A Reversible Data Hiding Scheme For JPEG Images

Qiming Li, Yongdong Wu, and Feng Bao

Institute for Infocomm Research, A*Star, Singapore
{qli,wydong,baofeng}@i2r.a-star.edu.sg

Abstract. When JPEG images are used as cover objects for data hiding, many existing techniques require the images to be fully or partially decompressed before embedding. This makes practical application of these methods limited. In this paper, we investigate ways to hide data in the compressed domain directly and efficiently, such that both the original content and the embedded message can be recovered at the same time during decompression. We propose a method to embed bits into selected components of the compressed data such that it does not require decompression of the JPEG images and introduces very little change to the original JPEG files. The proposed method can be implemented efficiently and it is possible to perform embedding and detection in a single pass, so that JPEG streams can be processed in real-time without waiting for the end of the data.

Keywords: Reversible data hiding, compressed domain embedding, JPEG

1 Introduction

Digital data hiding techniques, such as watermarking and steganography, have been intensively studied during recent years. In a typical data hiding system, an *encoder* $\mathcal{E}$ embeds a digital message M (e.g., a stego image) into a cover object C (e.g., a digital photo) by slightly modifying the cover object without changing its semantics in a given application scenario. On the other hand, a *decoder* $\mathcal{D}$ extracts, or detects the existence of, the message M from an object C', which could be the cover object C, or a slightly distorted version of it due to noise in the transmission channel.

Data hiding techniques can be very useful in many scenarios, including authentication, fingerprinting, and tamper detection. Many such techniques, however, inevitably degrades the quality of the original cover data, since it is modified in a irreversible way. This may not be acceptable in applications that require a high fidelity. Reversible data hiding techniques are designed for such scenarios, where the decoder not only extracts the embedded message but also restores the original cover object to a “clean” state.

The JPEG standard is the most commonly used digital image standard in daily life, since it achieves very high compression ratio while retaining much of the image quality as perceived by human eyes. Most of the consumer cameras output JPEG photos by default, if it was not the only supported format. Therefore, JPEG images are very good candidates for cover objects when applying data hiding techniques.

However, using JPEG images as cover objects poses some challenges. Currently, many data hiding techniques for digital images work on the pixel level or a transform domain. To use JPEG images as cover objects, it is often required that the images are decompressed, optionally transformed, encoded, inverse transformed if they were transformed, and re-compressed. This gives rise to a number of problems.

First, the data hiding technique has to be robust against JPEG compression, which rules out most of the LSB schemes based on manipulating the least significant bits of the pixels. Secondly, decompression and re-compression not only introduces distortions to the image that reduce its quality, but also introduces changes to the statistics of the DCT coefficients in the JPEG stream that can be detected (e.g., [11]), which makes it not suitable for steganography applications. In addition, these techniques are usually computationally more expensive than those do not require decompression. Therefore, it is important to study data hiding techniques work directly on the compressed domain without decompression.

In this paper, we investigate efficient data hiding techniques that hide messages into JPEG images without decompressing them. We propose a method that (1) works directly on the quantized and entropy coded DCT coefficients, (2) requires very little changes to the JPEG bit stream, and (3) is able to process the bit stream efficiently on-the-fly without seeing the entire JPEG data. Our main idea is that, for some selected DCT coefficients for each DCT block, we treat it as an integer and modify its value in a reversible way. The corresponding entries in quantization tables may be optionally modified to reduce the distortion as seen by a standard JPEG decoder.

In Section 2, we give a brief overview of the data compression in the JPEG standard. We will review previous reversible data hiding methods for JPEG images in Section 3. The proposed method is given in detail in Section 4. We evaluate the performance of the proposed scheme in Section 5. We further give an optional step of quantization level re-mapping as a way to reduce distortion in Section 6. We then compare our scheme with the previous schemes in Section 7, and conclude in Section 8.

2 Background

The JPEG standard is created by Joint Photographic Experts Group for digital image compression. There are a number of variants under the standard, among which the most commonly used ones are lossy compressions with Huffman coding. Given an RGB color image, a JPEG encoder first converts the pixels from the RGB color space to the YCbCr space. The chrominance information (i.e., the Cb and Cr components) is typically down-sampled by a factor of 2 in one or both dimensions.

