Compressing JPEG Compressed Image Using Reversible Data Hiding Technique

Sang-ug Kang*, Xiaochao Qu†, and Hyoung Joong Kim‡
*Department of Computer Science, Sangmyung University, Seoul, 157-715 Korea
E-mail: sukang@smu.ac.kr
†Graduate School of Information Security, Korea University, Seoul, 136-701 Korea
E-mail: quxiaochao@gmail.com
†Graduate School of Information Security, Korea University, Seoul, 136-701 Korea
E-mail: khj-@korea.ac.kr

[1] ***Abstract*—Since the concept of reversible data hiding technique was introduced, many researchers have applied it for authentication of uncompressed images. In this paper, an algorithm is introduced to compress JPEG files again without any loss in image quality. The proposed method can modify an entire segment of VLC codeword sequence to embed a bit of data. The modified codewords may destroy the correlation, or the smoothness, between neighboring pixels of the recovered image. The data extractor utilizes the smoothness change to know the hidden data. For this, a novel smoothness measurement function which uses both inter- and intra-MAD values is proposed. When the smoothness change is small, two consecutive segments are concatenated to extract correct data with higher smoothness sensitivity. As a result, compression ratio or embedding capacity is increased in most natural images.**

I. Introduction

Reversible data hiding techniques embed message into an image and guarantee perfect extraction of the hidden message and recovery of the original image without any loss. The techniques have been evolved into two categories. Those applicable to original or uncompressed images fall in the first category (Category I), and those applicable to compressed images are considered to be the second one (Category II). In Category I, outstanding reversible watermarking algorithms have been advanced including integer transform [1], lossless compression [3], difference expansion [4], [7], [9], [11], histogram modification [8], prediction expansion [10], and accurate sorting and prediction [5] methods. The important aim of advancement is to minimize the difference between the original and cover image while maximizing data hiding capacity.

Since reversible data hiding techniques usually rely on redundancy in the cover image, it has been believed, theoretically, that reversible data hiding into random-look data is impossible. However, Category 2 techniques are more useful by considering the real world situation, in which most digital images are generated in compressed formats from various digital devices. Fridrich et al. [14] and Liu et al. [15] hide data directly into the bitstreams of JPEG and MPEG-2 files, respectively, using a code mapping method. For a H.264/AVC bitstream, intra prediction mode modification method is proposed for lossless data hiding. One of the major objectives of these algorithms is their data hiding capacity. In a way of thinking, data hiding techniques in Category II are identical to data compression ones if a part of compressed bitstream is hidden in the rest part of bitstream by reducing the entire file size by the amount corresponding to the capacity of algorithm.

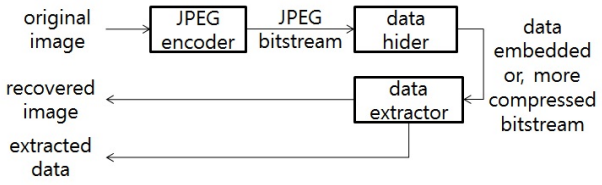


Fig. 1. Data hiding and extraction process of Category II techniques

Recently, Mobasseri et al. [2] exploit the redundancy in the entropy coded portion in a JPEG compressed bitstream to reversibly hide data by nearly preserving the file size. They observe that most commercial JPEG encoders use typical Huffman tables in the standard [13] and many of VLCs are not used for encoding a natural image due to the absence of customized entropy coding step. A used VLC is mapped to an unused one when a 1 is embedded and it is not mapped to embed a 0. Qian et al. [12] modify the approach in [2] to keep the JPEG file size and image quality the same after data hiding. Used VLCs are directly mapped to unused VLCs with the same code length.

