Journal of Visual Communication and Image Representation 52 (2018) 86–91

Contents lists available at ScienceDirect

Journal of Visual Communication and Image Representation

journal homepage: www.elsevier.com/locate/jvci

Lossless data hiding in JPEG bitstream using alternative embedding☆

Yingqiang Qiu[a], Han He[a], Zhenxing Qian[b,*], Sheng Li[b], Xinpeng Zhang[b]

[a] College of Information Science & Engineering, Huaqiao University, Xiamen 361021, China
[b] Shanghai Institute of Intelligent Electronics & Systems, School of Computer Science, Fudan University, 200433, China

ARTICLE INFO

Keywords:
Information hiding
Lossless data hiding
Reversible data hiding

ABSTRACT

This paper proposes a lossless data hiding method for JPEG images using adaptive embedding. By constructing an optimal mapping between the used and unused Huffman codes in each category, we take full use of the combination of mapping to achieve a high embedding rate. In order to improve the payload, we further use a code reordering based embedding algorithm. Both algorithms are alternatively used during data hiding. After modifying the Huffman Table defined in JPEG header and substituting the codes in entropy-encoded segments, additional messages are embedded into the JPEG bitstream. The proposed method is lossless to the image, *i.e.*, the decoded content of a marked JPEG bitstream is identical to the original JPEG image. Meanwhile, the file size can be well preserved after data hiding. Experimental results show that the proposed method has a better performance than state-of-the-art works.

1. Introduction

Information hiding is a technology that embeds secret bits into a cover imperceptibly for various purposes, e.g., covert transmission, copyright protection, multimedia management, etc. Reversible data hiding (RDH) is a kind of information hiding, which embeds additional messages into a cover, esp. an image or a video, and guarantees that the original image/video can be recovered exactly after data extraction. RDH is useful in managing images on cloud storage. A cloud server can embed additional messages, e.g., image labels, timestamps, user information and remarks, into digital images. Since the messages are attached inside the images, a better management can be achieved, and the storage overheads of the server can be saved. Meanwhile, the original content can be losslessly recovered before the user's downloading.

Generally, there are three kinds of RDH methods [1], *i.e.* the compression based RDH [2–5], the histogram modification based RDH [6–9], and the expansion based RDH [10–15]. With these methods, good embedding performance, in terms of visual quality and embedding capacity, can be achieved. However, these methods can only be used in uncompressed images. In recently years, RDH for JPEG has attracted much attention, since this format is widely used all over the world. There are four types of RDH approaches for JPEG images, *i.e.*, RDH by quantizing DCT coefficients [16–19], RDH by modifying Quantization Table [20–22], RDH by modifying Huffman Table [23–26], and RDH in encrypted JPEG bitstreams [27–29]. Unlike robust watermarking, RDH is widely used when perfect image reconstruction and data extraction are emphasized while robustness against malicious attacks is not considered [29].

While most RDH methods cause image distortion during data embedding [2–23], RDH by modifying Huffman Table surprisingly causes no distortion to the image content after data embedding [24–26], which is referred to as the lossless data hiding (LDH). Mobasseri et al. proposed the first LDH for JPEG bitstream by mapping a used *variable-length-code* (VLC) to an unused VLC [23]. As this method may result in a decoding failure, Qian and Zhang proposed a code mapping approach to overcome this drawback [25]. They construct the code mapping by modifying the Huffman Table in JPEG header. The used VLC in JPEG bitstream is then substituted by the mapped unused VLC to represent additional bits. After embedding, the marked JPEG bitstream can still be decoded by popular JPEG decoders, and the file size can be kept unchanged. This embedding algorithm is lossless, since the content of a marked JPEG image is identical to the original. Another LDH was proposed in [26] to make a better use of code mapping, in which a larger payload is achieved. In [24], a bijective mapping algorithm was proposed to construct a Huffman tree, and the JPEG bitstream is synchronically modified to realize LDH.

Although the methods in [23–26] have good capabilities of losslessly hiding data into JPEG bitstreams, the embedding capacity is quite small. In this paper, we propose a new strategy of losslessly hiding data into the JPEG bitstream. By analyzing the relationships between the used and the unused VLCs, we construct an alternative embedding based LDH using code mapping and reordering. After data hiding,

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: zxqian@fudan.edu.cn (Z. Qian).

https://doi.org/10.1016/j.jvcir.2018.02.005
Received 3 September 2017; Received in revised form 21 November 2017; Accepted 9 February 2018
Available online 10 February 2018

content of the image is identical to the original JPEG image. Meanwhile, the embedding payload is larger, and the file size is preserved.