There are several different possible coding configurations from this point. In the most common configuration, the color components are interlaced and coded in one pass (i.e., sequential coding). In such a configuration, the two dimensional image is padded (if necessary) and divided into Minimum Coded Units (MCUs). The size of an MCU may vary for different down-sampling factors. However, each dimension of an MCU will always be a multiple of 8. As a result, each MCU contains a number of 8×8 *blocks*.

For each 8×8 block, a two dimensional Discrete Cosine Transform (DCT) is applied to the pixels, which results in 64 DCT coefficients. These DCT coefficients are then quantized using 64 uniform scalar quantizers specified by a quantization table. Each color component may be quantized using a different quantization table. The quantized DCT coefficients are then arranged in a zig-zag order and entropy coded, typically using one or more Huffman codes. The Huffman encoding schemes for the first DCT coefficient (commonly referred to as the DC coefficient) and the other coefficients (i.e., AC coefficients) are slightly different. However, we will not go into details here since we are only interested in AC coefficients in this paper.

For each block B of quantized DCT coefficients arranged in zig-zag order, we will refer to the coefficients as $B = \{c_0, \cdots, c_{63}\}$, where c_0 is the DC coefficient, and $c_1, \cdots, c_{63}$ are AC coefficients. Let the quantization table for block B be $Q_B = \{q_0, \cdots, q_{63}\}$, where q_i represents the step size for the i-th scalar quantizer. When there is no ambiguity, we will drop the subscript B and simply use Q to represent quantization tables. Note that these quantizers will always quantize towards nearest integers. In other words, given an AC coefficient d_i $(1 \leq i \leq 63)$, which is the i-th coefficient resulted from two dimensional DCT, the i-th coefficient c_i is computed as

$$c_i = \left\lfloor \frac{d_i + q_i/2}{q_i} \right\rfloor$$

where $\lfloor x \rfloor$ denotes the largest integer no greater than x.

Conversely, given a quantized DCT coefficient c_i $(1 \leq i \leq 63)$, the dequantization step during decompression simply computes the restored DCT coefficient $\hat{d}_i$ as

$$\hat{d}_i = c_i q_i$$

for $1 \leq i \leq 63$.

We have purposely left out the DC coefficient (i.e., c_0) in the above formulation, since in this way there is a one-to-one correspondence between the quantized DCT coefficient c_i and a particular Huffman codeword. DC coefficients are coded slightly differently and one extra step is needed to convert between c_i and the actual Huffman codeword.

3 Related Work

Much work has been done on digital watermarking, such as that discussed by Cox et al. [1]. However, many schemes assume that the cover object is available in an uncompressed form, or the watermarked objects will be distributed in an uncompressed form, or both. When dealing with JPEG cover images, it is possible to apply a generic data hiding technique, for example, spread spectrum method, directly on the DCT coefficients, but this is usually not desirable since it may introduce large distortion to the images.

There are a number of data hiding techniques tailored for JPEG cover images. In some of these schemes (e.g., [5, 10, 6, 2, 3, 13]), the basic idea is to manipulate the quantized DCT coefficients and sometimes the quantization table as well as to achieve the desired capacity and fidelity. Whereas there are also schemes (e.g., [9, 8]) that modifies the Huffman code used to encode the quantized DCT coefficients to hide secret messages.

To use the DCT coefficients for data hiding, there are two major issues that have to be addressed. The first is the selection of the DCT coefficients. Some of the previous schemes (e.g., [10]) prefer high frequency coefficients, some prefer low frequency coefficients (e.g., [6]), and others use mid-frequency coefficients (e.g., [2, 3]).

The second issue is the actual technique to embed data into the selected coefficients. The most often considered paradigm is to apply LSB embedding techniques directly to the selected DCT coefficients (e.g., [5, 10, 2, 3, 13]). For example, the least significant bits of the DCT coefficients can be simply replaced by the data to be embedded.

Reversible data hiding techniques first appeared in a patent owned by The Eastman Kodak [4], where a watermark is embedded into the spatial domain of an image by simply adding to the pixel values modulo 256. After the watermark is read, the original image can be restored by subtracting the watermark from the watermarked image. A similar method is proposed by Macq [7] in a multi-resolution manner. However, these methods suffer from possible large distortions to the images due to the modulo operations.