Also Kim [6] hides data into JPEG compressed data making valid VLCs invalid and natural images unnatural. Symbol probability of JPEG output is almost uniform and random. Therefore, further entropy coding is almost impossible. However, since the method does not rely on the entropy coding, further compression, or data hiding, is possible by using the spatial information of the recovered image of JPEG bitstream. Kim et al. [6] show that one bit can be hidden in a 1024-bit segment of JPEG bitstream using content-aware code modification method. For the recovery of bitstream and data extraction illustrated in Figure 1, the method uses a measure called mean absolute difference (MAD) values computed with

[1]This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2012015587).

pixel values of a compressed image. The data extractor should know if the recovered image is a part of the image by using MAD values to extract a hidden data. In this paper, a new measurement function and data extraction method are introduced to achieve further compression, or data hiding capacity, and the result is compared with [2], [12] and [6].

The code mapping methods hide data only into the entropy coded portion shown in Figure 2 and the header portion, including auxiliary information for JEPG decoding, is modified to reflect the VLC codeword change due to the code mapping. The entropy coded portion contains many VLCs that are called as "used VLC". In the meantime, another group of VLCs are defined in JPEG header portion but those do not appear in the entropy coded portion that are called as "unused VLC". The unused VLCs do not exist if a JPEG encoder generates Huffman Tables optimized to a specific image contained in the bitstream. Since many JPEG encoder products skip this optimization step by simply using example Huffman tables provided in the standard, the code mapping methods utilize the unused VLC code space which occupies more than 50% of totally 162 VLCs in Table K. 5 & K. 6 in [13]. Qualified VLC pairs of (used VLC, unused VLC) are searched by building the Huffman code tree and those are used for code mapping. To embed a 0, a used VLC is not changed. To embed a 1, a used VLC is mapped to the unused VLC in the pair. The pairing strategy is different each other in [2] and [12]. In [2], a pair is generated if any 1-bit flip of a used VLC matches with an unused VLC. The run/size is modified in the JPEG header to avoid decoding synchronization and visual quality degradation because a mapping frequently causes run/size mismatch between the original VLC and the mapped VLC. In [12], a pair is established if there exists at least one unused VLC with the same code length as the used one. The run/size of mapped VLC is changed to the same value of the original's run/size for the purpose of synchronization in a decoder. This duplicated run/size value in the file header, unlike any other Huffman table, tells a data extractor all the VLC pairs so that the extractor can extract the hidden data and recover the image without any loss. The pair can be one to many for multiple-bit hiding.

However, the code mapping techniques are not applicable if a JPEG encoder is designed to maximize compression performance with customized Huffman tables which is recommended by the standard [13]. Even though code mapping techniques modify the entropy coded portion like in [6], the content-aware code modification approach works regardless of the Huffman table type, customized or typical. Also both different techniques can be used to a JPEG bitstream simultaneously because the two approaches are orthogonal each other. Throughout the paper, 512×512 grayscale images are used for computational convenience.

II. Data Hiding Algorithm

Note that this algorithm does not use probability or entropy theory for further compression. Instead it utilizes recovered image itself. A correctly recovered image looks natural and its neighboring pixels have strong correlation in spatial domain. The method exploits this correlation for compression as did in [6].

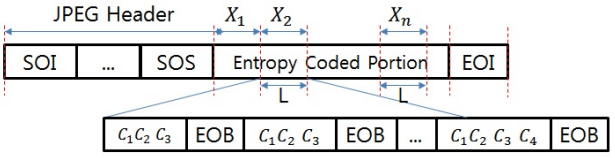


Fig. 2. Binary JPEG bit stream divided into n segments with length L.

