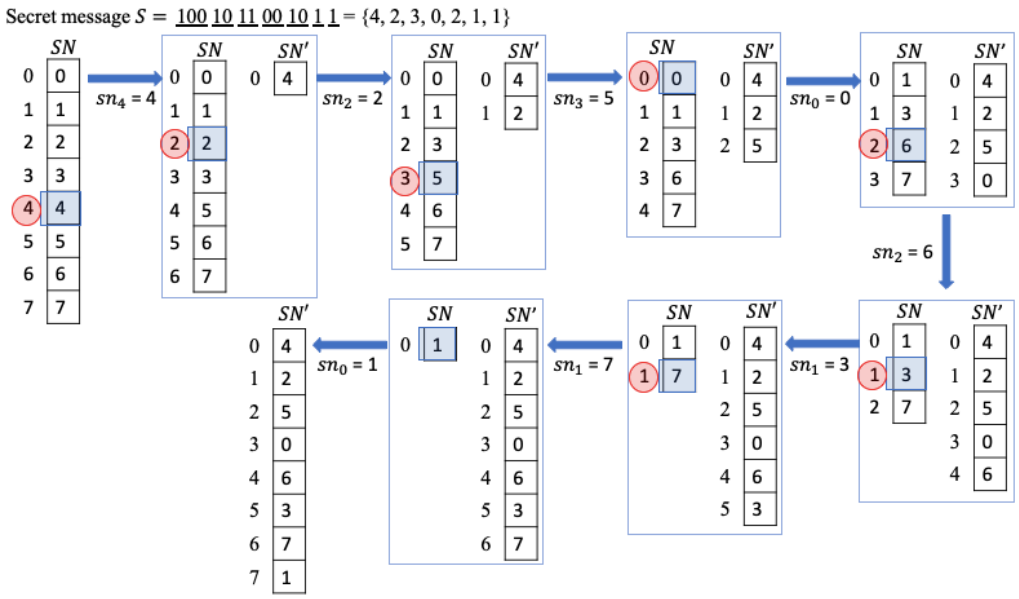

Figure 3: Embedding a secret S into a sorted set SN to generate the stego set SN'.

Figure 4 demonstrates a sorted codebook CB' employs the same index reordering technique to create a stego codebook SCB. Consider an original CB contains eight codewords and each codeword consists of two elements. Let $CB = \{cw_0, cw_1, \cdots, cw_7\} = \{(4,2), (3,7), \cdots, (5,4)\}$. Let A be the set of their corresponding projected values to a line L. $A = \{\alpha_0, \alpha_1, \cdots, \alpha_7\} = \{14, 27, \cdots, 22\}$ is the projected value set of the codebook CB. The sorted codebook CB' is obtained by sorting the projected values of the codewords. The first embeddable bit count $bc = \lfloor log_2 8 \rfloor = 3$ and thus the embedded decimal value for the first 3 bits is 4. The codeword cw_4' is selected and moved to the stego codebook SCB as the first codeword. Since there are 7 codewords left in CB', the next embeddable bit count $bc = \lfloor log_2 7 \rfloor = 2$, the index value desired is $(10)_2 = 2$ and cw_2' is moved to SCB as the second codeword. The process continues until there is no more embedding bit and the stego codebook is formed.