Fridrich et al. [2, 3] propose two practical methods for reversible data hiding for JPEG images using mid-band AC coefficients. In the first method, the LSBs of some selected DCT coefficients are compressed first and embedded together with the payload to achieve reversibility. We will refer to this as the DCT-LSB method. In the second method, the entries in the quantization tables corresponding to the selected DCT coefficients are either reduced to half of the original values or smaller (1 in the extreme case), so that there are at least two candidates for each new DCT coefficient such that it will be quantized to the same value as the original when the original quantization step is applied. In this way, one bit can be embedded in each DCT coefficient by choosing a particular candidate for it. The second method can also be considered as applications of the difference expansion technique [12] on the quantization error. We will refer to the second method as DCT-Q.

We note that the requirement that the new quantization step has to be less than half of the original is unnecessarily restrictive, which is only imposed by the use of quantization error. In our proposed scheme, we relax this requirement and allow the quantization step to remain unchanged if quantization level re-mapping is not used. Even when quantization level re-mapping is used, we allow any positive quantization step that is less than the original.

4 Proposed Method

4.1 Models and Notations

In this paper we are only concerned with the quantization tables and the quantized DCT coefficients in the JPEG compressed stream. As mentioned earlier in Section 2, the DCT coefficients in a JPEG stream are organized into MCUs, each of which in turn contains a number of blocks. Hence, we consider a JPEG stream as a sequence of blocks of DCT coefficients. Note that this can be applied to both grayscale and color sequential JPEG images.

For each block $B = \{c_0, \cdots, c_{63}\}$, we consider a subset of it to be suitable for data embedding, and we denote the indices of the subset as $I = \{i_1, \cdots, i_n\}$, where $n < 63$. In other words, the coefficients $c_{i_1}, \cdots, c_{i_n}$ will be used for data hiding. Without loss of generality, we assume that we embed m message bits per block.

As mentioned in Section 3, the encoder $\mathcal{E} = (\mathcal{S}, \mathcal{C})$ for DCT based data hiding techniques consists of two main algorithms, namely, the selection algorithm $\mathcal{S}$ that selects a subset of DCT coefficients, and the embedder $\mathcal{C}$ that actually hides the bits in the selected coefficients.

The decoder $\mathcal{D}$, however, may or may not know how the coefficients are selected. It suffices if the decoder knows which coefficients are selected. After that, the embedding process is reversed to obtain the hidden message as well as the original content. Hence, in the following, we focus on the selection algorithm $\mathcal{S}$ and the embedder $\mathcal{C}$, and how the later can be reversed.

4.2 DCT Coefficient Selection

A generic paradigm is proposed by Fridrich et al. [3] for lossless embedding. When it is applied to DCT coefficient selection in JPEG images, the LSBs of the DCT coefficients are required to be (1) randomizable without causing visible artifacts, and (2) compressible.

As we will see, since most quantized AC coefficients in a typical JPEG block are very small, this paradigm is unnecessarily restrictive. Hence, in this paper, we only consider requirements directly related to applications. In particular, a number of different strategies can be used as the following.

1. (**Fixed Subset**) The simplest way to specify a subset for data hiding is to choose a particular subset based on the analysis of a set of existing cover objects and the requirements posed by the application scenarios. Note that the selected subset must be known by the decoder before the communication begins. It could be "hard-coded" into the software, for example, or distributed as part of the secret key to decode the hidden message.

2. (**Capacity Requirement**) As we will see in Section 4.3, we only hide data in DCT coefficients that are zeros. Hence, the subset of DCT coefficients must contain enough number of zeros to hold the message that we want to hide. As a result, the selection result can be different for different images, and the decoder has to either understand the selection rule to pick the right coefficients for decoding, or be informed of the selection results.
3. (**Distortion Requirement**) Since each DCT coefficient may have different effects in terms of leaving artifacts in the resulting image, we may need to analyze the cover image and choose the coefficients that have the least visual impact. Similar to the previous case, the selection results has to be known by the decoder.
4. (**Least Modifications**) It is often desirable that a data hiding method should make as few changes to the cover object as possible. Based on this requirement, the selection of DCT coefficients should have a reasonable ratio of embedded bits against the amount of changes we make.