Assume that JPEG binary stream to be recompressed is shown in Figure 2. Exact JPEG stream format is somewhat different from Figure 2, but for laconic explanation purpose, its model is simplified as such. The proposed method uses not the JPEG header portion but the entropy coded portion. The entropy coded portion is divided into many segments with fixed length L. The segment length L is decided such that each segment X_k contains bitstream portion corresponding to at least one JPEG block. There are more than one VLC codeword representing quantized DCT coefficients and End-of-block (EOB) in each block. Data embedding is carried out into the segments in the entropy coded portion. Data hiding scheme is given as follows:

$$Y_{k+1} = X_{k+1} \oplus b_k \cdot u \text{ for } k = 1, 2, \cdots, m, \qquad (1)$$

where X_k is the k-th segment, Y_k is the modified segment of X_k, and b_k is the k-th bit to be embedded. Unit vector u is a string of L ones. The symbol $\oplus$ denotes a bit-wise XOR (exclusive OR). To embed 0, set Y_k as X_k, and to embed 1, set Y_k as $\bar{X}_k$, where $\bar{X}_k$ is a bit-wise complement of X_k. At the data hiding stage, the first segment X_1 is not used for data embedding since this one has to be used as a reference image at the data extraction stage. As a result, Y_1 is simply X_1. Other segments are modified according to Equation 1.

The segment size L should not be too short or long. The data extractor can not work with too short L and the data embedding capacity becomes lower if L gets longer. The data hider should find a proper L through the method of trial and error using the extraction algorithm which will be explained in the next section. Therefore it is hard for the hider to find the exact minimum value of L. Instead the term of "optimum L" is used to denote that the L is not guaranteed minimum but nearly minimum while assuring the data extraction.

III. Data Extraction Algorithm

Recovering algorithm is also quite simple as shown in Figure 3. We recover the first a few 8×8 JPEG blocks by decoding Y_1. Since Y_1 is X_1, constructing these blocks is done easily. Recovering of the other blocks in segments Y_k for $k = 2, 3, \cdots, n$ comprises of two phases: a codeword validity phase and a smoothness check phase. In the first phase, we decode Y_k under the assumption that Y_k is X_k. If the JPEG decoding stops due to an invalid codeword or too many coefficients more than 64, it is obvious that b_{k-1}

TABLE I
THE SUCCESS RATIO OF JPEG CODEWORD VALIDITY PHASE OF $\bar{X}_k$

image (Q factor)	L	no. of segments	no. of success	no. of failure	success ratio (%)
Lena (50)	512	306	276	30	90.2
	1024	152	122	30	80.26
	2048	76	47	29	61.84
	4096	37	16	21	43.24
Baboon (50)	512	693	611	82	88.2
	1024	346	251	95	72.5
	2048	172	93	79	54.1
	4096	85	22	63	25.9
Barbara (50)	512	459	406	53	88.5
	1024	229	172	57	75.1
	2048	114	55	59	48.3
	4096	56	14	42	25.0
F16 (50)	512	340	302	38	88.8
	1024	169	136	33	80.5
	2048	84	55	29	65.5
	4096	41	16	25	39.0

is 1 and Y_k is $\bar{X}_k$. On the other hand, in case of successful JPEG decoding of Y_k, we need to do the codeword validity phase again for $\bar{Y}_k$. If the decoding stops, b_{k-1} is 0 and Y_k is X_k. When both cases are successful in the first phase, we have to decide which one is truly decoded and which one is luckily decoded. Not surprisingly, many flipped JPEG bitstreams are decoded because the typical Huffman tables in [13] encompasses many codewords and a variable length codeword has higher matching possibility than fixed length one because a codeword can be interpreted as many. To verify this, we flip all the segments, which is equivalent to embedding $b_k = 1$ into all the segments, and try to decode them. The simulation result is depicted in Table I. If a segment passes the codeword validity phase, it is called as "success" and otherwise "failure". The success ratio is quite high when L is small and it decreases as L increases. With a small L, more data can be hidden, but it is hard to take advantage of the simplicity and clarity of data extraction using the codeword validity phase. Also note that the success ratio is not dependent on image types.