2. Proposed method

2.1. General framework

The proposed LDH framework is depicted in Fig. 1. We first parse a JPEG bitstream and reconstruct 162 Huffman codes for AC coefficients, represented by VLCs. These codes are then classified into 16 categories. For each category, we build a rule to represent additional bits by modifying VLCs, including the code mapping and reordering. With this rule, we embed additional bits into the JPEG bitstream using the alternative embedding. This way, a marked JPEG bitstream is generated.

2.2. VLC classification

For the brevity of description, we only introduce the baseline JPEG for grayscale images. According to the JPEG standard, A JPEG bitstream contains a marker *start-of-image* (SOI), a JPEG header, the entropy-coded segments (ECS), and a marker *end-of-image* (SOI). In the JPEG header, the Quantization Tables and the Defined Huffman Tables (DHT) for DC/AC coefficients are declared. Each ECS represents the compressed bits of an 8×8 block, containing the Huffman codes and the appended bits of *variable-length-integers* (VLI). In JPEG, the DHT includes 162 VLCs for AC coefficients. Each VLC is represented by the code *lengths* and *run/size* defined in DHT. With these data, the Huffman Table can be reconstructed. Actually, many codes are not used in a specific compression.

After parsing a JPEG bitstream, we classify all Huffman codes into two types: the *used*, and the *unused*. We further classify 162 VLCs into 16 categories $\{C_1, C_2, \dots, C_{16}\}$. Each C_i has L_i codes of length i, including p_i *used* VLCs and q_i *unused* VLCs,

$$C_i = \{VLC_{i,1}^{(u)}, VLC_{i,2}^{(u)}, \dots, VLC_{i,p_i}^{(u)}; VLC_{i,1}^{(n)}, VLC_{i,2}^{(n)}, \dots, VLC_{i,q_i}^{(n)}\}$$

where $VLC_{i,1}^{(u)} \sim VLC_{i,p_i}^{(u)}$ stands for the *used* VLCs, $VLC_{i,1}^{(n)} \sim VLC_{i,q_i}^{(n)}$ the *unused*, and $i = 1, 2, \dots, 16$.

2.3. Bit representation

After VLC classification, we build a rule for each category to represent additional bits. We have two alternative algorithms for bit representation, including the VLC mapping and the VLC reordering.

2.3.1. Improved VLC mapping

As analyzed in [25,26], the *run/size* value of an *unused* VLC can be modified to the *run/size* of a *used* VLC. In other words, several Huffman codes can be used to represent one *run/size*. This modification is named as VLC mapping. Next, we propose an improved VLC mapping algorithm.

We first calculate the number of VLC occurrences in a JPEG bitstream. With a descending order of the occurrences, VLCs in each category C_i ($i = 1, 2, \dots, 16$) are sorted to

$$C_i' = \{VLC_{i,1}^{(u)'}, \dots, VLC_{i,p_i}^{(u)'}; VLC_{i,1}^{(n)}, \dots, VLC_{i,q_i}^{(n)}\}$$

where $VLC_{i,1}^{(u)'}$ stands for a sorted *used* VLC.

For category C_i', we construct several mapping sets using a one-to-$(2^j - 1)$ manner. In the standard Huffman table, there are 162 VLCs in total for AC coefficients, and the largest category is C_{16} which has 125 VLCs. In one-to-$(2^j - 1)$ manner, the maximum j should satisfy $2^j - 1 \le 125$, so $j \le 6$. Let the number of mapping sets in C_i' be $m_{i,j}$, where $m_{i,j}$ is integer and $m_{i,j} \ge 0$ ($1 \le j \le 6$). Each set contains one *used* VLC and 2^j–1 *unused* VLCs. Hence, each code in a set can be used to represent a j-bit additional message. According to the number of VLC occurrences in C_i, the mapping relationships are constructed to maximize the embedding capacity $EC_{i,1}$,