It is known that modifications to the DC coefficients would easily cause blocking artifacts. Hence we only consider AC coefficients here. If we select high frequency coefficients, it would be necessary to reduce the corresponding quantization steps to make the changes visually insignificant, since those quantization steps are often the largest. However, this requires extra modifications to the quantization tables and there are applications (e.g., steganography) where this is not desirable. Furthermore, the compression ratio may be affected since part of the compression power of JPEG comes from the fact that most of the high frequency coefficients are zeros and can be omitted all together during the encoding by the use of the EOB (end of block) symbol.

Therefore, we believe that the best coefficients for data hiding would be in the mid-frequency band. As we will see, our embedding method hides bits into zero coefficients. Hence, for capacity considerations, we need to choose coefficients that are very likely to be zeros. This can be either a fixed subset resulted from statistical analysis of cover images, or adaptively chosen after analyzing a given cover image. A disadvantage of the adaptive approach is that it needs one additional pass before the data is embedded or decoded, whereas a fixed subset allows real-time encoding and decoding.

4.3 Proposed Data Hiding Method

Our main idea is to design a function f that maps an original number $c \in \mathcal{M}$ for some domain $\mathcal{M}$ (in our case, all possible values for a quantized DCT coefficient) to a new value $c' = f(c)$, such that the following conditions hold.

1. The absolute value of the difference $|c - c'|$ is small.
2. There exists a function g such that $c = g(c')$ for every c and c'.
3. There exists a function h such that $h(c')$ outputs either a bit of 1 or 0, or the symbol $\perp$, which indicates that no bits were hidden in c'.

For example, if the domain $\mathcal{M} = \mathbb{Z}$ is the set of all integers, one way to define functions that satisfy the above requirements is as the following.

$$\begin{aligned} f(x) &= \begin{cases} 0 \ \text{ or } \ \pm 1, \text{ when } x = 0 \\ x + 1, \text{ when} x > 0 \\ x - 1, \text{ otherwise.} \end{cases} \\ g(x) &= \begin{cases} 0, \text{ when } |x| \leq 1 \\ x - 1, \text{ when} x > 1 \\ x + 1, \text{ otherwise.} \end{cases} \\ h(x) &= \begin{cases} |x|, \text{ when } |x| \leq 1 \\ \perp, \text{ otherwise.} \end{cases} \end{aligned} \tag{1}$$

As can be easily seen here, the secret message to be embedded can be used as the randomness in function f, which can then be extracted by h. We will refer to these functions as integer expansion functions.

Now, suppose we have selected n DCT coefficients in a block B for data hiding. Let $C = \{c_{i_1}, \cdots, c_{i_n}\} \subset B$ be the selected coefficients. For each coefficient $c \in C$, we apply the integer expansion functions as in (1). Note that a message bit $m \in \{0, 1\}$ can be embedded when c is zero. In particular, given a coefficient c and the message bit m, the encoder does the following.

1. If c is non-zero, increase its absolute value by 1 and stop. The sign is left unchanged.
2. Choose c' such that $|c'| = m$, and the sign of c' is randomly chosen if c' is non-zero.

Similarly, on the decoder side, the subset $C' = \{c'_{i_1}, \cdots, c'_{i_n}\}$ is selected. Given an coefficient $c' \in C'$, the decoder does the following.

1. If $|c'| > 1$, decrease its absolute value by 1 and stop. The sign is left unchanged.
2. Output $|c'|$ as a message bit extracted from cover data.
3. Change c' to 0.

At the end of decoding, the DCT coefficients would be restored, and the hidden message would be extracted.

5 Performance Evaluation

We measure the performance of our data hiding scheme by examining its capacity, distortion and its effect on the file size. We do not intend to make our scheme robust to noise, since noise is generally not tolerated in the compressed domain regardless of the existence of hidden messages.

We modify libjpeg[1] to hide data into AC coefficients of our choice. A color Lena image of dimension 512x512 with 6144 DCT blocks is used with different

[1] A JPEG compression library developed by the Independent JPEG Group (http://www.ijg.org/).