The truly or luckily can be decided by using characteristics of images in an automatic manner which is the smoothness check phase. Natural images have strong correlation between neighboring pixels. If the JPEG decoding of Y_k is truly successful, the recovered image blocks will have strong correlation between blocks. In other words, the blocks are smoothly connected. However, if the JPEG decoding is luckily successful, the blocks are not strongly correlated each other. The phrase "luckily successful" means that the flipped JPEG bitstream is decoded by chance and the recovered image blocks look unnatural, meaningless, and inconsistent with neighboring blocks. To make automatic decision without human visual inspection, we introduce a smoothness measure M_k, which will be discussed in detail later on. Since M_k is designed to be smaller for truly successful images, we can extract the embedded data by comparing two M_ks of Y_k and $\bar{Y}_k$. If M_k of Y_k is less than M_k of $\bar{Y}_k$, then $b_{k-1} = 0$ and vice versa. The overall recovery process is portrayed in Figure 3.

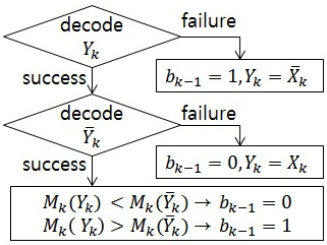
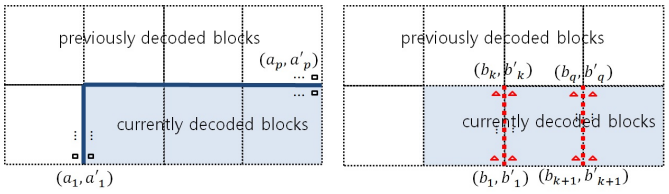


Fig. 3. Flowchart of the data extraction process.



(a) Inter MAD, $\tilde{M}_k$ (b) Intra MAD, $\hat{M}_k$

Fig. 4. Pixels associated with MAD calculation.

Assume that X_2 contains two complete JPEG blocks and one incomplete block, and X_3 has one incomplete, three complete and one incomplete blocks in sequence as shown in Figure 2. Note that a complete block starts with quantized DCT coefficient C_1 and ends with an EOB. In X_2, last block contains C_1, but following coefficients and EOB belong to X_3. Thus, last block is called incomplete and it is included in two segments. Coefficients of incomplete block in X_2 are used for X_2 data extraction in the codeword validity phase, but not in the smoothness check phase. In the smoothness check phase, only complete blocks in X_2 are used for X_2 data extraction and coefficients of an incomplete block in X_2 are decoded together with the following ones of an incomplete block in X_3 for X_3 data extraction. The coefficients of an incomplete block in X_2 is modified according to the X_2 recovery decision before X_3 data extraction. Those are used for both codeword validity and smoothness check phases of X_3 together with the coefficients of an incomplete block in X_3. And other three complete blocks in X_3 are processed continuously.

A. Smoothness Measurement Functions

A M_k value is calculated based on mean absolute difference (MAD). In the paper, the combination of two different MAD functions is used as a decision measure. First one is called as inter-MAD. The inter-MAD value, $\tilde{M}_k$, is computed using pixels which are placed between currently decoded blocks and the previously decoded neighboring blocks. In [6], only this $\tilde{M}_k$ is used. It is computed as follows:

$$\tilde{M}_k = \frac{\sum_{i=1}^{p} |a_i - a'_i|}{p} \quad (2)$$

where p is the number of pixel pairs. The relevant pixels a_i and a'_i, marked with square, are placed along the thick line as shown in Figure 4(a). The pixel element a'_i belongs to

the currently JPEG decoded blocks, and a_i belongs to the previously decoded blocks which are called reference blocks. The pixels a_i and a_i' face each other along the border.