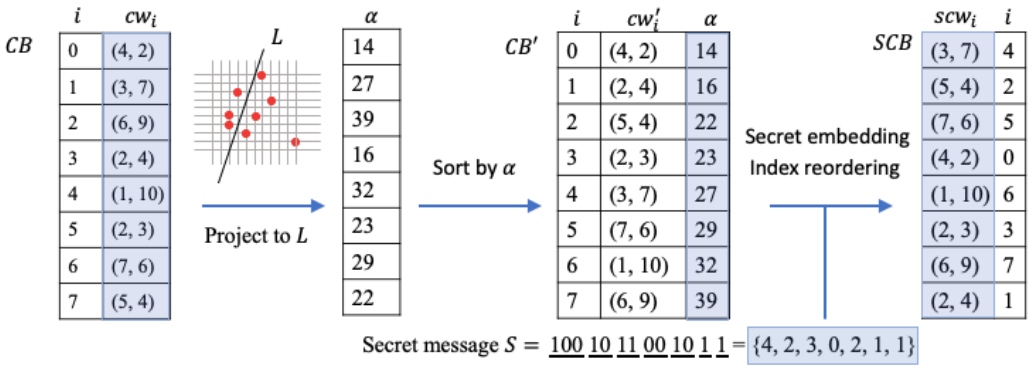

Figure 4: Embedding a secret S into a sorted codebook CB to generate the stego codebook SCB.

Algorithm 3 is the process of extracting secret message and recovering the sorted codebook.

Algorithm 3 – Codeword Index Reorder Extraction	
Input	A stego VQ codebook SCB and the projecting line L.
Output	A sorted VQ codebook RCB, the extracted secret data RS.
Step 1	Project codewords in the codebook SCB on to the line L to get a set of projected values.
Step 2	Sort the codebook SCB by the projected values to gain RCB.
Step 3	Set $RS = \{\}$
Step 4	Copy RCB to RCB'.
Step 5	WHILE $RCB' \neq \{\}$ Get the index of scw_i which matches the codeword in RCB'. Convert the index i to binary $rs_0 + rs_1 + \cdots + rs_{bn}$ and append to RS. Remove the codeword $scw_i \; from \; RCB'$. END
Step 6	Export RCB and RS.

Figure 5 illustrates the recovery process of the above stego codebook. After receiving the stego codebook SCB, project the codewords to the same line L and collect the projected value set $\{s\alpha_0, s\alpha_1, \cdots, s\alpha_7\} = \{27, 22, \cdots, 16\}$. The recovery codebook RCB can be gained after sorting the corresponding projected values. By matching the codewords in the newly sorted codebook RCB, we can simulate the movements of codewords and extract the secret message by binarizing and concatenating the matched index values.

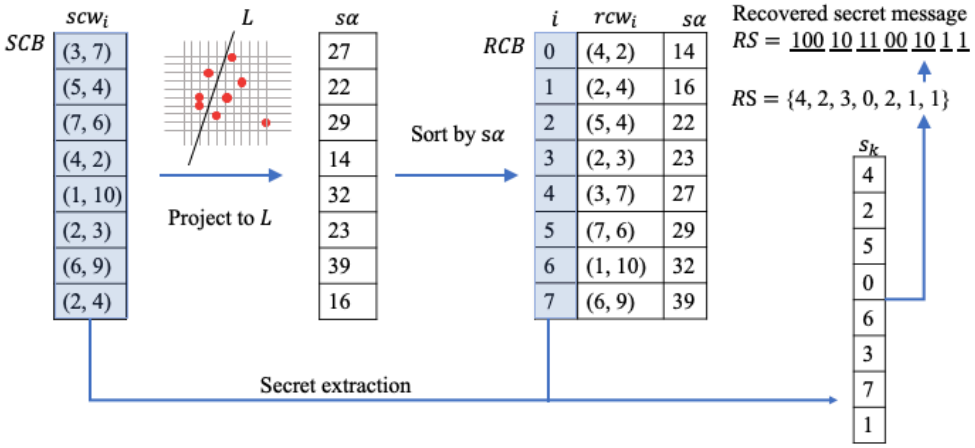

Figure 5: Extracting secret RS from the stego codebook SCB and the recovered codebook RCB.

3 Preliminary Scheme

In the proposed scheme, a fundamental pixel-pair data embedding and extraction are introduced first in Subsection 3.1 and Subsection 3.2 to understand the core idea of how pixels are modified and recovered. Examples are given in Subsection 3.3 to clarify the basic concept. The basic function is then derived and employed to a sorted codebook after learning the principles of pixel-pair manipulations. Using a codebook as the secret carrier is explained in Section 4.