AC coefficient selections, and each time we try to embed as many bits as possible. In our experiment we only embed into one AC coefficient to show the effect on different AC coefficients, but it should be noted that it is possible to embed into multiple AC coefficients at the same time.

The results of the first experiment are summarized in Table 1. In this experiment, the Lena image is JPEG compressed with a quality factor of 75, which is used as the "original" cover data, and random secret messages are embedded into different AC coefficients. The distortion is measured by the PSNR between the cover JPEG and the stego image to simulate the application scenario where only JPEG images are available as the cover data. The same experiment is repeated 10 times and the average values of the file size and PSNR are shown in the table.

	Original ($q = 75$)	$c[5, 2]$	$c[4, 4]$	$c[2, 5]$
File size (bytes)	35731	36588.9	37554.7	39150.3
File expansion	0	2.4%	5.1%	9.6%
Payload (bits)	0	3788 (61.7%)	4480 (72.9%)	5201 (84.7%)
PSNR (dB)	inf	43.4219	39.778	34.5169

Table 1. Performance at quality 75.

From Table 1, it is clear that there is a trade-off among file size expansion, capacity and distortion. In particular, as we choose an AC coefficient that is more likely to be 0 in a given image, we obtain a larger capacity (i.e., we can embed more bits into the image), but the file becomes larger and the distortion becomes higher as measured by PSNR.

For example, the selection of the 16-th AC coefficient (the one on the fifth column and second row) gives the best image quality in terms of PSNR, as well as the smallest file size expansion (about 2.4%), at the expense of smaller data hiding capacity (only about 61.7% of the DCT blocks can be embedded). When we choose the 19-th AC coefficient (second column and fifth row), however, we can embed into 5201 (about 84.7%) of the blocks at the cost of larger file size (expansion of about 9.6%) and lower PSNR.

Quality	75	80	85	90
File size (bytes)	39150.3	52560.4	63655.5	70968.2
File expansion	9.6%	5.3%	3.4%	2.8%
Payload (bits)	5201 (84.7%)	4393 (71.5%)	3747 (61.0%)	3733 (60.8%)
PSNR (dB)	34.5169	36.3063	38.5455	41.7728

Table 2. Performance at different qualities.

In our second experiment (Table 2), we examine how the quality factor affects the performance. In this experiment, we always choose the 19-th AC coefficient, but use Lena images compressed with different qualities as the original. In this case, the PSNR and file expansion are computed using the corresponding original for each quality. As we can see, when the quality increases, the capacity drops, and file expansion and PSNR become better. This is expected since higher quality images contain more details and the same AC coefficient becomes less likely to be zero.

6 Quantization Level Re-mapping

In our proposed method, all the non-zero coefficients need to be increased, and the distortion may become too large under certain scenarios. As we will see in this section, the distortion due to the increment of coefficients can be reduced if we modify the quantization table slightly.

Given a DCT coefficient c, let q be the the corresponding quantization step in the quantization table. Let p be another quantization step such that $0 < p < q$. Let α be the smallest positive integer such that $\alpha > 1$ and

$$\left\lfloor \frac{\alpha p + q/2}{q} \right\rfloor = \left\lfloor \frac{(\alpha+1)p + q/2}{q} \right\rfloor \triangleq \beta. \tag{2}$$

In other words, both αp and $(\alpha+1)p$ will be quantized to the same symbol (i.e., β) using q as the quantization step.

For example, suppose $p = 0.8q$, we have $\alpha = 2$ and $\beta = 2$, since both $\alpha p = 1.6q$ and $(\alpha+1)p = 2.4q$ will be both quantized to $\beta = 2$ if q is used as the quantization step. This is illustrated in Fig. 1.

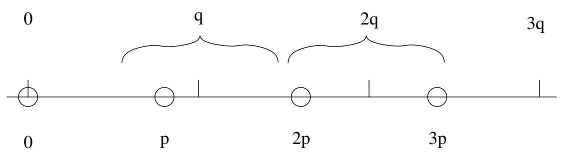

Fig. 1. Minimum α and corresponding β for $p = 0.8q$.