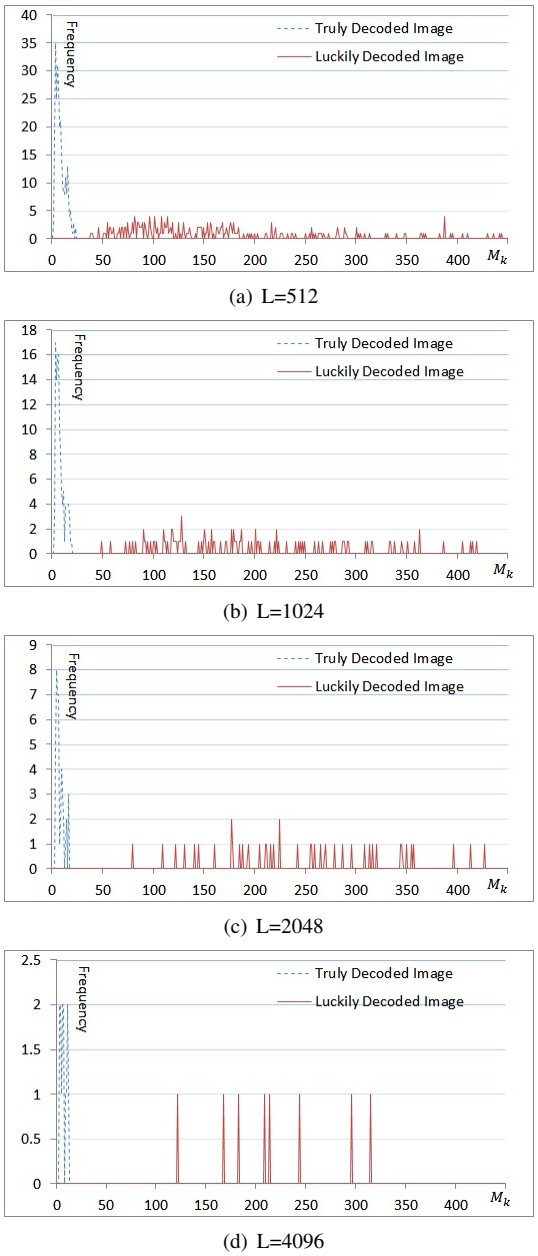
The second one is called as intra-MAD. The value, $\hat{M}_k$, is computed using pixels which are placed between currently decoded blocks as shown in Figure 4(b). If there is only one single block in the segment, the intra-MAD value cannot be computed. $\hat{M}_k$ is computed as follows:

$$\hat{M}_k = \frac{\sum_{i=1}^{q} |b_i - b_i'|}{q} \tag{3}$$

where q is the number of pixel pairs. The relevant pixels b_i and b_i', marked with triangle, are placed along the dotted thick line as shown in Figure 4.

$$M_k = \frac{\tilde{M}_k + \hat{M}_k}{2} \tag{4}$$

A MAD function is used in the smoothness check phase to differentiate a truly recovered image from a luckily recovered one. To illustrate the performance of M_k, the histograms of M_k at various L on Lena image are shown in Figure 5. MAD values in left side of the graph are from truly recovered images and those in right side are from luckily recovered ones, which passed the first phase. The performance of a MAD function is considered to be high when MAD_d is large, where MAD_d is the distance subtracting the maximum MAD value of truly recovered images from the minimum one of luckily ones. In Figure 5, it is apparent that MAD_d increases as L increases such as $MAD_d = 15$ at $L = 512$, $MAD_d = 29$ at $L = 1024$, $MAD_d = 63$ at $L = 2048$, and $MAD_d = 109$ at $L = 4096$. If MAD_d is greater than 0, data extraction is guaranteed for the image. This analysis can be done by a data hider to decide L. That's why the hider does not have to send a threshold information as in [6] and the extractor can still extract the hidden data without ambiguity. Note that the condition $MAD_d > 0$ is not a necessary condition but a sufficient one for a data hider to safely embed data. So it is possible to embed data even when $MAD_d \leq 0$. The performance of proposed MAD function is compared with the one in [6] as shown in Figure 6. The combination of inter and intra MAD outperforms $\tilde{M}_k$ over wide range of segment sizes by showing that MAD_d of M_k is larger than MAD_d of $\tilde{M}_k$ in most cases.