3.1 Reversible Pairwise Data Embedding

There are many state-of-the-art schemes manipulate pixel pairs and one of them is difference expansion (DE) [26] which requires simple computation effort and is commonly chosen for data hiding. Unfortunately, it requires to save extra information along the secret data to overcome underflow and overflow problems. In order not to keep the embedding capacity high without generating subordinate data, our novel scheme presents a method which is simple but elegant.

When there is a cover pixel-pair $pp = (a, b)$, where $0 \leq a, b \leq 255$ and a secret bit $s_k \in \{0, 1\}$,

$$pp' = (a', b') = \begin{cases} (a, b), & if\ (a = 0 \vee b = 0) \\ (a, b), & if\ (a + b) > 255 \\ (a, b), & if\ (a + b + a) > 255 \\ (a, b), & if\ (a + b + b) > 255 \\ (a + b, b), & if\ s_k = 0 \\ (a, a + b), & if\ s_k = 1 \end{cases}, \tag{2}$$

where pp' is the stego pixel-pair. In order to determine if a $(a' + b') > 255$ situation is caused by not embeddable of the original pixel pair or the pair turned into non-embeddable after embedding, two conditions $(a + b + a) \leq 255$ and $(a + b + b) \leq 255$ are derived to qualify the embeddability. Algorithm 4 clarifies data embedding process in steps.

Algorithm 4 – Pairwise Data Embedding	
Input	A cover pixel pair pp, a secret bit s_k.
Output	Stego pixel pair pp'.
Step 1	Set $pp = (a, b)$
Step 2	$a' = a$ and $b' = b$

Step 3	IF both a and b are not 0 THEN
	IF $(a + b) \leq 255$ THEN
	IF (s_k is 0) THEN
	$a' = a + b$
	ELSE
	$b' = a + b$
	ENDIF
	IF $(a' + b') > 255$ THEN
	Mark the pixel pair is embeddable
	ENDIF
	ELSE
	Mark the pixel pair is non-embeddable
	ENDIF
	ENDIF

Step 3	Set $pp' = (a', b')$
Step 4	Export pp'.

Markers are collected and are used to indicate whether pixel pairs were after embedding or not.

The indicator markers can be compacted by any compression algorithms, such as commonly used extended run length encoding (ERLE).

3.2 Reversible Pairwise Data Extraction

The secret extraction and recovery process carry the simplicity from embedding process. The original pixel pair can be recovered fully after secret bit is extracted. Algorithm 5 elucidates the steps of data extraction and image recovery.

Algorithm 5 – Pairwise Data Extraction	
Input	The stego pixel pair pp'.
Output	Recovered pixel pair rpp, recovered secret bit s'_k.
Step 1	Set $pp' = (a', b')$
Step 2	Set $ra = a', rb = b'$

Step 2	IF both a and b are not 0 THEN
	IF $(a' + b') \leq 255$ OR pair is embeddable THEN
	IF $(a' > b')$ THEN
	Set $ra = a' - b'$ and $rb = b'$
	$s'_k = 0$
	ELSE
	Set $ra = a'$ and $rb = b'$ - a'
	$s'_k = 1$
	ENDIF
	ENDIF
	ENDIF

Step 3	Set $rpp = (ra, rb)$
Step 4	Export rpp and s'_k.

3.3 Examples of Reversible Pairwise Embedding and Extraction

Figure 6 illustrates the value adjustments during the pixel pair embedding and extracting process.