If we reduce the quantization step from q to p, we can always modify the value of the quantized DCT coefficient c to c' such that $c'p$ will be quantized to c using q as the quantization step. When $p < q$, as can be seen from Fig. 1, there will be multiple choices for c' that would achieve the same effect. Therefore, we can make use of this observation to mark the end of the increment due to embedding.

In particular, given a quantized DCT coefficient c with quantization step q and message bit m, let p be a new quantization step such that $0 < p < q$ and let α and β be defined as above, the enhanced encoder now does the following.

1. If $c = 0$, choose $|c'| = m$, and randomly choose a sign for c' if it is non-zero. Otherwise do the following.

2. If $0 < |c| < \beta$, increase its absolute value by 1. The sign is left unchanged.
3. If $|c| = \beta$, $|c'| = \alpha + 1$ and the sign of c' is the same as c.
4. If $|c| > \beta$, choose c' such that $|c'p - cq|$ is minimized.

Accordingly, given a DCT coefficient c' with quantization step p, let p, α and β be as defined above, the decoder performs the following steps.

1. If $|c'| \leq 1$, output $|c'|$ as a message bit, and output $c = 0$ as the restored coefficient. Otherwise do the following.
2. If $1 < |c'| \leq \alpha$, output c such that $|c| = |c'| - 1$ and the sign of c' is the same as that of c.
3. If $|c'| > \alpha$, output c such that $|c'p - cq|$ is minimized.

Let us give some numerical examples with $p = 0.8q$ where $\alpha = 2$ and $\beta = 2$ as in Fig. 1. If a selected DCT coefficient $c = 0$, we can embed the message bit by directly modifying its value. If the message bit is 0, the coefficient remain unchanged, otherwise it is changed to 1 or -1 randomly. If $c = 1$, we cannot embed data into it. We then follow the second step in the encoder, and change its value to 2. When $c = 2$, we see that $c = \alpha$, and follow the third step of the encoder and change its value to $\alpha + 1 = 3$. If $c = 3$, according to the fourth step of the encoder, we search for a c' such that $c'p - cq$ is minimized, and this c' happens to be 4. The decoding process is simply the reverse of the above steps. This is illustrated in Fig. 2, where the dashed lines represent random choices determined by the message bit.

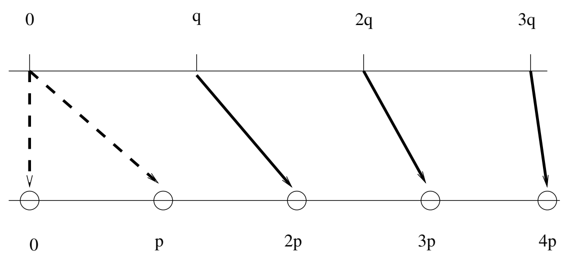

Fig. 2. Numerical encoding examples for $p = 0.8q$.

Note that whether to apply the quantization level re-mapping, as well as the choice of the values for p would be very much dependent on the application scenario, and we will leave it for future investigations.

7 Comparisons with Previous Schemes

As we have mentioned in Section 3, prior to our studies, there are two practical reversible data hiding methods in the JPEG compressed domain, which are proposed by Fridrich et al. [2, 3]. We refer to these methods as DCT-LSB and DCT-Q.

The DCT-LSB method requires compression of the LSB plan of the quantized DCT coefficients. As a result, the entire cover JPEG image has to be available before the encoding could start. In contrast, our method processes one DCT block at a time, hence is able to perform encoding and decoding on-the-fly when data becomes available. Furthermore, the DCT-LSB method requires the LSB plan of the quantized DCT coefficients to be highly compressible, whereas our method allows any selection of a subset of DCT coefficients, as long as there are sufficient number of 0's. In other words, we can embed data into DCT coefficients with almost truly random LSB.

The DCT-Q method requires the quantization step corresponding to the DCT coefficient to be either halved or reduced to some integer factor of it (in the extreme case, 1). While a small quantization step does allow smaller distortions, it can be undesirable in certain scenarios. For example, with a small quantization step, the absolute value of the quantized coefficient becomes larger. This may make it more difficult to compress the data stream.