(a) L=512

(b) L=1024

(c) L=2048

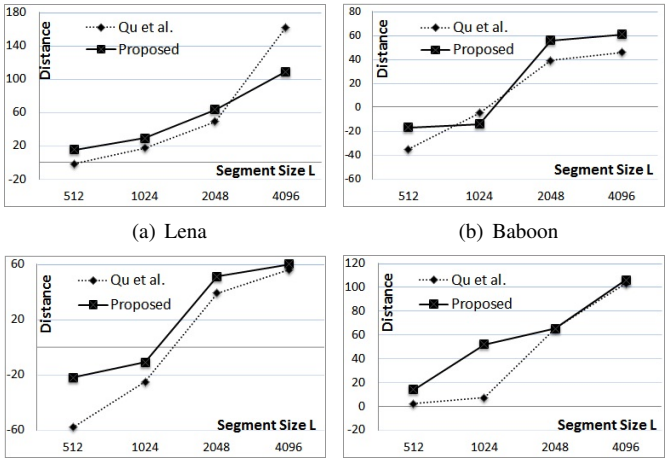
(d) L=4096

Fig. 5. The histograms of M_k at various L for Lena image.

B. Segment Concatenation

The motivation of segment concatenation is that longer segment size provides more information to decide what the hidden data is. In a segment X_k, a data hider can not embed data if the decision rule $M_k(X_k) < M_k(\bar{X}_k)$ is violated. If there exists at least one violating segment in the whole entropy coded portion, then the data hider should make L larger to avoid such a situation. Sometimes, however, the data hider can hide data with the same L size by merging two consecutive segments and extracting two bits simultaneously at the data extractor. Assume that a data extractor is handling Y_k and the decision rule is violated in that segment. Then a data extractor extracts $b_{k-1} = 1$ at the condition $M_k(Y_k) \geq M_k(\bar{Y}_k)$ even though $b_{k-1} = 0$ is embedded. Apparently this is an erroneous situation. However, the reversing of $M_k(X_k)$ and $M_k(\bar{X}_k)$ can be sometimes overcome using segment concatenation method especially when the reversing is small.

The segment concatenation starts when a segment meets the condition $|M_k(Y_k) - M_k(\bar{Y}_k)| < T_1$, where T_1 is a threshold value. Two consecutive segments Y_k and Y_{k+1} are concatenated. First, the codeword validity is checked for the combination of two segments, Y_kY_{k+1}, $Y_k\bar{Y}_{k+1}$, $\bar{Y}_kY_{k+1}$ and $\bar{Y}_k\bar{Y}_{k+1}$. If a combination fails, then the MAD value of corresponding combination is set to a large value so that it can not be the minimum value in the upcoming MAD com-



(a) Lena (b) Baboon

(c) Barbara (d) F16

Fig. 6. The comparison of MAD_d of MAD functions at various L on four test images

parison. Second, the smoothness is checked. $M_k(Y_kY_{k+1})$, $M_k(Y_k\bar{Y}_{k+1})$, $M_k(\bar{Y}_kY_{k+1})$ and $M_k(\bar{Y}_k\bar{Y}_{k+1})$ are calculated and the smallest is picked. If $M_k(Y_kY_{k+1})$ is the smallest, the extracted data $b_{k-1}b_k$ is 00. Similarly, 01, 10 and 11 are extracted when $M_k(Y_k\bar{Y}_{k+1})$, $M_k(\bar{Y}_kY_{k+1})$ and $M_k(\bar{Y}_k\bar{Y}_{k+1})$ are the minimum, respectively. The segment concatenation can be extended theoretically to many segments, but we found that there is no significant improvement.

C. Data Hiding Capacity

It is clear that more data can be hidden when the segment size is small. The data hider decides the L size according to a certain criterion or the performance of the data extractor.