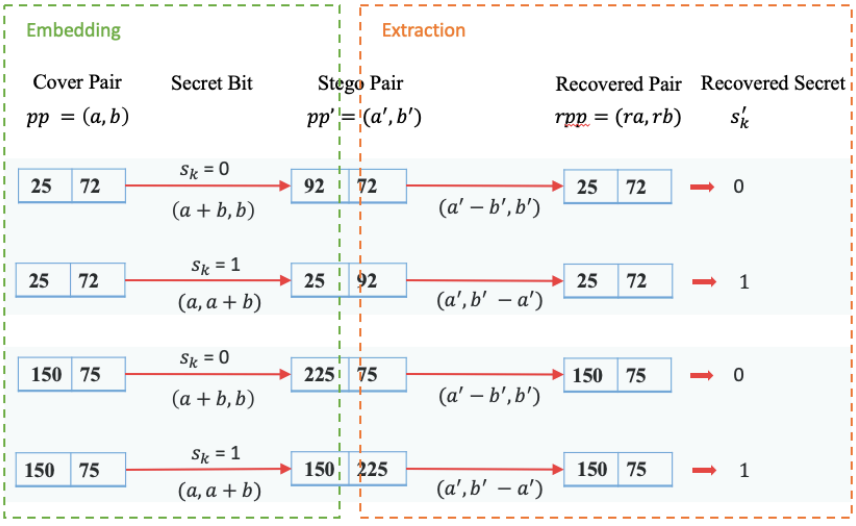

Figure 6: Examples of proposed fundamental process.

4 Proposed Scheme on VQ Codebooks

The proposed novel scheme not only applies the codeword index reordering method, but also derives the usage of pairwise data embedding on codewords in the codeword index reordered codebooks. Figure 7 demonstrates the whole flow of the proposed scheme. Codeword pairwise data embedding and data extraction are described in detail in Subsection 4.1 and Subsection 4.2. Subsection 4.3 is given out the algorithms of embedding and extraction of a codebook.

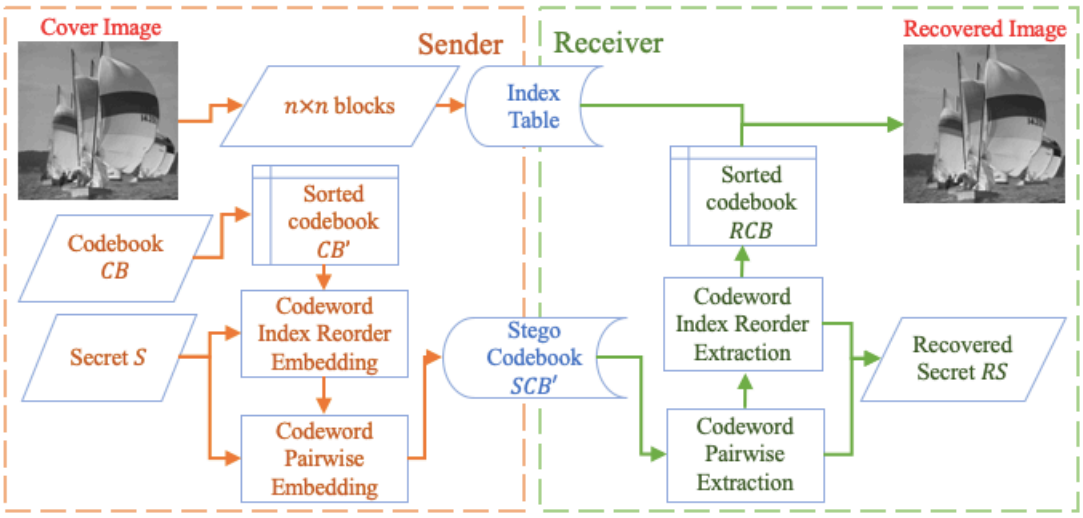

Figure 7: Flow of the proposed scheme.

4.1 Codeword Pairwise Embedding

There is a codeword index reordered codebook SCB generated from Algorithm 2. Assuming a codeword which contains of n pixels, the scheme pairs every two pixels into a pair. The proposed fundamental scheme embeds $n / 2$ bits in each codeword. Let $SCB = \{scw_0, scw_1, \cdots, scw_{m-1}\}$, where m is the size of the codebook. If a secret message $S = \{s_0, s_1, \cdots, s_k, \cdots, s_{l-1}\}$, where $s_k \in \{0, 1\}$, a stego codebook SCB' is formed after the secret embedding into the codeword index reordered codebook SCB using Algorithm 6.