In addition, reducing quantization steps drastically makes it easy to spot the existence of embedded data, hence making it undesirable for steganography applications. In our proposed method without quantization level re-mapping, the quantization table remains unchanged and it leaves no trace in the quantization table.

Even with the quantization level re-mapping, the relevant entries in the quantization table only needs to be slightly modified. This makes it possible to modify the entire quantization table so that it appears to be a JPEG image of a slightly higher quality factor. This would be difficult with the DCT-Q method without making the data stream much more difficult to compress.

To illustrate, let us look at an experiment with the Lena image. When compressing the Lena image using libjpeg version 7 and a quality factor of 75, we get a JPEG image of 35731 bytes. If we reduce all the quantization steps to no more than half of the values, we will need to use a quality factor of at least 88, which would result in a JPEG image of 65891 bytes, or a 84% increase in size. On the other hand, if we only reduce the quantization steps to 0.8 of the original values, the quality factor required is 80, which gives a JPEG image of 49908 bytes, or a 40% increase.

8 Conclusions

In this paper we investigate data hiding methods for JPEG images, which are the most common source of digital images nowadays.

Despite many existing methods for data hiding, most of them are not suitable for efficient processing of JPEG images. For example, some of them require the cover images to be in spatial domain or a transform domain that is not "native" to JPEG. There are a few previous embedding methods for JPEG images that directly manipulate the compressed stream. However, we found that their assumptions may be unnecessarily strong for generic JPEG data hiding.

We therefore propose a new paradigm of data hiding in JPEG images. Our methods are based on the observations that in most cases, the part of the compressed stream of the JPEG image contain mostly integers with small absolute values. Hence we believe that it is sufficient to embed bits as if the cover data are integers. To reduce the distortion the embedding method may introduce to the cover images, we further introduce an enhanced method that require the quantization table to be slightly changed.

We further note that the changes required to the quantization table and the compressed stream are very small and require only a single pass of the data. Furthermore the algorithms can be implemented efficiently by slightly modifying any existing encoder and decoder. These make our scheme suitable for efficient real-time applications.

References

1. Cox, I., Miller, M., Bloom, J.: Digital Watermarking. Morgan Kaufmann (2002)
2. Fridrich, J., Goljan, M., Du, R.: Invertible authentication watermark for JPEG images. In: International Conference on Information Technology: Coding and Computing. pp. 223–227 (2001)
3. Fridrich, J., Goljan, M., Du, R.: Lossless data embeddingnew paradigm in digital watermarking. EURASIP Journal on Applied Signal Processing 2, 185–196 (2002)
4. Honsinger, C., Jones, P., Rabbani, M., Stoffel, J.: Lossless recovery of an original image containing embedded data. US Patent Application, Docket No: 77102/E-D (1999)
5. Johnson, N., Jajodia, S.: Steganalysis of images created using current steganography software. In: Information Hiding Workshop. LNCS, vol. 1525, p. 273289. Portland, Oregon, USA (April 1998)
6. Luoa, W., Gregory L. Heilemana, b., Pizano, C.E.: Fast and robust watermarking of jpeg files. In: IEEE Symposium on Image Analysis and Inerpretation. pp. 158–162 (2002)
7. Macq, B.: Lossless multiresolution transform for image authenticating watermarking. In: European signal processing conference. pp. 1973–1976 (2000)
8. Mobasseri, B.G., Cinalli, D.: Lossless watermarking of compressed media using reversibly decodable packets. Signal Processing 86(5), 951–961 (May 2006)
9. Mobasseri, B.G., II, R.J.B.: A foundation for watermarking in compressed domain. Signal Processing Letters 12(5), 399–402 (May 2005)
10. Noguchi, Y., Kobayashi, H., Kiya, H.: A method of extracting embedded binary data from jpeg bitstreams using standard jpeg decoder. In: International Conference on Image Processing. pp. 577–580 (2000)
11. Pevny, T., Fridrich, J.: Detection of double-compression in jpeg images for applications in steganography. IEEE Transactions on Information Forensics and Security 3(2), 247–258 (2008)
12. Tian, J.: Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 13(8), 890–896 (2003)
13. Tseng, H.W., Chang, C.C.: High capacity data hiding in jpeg-compressed images. Informatica 15(1), 127–142 (January 2004)