Assume that the hider decides L such that MAD_d is always greater than 0. It is simple, but L is somewhat long. For example, the size of JPEG compressed Lena image used in this paper is 157,557 bits excluding the JPEG header portion, compressed from the original size 2,097,152 bits. With $L = 512$, decided by referring to Figure 6(a), the number of segments is 306 as in Table I so that 305-bit long message can be embedded in the JPEG compressed stream. Assume that the message is detached from the JPEG stream itself, then 157,557 bit-long stream is decreased to 157,252 showing 0.19% lossless compression ratio. In case of Baboon image, the proper L size turns out to be 2048 bits as shown in Figure 6(b) and the total number of segments are 172. So the JPEG stream size 355,504 bits can be reduced to 355,333 bits, compressed 0.05%. Of course MAD_d is greater than 0 at $L = 1500$ and it can be also used as a proper L. At $L = 256$ for Lena image, theoretically the compression ratio is expected to increase to 0.38%, but MAD_d is -5. In this case, successful data extraction is not guaranteed. It does not mean, however, that data extraction is impossible because the overlapping of MAD histograms can be caused by MAD values calculated from different segments. Roughly, as seen in Figure 6, this approach works well at $L = 2048$ for most natural images at JPEG quality factor 50.

TABLE II
THE OPTIMUM SEGMENT SIZE OF DIFFERENT IMAGES WHEN SEGMENT CONCATENATION IS NOT USED AND THE HISTOGRAM OVERLAPPING IS ALLOWED

image	Lena	Baboon	Barbara	F16
Optimum L	280	380	360	280
Capacity (comp. ratio (%))	561(0.36)	934(0.26)	654(0.28)	636(0.36)

The overlapping, $MAD_d \leq 0$, is allowed if it is guaranteed that M_k of truly recovered image is less than that of luckily one in the same segment. With this approach, the L size can be reduced and the capacity is improved as enumerated in Table II. For example, Lena image allows $L = 280$, showing 0.36% compression, and the L information is sent to the extractor. Other images permit different L sizes. Note that the optimum L value is proportional to textual complexity of an image.

In [6], a threshold value of MAD, M_{th}, is sent to a data extractor. If M_k of truly recovered image is less than M_{th}, then the current block is truly recovered image and $b_{k-1} = 0$. When both M_{th} and L are sent, the extracting complexity is reduced because b_k is extracted if either $M_k(Y_k)$ or $M_k(\bar{Y}_k)$ is computed and compared with M_{th}. Only one global M_{th} is needed to be sent when global MAD histogram method is used. On the other hand, many M_{th}s, one M_{th} per segment, should be sent when overlapped histogram is allowed.

The proposed algorithm includes the segment concatenation method. Needless to say, the capacity is maximized in this case and the comparisons with existing algorithms are performed.

D. Experimental Results

The proposed algorithm preserves file size and hides data in a JPEG bitstream like in [2], [12] and [6]. Those are comparable with each other in terms of data hiding capacity. For the purpose of fair comparison, the same test images are chosen, except synthetic images, and the same JPEG quality factors are used for JPEG compression using typical Huffman Tables in [13]. The results are shown in Table III. Different quality factor results in different number of VLCs and different size of entropy coded portion. The changes due to quality factor variation directly effect the embedding capacity. By comparing to Kim's method, the proposed algorithm always results in better performance caused by improved smoothness check functions and decision strategy. The results compared with [12] show that the proposed one has higher data hiding capacity on average as shown in Table IV. However, for some images like Baboon and Splash, the result is somewhat poorer than [12]. This is due to the use of image's smoothness in the process of data hiding. Note, however, that both methods are compatible with each other, meaning that more data can be hidden by using two methods simultaneously. Besides, the proposed algorithm is fully standard compatible because optimized Huffman Tables and VLC codewords can be used. The use of optimum Huffman tables is known to give 2-6% extra savings in file size.