Because of the characters of trained codebooks, there is no chance to embed data when both pixel values are over 128. The proposed scheme adds a preprocess to adjust pixel values for each pixel pair in order to increase the successful rate of the pairwise embedding. The goal of the preprocess is to decrease the pixel values and thus increase the chances of embedding secret using Eq. (2). When there is a cover pixel-pair $pp = (a, b)$, where $0 < a, b \leq 255$ and $a \neq b$.

$$(a_1, b_1) = (\max(a, b), \min(a, b)). \tag{3}$$

$$(a_2, b_2) = (\lfloor (a_1 + b_1)/2 \rfloor, a_1 - b_1). \tag{4}$$

To make sure we can embed and extract secret bit successfully by using these preprocessed pixel values, there are a few restricted conditions of a and b are derived,

1) The new pixel range $\mathrm{R}(a_2, b_2) = [a_2, b_2]$ is smaller than the range of $\mathrm{R}(a_1, b_1) = [a_1, b_1]$; otherwise, the embedding possibility can be decreased.

2) The new pixel range values a_2 and b_2 cannot be the same.

Therefore, the preprocess of adjusting pixel values to increase the embeddable pixel pairs only needs to meet the following conditions:

$$ppa = (a_{ppa}, b_{ppa}) = \begin{cases} (a, b), & if\ \mathrm{R}(a_2, b_2) > \mathrm{R}(a_1, b_1) \vee a_2 == b_2 \\ (a_2, b_2), & if\ (a > b), \\ (b_2, a_2), & if\ (a < b) \end{cases} \quad (5)$$

where ppa is the adjusted stego pixel pair. When $\mathrm{R}(a_2, b_2) > \mathrm{R}(a_1, b_1)$ or $a_2 == b_2$, there is range adjustment. In order to recover the original pixel, the order of a_2, b_2 is based on the original location of a and b. A list of shrinking indicators is collected for range recovery. The indicator list can be compressed by an extended run length encoding (ERLE) as well.

Algorithm 6 – Codeword Pairwise Embedding	
Input	A codeword index reordered codebook SCB, secret message S.
Output	Stego codebook SCB'.
Step 1	Initialize $SCB' = \{\}$
Step 2	FOR each codeword scw_i in CB Divide scw_i into $n\ /\ 2$ pixel pairs pps_i $scw_i' = \{\}, k = 0$ FOR each pixel pair pp in pps_i Get ppa by adjusting pp using Eqs. (3), (4) and (5) Set s_k to bit k in S Get stego pixel pair pp' From Algorithm 4 using ppa and s_k Increase k Append pp' to scw_i' END Add scw_i' to SCB' END
Step 3	Export SCB'.

4.2 Codeword Pairwise Data Extraction

After a receiver receives the stego codebook SCB', the stego codewords inside are split into stego pixel pairs. Each stego pixel pair is recovered by using Algorithm 5 and a postprocess afterwards. The postprocess utilizes Eqs. (6), (7) and (8) to gain recovered pixel values which was adjusted by shrinking ranges of pixel pairs. Eq. (7) for pixel value recovery is derived directly from Eq. (4). If the second value of a stego pixel pair is an even number, the recovered pixel pair values $(a_2', b_2') = ((2a_1' + b_1')/2,\ b_1' - (2a_1' + b_1')/2)$. If the second value of a stego pairwise is an odd number, the recovered pixel pair values $(a_2', b_2') = ((2a_1' + b_1' + 1)/2,\ b_1' - (2a_1' + b_1' + 1)/2)$.

$$(a_1', b_1') = (\max(a', b'), \min(a', b')), \quad (6)$$

$$(a_2', b_2') = \begin{cases} ((2a_1' + b_1')/2, b_1' - (2a_1' + b_1')/2), \\ if\ b_1'\ is\ an\ even\ number; \\ ((2a_1' + b_1' + 1)/2, b_1' - (2a_1' + b_1' + 1)/2), otherwise; \end{cases} \quad (7)$$

$$rpp' = (a_{rpp'}, b_{rpp'}) = \begin{cases} (a_2', b_2'), & if\ (a' > b'); \\ (b_2', a_2'), & if\ (a' < b'); \end{cases} \quad (8)$$

where rpp' is the recovered pixel-pair and ra, rb are the recovered pixel values.