$$\begin{aligned} EC_{i,1} &= \max\Bigg\{6\cdot\sum_{v=1}^{m_{i,6}} h(i,v) + 5\cdot\sum_{v=m_{i,6}+1}^{m_{i,6}+m_{i,5}} h(i,v) + 4\cdot\sum_{v=m_{i,6}+m_{i,5}+1}^{m_{i,6}+m_{i,5}+m_{i,4}} h(i,v) \\ &\quad + 3\cdot\sum_{v=m_{i,6}+m_{i,5}+m_{i,4}+1}^{m_{i,6}+m_{i,5}+m_{i,4}+m_{i,3}} h(i,v) + 2\cdot\sum_{v=m_{i,6}+m_{i,5}+m_{i,4}+m_{i,3}+1}^{m_{i,6}+m_{i,5}+m_{i,4}+m_{i,3}+m_{i,2}} h(i,v) \\ &\quad + 1\cdot\sum_{v=m_{i,6}+m_{i,5}+m_{i,4}+m_{i,3}+m_{i,2}+1}^{m_{i,6}+m_{i,5}+m_{i,4}+m_{i,3}+m_{i,2}+m_{i,1}} h(i,v)\Bigg\} \\ &= \max\sum_{j=1}^{6}\left\{j\cdot\sum_{v=m'_{i,j+1}+1}^{m'_{i,j}} h(i,v)\right\} \\ s.\,t.\ &\sum_{j=1}^{6} m_{i,j} \leqslant p_i, \sum_{j=1}^{6} (2^j-1)\cdot m_{i,j} \leqslant q_i \end{aligned} \tag{1}$$

where $h(i, v)$ stands for the number of occurrences of $VLC_{i,v}^{(u)'}$,

$$m'_{i,j} = \sum_{l=j}^{6} m_{i,l},\ j = 1,2,\dots,6,\ m'_{i,7} = 0$$

This is an integer programming problem. The Eq. (1) can be resolved by traverse all possible $m_{i,j}$ parameters to find the result. After solving Eq. (1), the optimal values of $m_{i,j}$ can be identified, and the optimal code mapping for each category can be constructed.

During constructing the optimal code mapping, an unused VLC can be arbitrarily mapped to a specific used VLC, producing different combinations of the unused VLCs mapping. This feature can be used to improve the embedding capacity. In each category, when mapping the unused VLCs to a specific used VLC, we further use combinations of the unused VLCs represent more bits. The number of combinations is equal to

$$\begin{aligned} C_{i,e} &= \prod_{v=1}^{m_{i,6}} C_{63}^{p_i-63\cdot(v-1)} \cdot \prod_{v=m_{i,6}+1}^{m_{i,5}+m_{i,6}} C_{31}^{p_i-63\cdot m_{i,6}-31\cdot(v-m_{i,6}-1)} \cdot \\ &\prod_{v=m_{i,5}+m_{i,6}+1}^{m_{i,4}+m_{i,5}+m_{i,6}} C_{15}^{p_i-63\cdot m_{i,6}-31\cdot m_{i,5}-15\cdot(v-m_{i,5}-m_{i,6}-1)} \cdot \\ &\prod_{v=m_{i,4}+m_{i,5}+m_{i,6}+1}^{m_{i,3}+m_{i,4}+m_{i,5}+m_{i,6}} C_{7}^{p_i-63\cdot m_{i,6}-31\cdot m_{i,5}-15\cdot m_{i,4}-7\cdot(v-m_{i,4}-m_{i,5}-m_{i,6}-1)} \cdot \\ &\prod_{v=m_{i,3}+m_{i,4}+m_{i,5}+m_{i,6}+1}^{m_{i,2}+m_{i,3}+m_{i,4}+m_{i,5}+m_{i,6}} C_{3}^{p_i-63\cdot m_{i,6}-31\cdot m_{i,5}-15\cdot m_{i,4}-7\cdot m_{i,3}-3\cdot(v-m_{i,3}-m_{i,4}-m_{i,5}-m_{i,6}-1)} \cdot \\ &\prod_{v=m_{i,2}+m_{i,3}+m_{i,4}+m_{i,5}+m_{i,6}+1}^{m_{i,1}+m_{i,2}+m_{i,3}+m_{i,4}+m_{i,5}+m_{i,6}} C_{1}^{p_i-63\cdot m_{i,6}-31\cdot m_{i,5}-15\cdot m_{i,4}-7\cdot m_{i,3}-3\cdot m_{i,2}-1\cdot(v-m_{i,2}-m_{i,3}-m_{i,4}-m_{i,5}-m_{i,6}-1)} \\ &= \prod_{l=1}^{6}\prod_{v=m'_{i,l+1}+1}^{m'_{i,l}} C_{2^l-1}^{p_i-\sum_{j=l+1}^{6}(2^j-1)m_{i,j}-(2^l-1)\cdot(v-m'_{i,l+1}-1)} \end{aligned} \tag{2}$$