Algorithm 7 – Codeword Pairwise Extraction	
Input	A stego codebook SCB'
Output	Recovered codebook $RSCB$, recovered secret message RSS.
Step 1	Initialize $RSCB = \{\}$ and $RSS = \{\}$
Step 2	FOR each codeword scw_i' in SCB', Divide scw_i' into $n\ /\ 2$ pixel pairs pps_i' $rcw_i = \{\}$, FOR each embeddable pixel pair pp' in pps_i' Get recovered pixel pair rpp and recovered secret bit s_k' from Algorithm 5 using pp' Get rpp' using Eqs. (6), (7) and (8) Append rpp' to rcw_i Append s_k' to S' END Add rcw_i to $RSCB$ END
Step 5	Export $RSCB$ and RSS

4.3 Codebook Embedding and Extraction

In order to complete the whole flow, Algorithm 8 and Algorithm 9 step through the embedding process and the extraction process of a codebook.

Algorithm 8 – Proposed Codebook Embedding	
Input	A well-trained codebook CB, a projecting line L, secret message S.
Output	Stego codebook SCB'.
Step 1	Get stego codebook SCB from Algorithm 2 using CB, L, and S
Step 2	Get stego codebook SCB' from Algorithm 6 using SCB and S
Step 3	Export SCB'.

Algorithm 9 – Proposed Codebook Extraction	
Input	A stego codebook SCB', the projecting line L
Output	Recovered codebook RCB, recovered secret message RS.
Step 1	Get recovered codebook $RSCB$ and recovered secret RSS from Algorithm 7 using SCB'
Step 2	Get recovered codebook RCB and recovered secret RS from Algorithm 3 using $RSCB$ and L
Step 3	Append RSS to RS
Step 4	Export RCB and RS

5 Experimental Results and Analysis

Since this novel scheme uses codebooks as data hiding carriers, well-trained VQ codebooks plays a critical role in the scheme. Well-represented codebooks used for VQ compression are desired to conduct meaningful experiments. Our VQ training image set is showed in Figure 8.

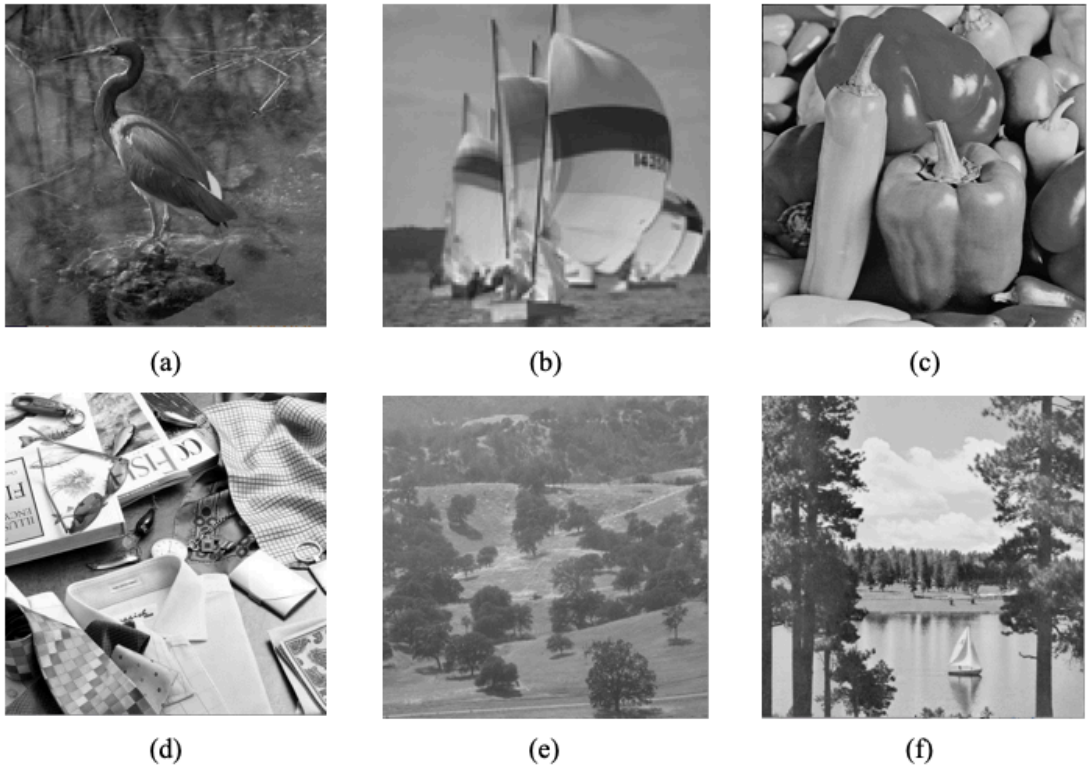

Figure 8: VQ codebook training images sized 512×512 pixels: (a) “Egretta”, (b) “Sail”, (c) “Peppers”, (d) “Fashion”, (e) “Woodland”, and (f) “Lake”.

To effectively evaluate the efficiency of the proposed scheme in a compression domain, we recorded the numbers of embeddable bits in various sizes of codebooks. Embeddable rates are calculated by

$$ER = C/np \text{ (bpp)}, \tag{9}$$

where C represents the total embedding capacity and np is the number of pixels in a codebook. We had completed two experiments: one was embedding directly to pixel pairs of codewords in various sizes of codebook and the other one was adjusting ranges of pixel-pairs of the codewords as the preprocess. The results are displayed in Table 1 and Table 2. From the two tables, we could find that the embedding capacity and rates were much higher in the Table 2 which meant that our shrinkage algorism is significantly efficient to meet our purpose.

Table 1. Embedding capacity without pixel-pair range adjustment

	Embedding capacity (bits)		**Embeddable rate (bpp)**	
Size of codebook	CW Index Reordering	Proposed	CW Index Reordering	Proposed
64	264	321	0.2578	0.3134
128	649	744	0.3169	0.3632
256	1546	1741	0.3774	0.4250
512	3595	3819	0.4388	0.4662
1024	8204	8738	0.5007	0.5333

Table 2. Embedding capacity with pixel-pair range adjustment

	Embedding capacity (bits)		**Embeddable rate (bpp)**	
Size of codebook	CW Index Reordering	Proposed	CW Index Reordering	Proposed
64	264	596	0.2578	0.5820
128	649	1433	0.3169	0.6997
256	1546	3128	0.3774	0.7637
512	3595	6876	0.4388	0.8394
1024	8204	14809	0.5007	0.9039

When overserving the capacity experiment, we can increase the capacity by substantial amount. Another observation is that the larger the codebook is and the higher the embeddable rate is. The two key points of this study is that we like to be able to make full usage of this rare used carrier and maintain its reversibility. According to our experiments, this novel scheme does boost up codebooks' embedding capacity and avoid degrading the visual performance of VQ compress images by recovering codebooks completely.

6 Conclusions

Our experiment results show the improvement on embedding capacity of codebook over the existing codeword index reordering method by making use of the codewords as well. Since we combine the index reordering scheme and newly proposed scheme, the codebook embedding capability is significantly increased. Using simple mathematics computations to embed extra secret message into codebooks is an interesting area to explore. Because of codebooks' sizes can be small, it's very challenge to discover simple computations to enlarge the usage of the new carrier.

It will be interesting to explore more on applying our scheme to an encrypted domain in the future. Even the embedding capacity is small when using codebooks as carriers, it's worth of exploring when applications require some extra space for data hiding.

References

1. Y. C. Lin, C. C. Wang. 1999. Digital images watermarking by vector quantization. Proc Natl Comput Symp 3:76–87

2. W. J. Wang, C. T. Huang, C. M. Liu, P. C. Su, S. J. Wang. 2013. Data embedding for vector quantization image processing on the basis of adjoining state-codebook mapping. Inf Sci 246:69–82
3. Z. B. Pan, X. X. Ma, X. M. Deng, S. Hu. 2013. Low bit-rate information hiding method based on search-order-coding technique. J Syst Softw 86(11):2863–2869
4. C. C. Lin, X. L. Liu, S. M. Yuan. 2015. Reversible data hiding for VQ-compressed images based on search-order-coding and state-codebook mapping. Inf Sci 293:314–326
5. B. Xia, A. Wang, C. C. Chang, L. Liu. 2017. Reversible data hiding for VQ indices using hierarchical state codebook mapping. Multimed Tools Appl 77(16):20519–20533
6. J. Tian. 2003. Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 13(8): 890-896
7. Z. Ni, Y. Q. Shi, N. Ansari, W. Su. 2006. Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3): 354–362
8. X. T. Wang, C. C. Chang, T. S. Nguyen, and M. C. Li. 2013. Reversible data hiding for high quality images exploiting interpolation and direction order mechanism. Digital Signal Processing 23(2): 569-577
9. C. C. Chang and T. D. Kieu. 2010. A reversible data hiding scheme using complementary embedding strategy. Information Sciences 180(16): 3045-3058
10. C. C. Chang and Y. C. Hu. 1998. A fast LBG codebook training algorithm for vector quantization. IEEE transactions on consumer Electronics 44(4): 1201-1208
11. J. M. Barton. 1997. Method and apparatus for embedding authentication information within digital data. United States Patent, No. 5646997, Patent and Trademark Office, Washington, DC
12. X. Xiong, Y. Chen, M. Fan, and S. Zhong. 2022. Adaptive reversible data hiding algorithm for interpolated images using sorting and coding. Journal of Information Security and Applications 66:103137
13. F. J. Huang, X. C. Qu, H. J. Kim, and J. W. Huang. 2016. Reversible data hiding in JPEG images. IEEE Transactions on Circuits and Systems for Video Technology 26(9):1610–1621
14. Y. Linde, A. Buzo, and R. Gray. 1980. An algorithm for vector quantizer design. IEEE Trans Communication 28(1): 84–95
15. W. C. Du and W. J. Hsu. 2003. Adaptive data hiding based on VQ compressed images. IEE Proc Vis Image Signal Process 150(4):233–238
16. C. C. Chang and W. C. Wu. 2006. Hiding secret data adaptively in vector quantisation index tables. IEE Proc Vis Image Signal Process 153(5):589–597
17. C. C. Chang, W. C. Wu, and Y. C. Hu. 2007. Lossless recovery of a VQ index table with embedded secret data. J. Vis. Commun. Image Represent 18(3):207–216
18. C. C. Lin, S. C. Chen, and N. L. Hsueh. 2009. Adaptive embedding techniques for VQ-compressed images. Inf Sci 179(1–2):140–149
19. C. H. Yang and Y. C. Lin. 2009. Reversible data hiding of a VQ index table based on referred counts. J Vis Commun Image Represent 20(6):399–407
20. C. H. Yang and Y. C. Lin. 2010. Fractal curves to improve the reversible data embedding for VQ-indexes based on locally adaptive coding. J Vis Commun Image Represent 21(4):334–342
21. J. Yuan, H. Zheng, and J. Ni. 2023. Efficient reversible data hiding using two-dimensional pixel clustering. Electronics 12(7):1645
22. C. Qin, C. C. Chang, and Y. C. Chen. 2013. A novel reversible data hiding scheme for VQ-compressed images using index set construction strategy. KSII Trans. Internet Inf. Syst. TIIS 7(8) 2027–2041

23. C. Qin, C. C. Chang, and Y. C. Chen. 2013. Efficient reversible data hiding for VQ-compressed images based on index mapping mechanism. Signal Process 93(9):2687–2695
24. J.C. Liu, C. C. Chang, C. C. Lin, and C. C. Chang. 2023. Hiding Information in a Well-Trained Vector Quantization Codebook. ACM International Conference on Signal Processing and Machine Learning (SPML) https://doi.org/10.1145/3614008.3614052
25. X. Wang, J.C. Liu, and C. C. Chang. 2023. A novel reversible data hiding scheme for VQ codebooks. Security Privacy https://dor.org/10.1002/spy2.315
26. J. Tian. 2003. Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 13(8):890-89