TABLE III
DATA EMBEDDING CAPACITY (IN BITS) VS. JPEG QUALITY FACTORS.

Image	Method	10	20	30	40	50	60	70	80	90
Baboon	[2]	792	2035	1071	1342	820	887	953	317	195
	[12]	2398	1395	1047	1100	630	1081	1178	306	618
	[6]	287	384	423	341	394	271	266	120	56
	Proposed	502	921	979	1027	1015	627	804	201	286
Boat	[2]	476	350	457	355	439	514	231	270	528
	[12]	975	596	394	541	587	645	513	792	942
	[6]	248	293	217	259	348	399	357	247	80
	Proposed	338	536	545	732	566	649	844	747	711
Bridge	[2]	719	554	469	583	683	244	283	161	188
	[12]	1832	519	305	804	878	497	425	584	455
	[6]	173	293	256	347	320	307	365	183	78
	Proposed	372	654	610	733	920	1232	628	928	880
Elaine	[2]	111	130	139	200	249	79	92	146	339
	[12]	347	270	445	398	465	291	408	535	848
	[6]	181	328	365	344	330	394	272	236	79
	Proposed	249	411	609	689	720	899	981	1336	1122
F16	[2]	545	400	310	367	401	298	259	210	222
	[12]	751	780	980	624	787	873	526	549	318
	[6]	171	256	215	255	217	248	293	147	84
	Proposed	328	467	642	697	874	906	1022	1286	1337
Lena	[2]	263	197	235	259	317	94	99	86	152
	[12]	552	310	480	261	359	152	248	320	416
	[6]	201	300	228	272	314	258	311	140	141
	Proposed	319	392	498	598	749	824	1040	940	776
Peppers	[2]	350	247	338	242	278	357	164	144	378
	[12]	522	298	485	351	398	465	372	333	634
	[6]	205	300	287	277	269	267	226	245	95
	Proposed	309	451	500	604	703	815	988	740	120
Splash	[2]	284	369	261	296	378	547	383	359	296
	[12]	545	316	765	842	893	1058	702	691	605
	[6]	161	167	216	265	312	294	254	330	68
	Proposed	161	292	435	426	503	492	893	927	1083
Tiffany	[2]	226	348	222	310	332	391	534	332	279
	[12]	836	820	507	638	468	430	625	629	339
	[6]	188	267	346	419	366	244	299	228	102
	Proposed	265	402	416	504	588	688	601	611	931

TABLE IV
COMPARISON OF AVERAGE EMBEDDING CAPACITY (IN BITS) OF ALGORITHMS

Image	Qian	Kim	Proposed	Improved Rate
Baboon	1083.70	282.44	706.88	-34.77
Boat	665.00	272.00	629.78	-5.30
Bridge	699.89	258.00	773.00	10.45
Elain	445.22	281.00	779.56	75.10
F16	687.67	209.55	839.89	22.14
Lena	344.22	240.56	681.78	98.07
Peppers	428.67	241.22	772.00	80.09
Splash	713.00	229.67	579.11	-18.78
Tiffany	588.00	273.22	556.22	-5.40
Total bits	5655.37	2287.66	6318.22	11.72

IV. CONCLUSION

Compressing compressed image technique is described and analyzed in terms of a smoothness check function, segment concatenation and data hiding capacity. The smoothness check function is used to recognize if a recovered image is natural or synthetic. It tells that more understanding on natural images can enhance the overall performance. By introducing inter MAD in addition to intra MAD, the smoothness is checked with higher precision. Since the proposed method embeds data into fixed length segments at regular interval, more data can be hidden with shorter L. With segment concatenation method, it is possible to get more information on smoothness check. The proposed algorithm makes images re-compressed independently on their formats and the same idea can be applied to motion pictures and other multimedia formats.

REFERENCES

[1] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform,"*IEEE Trans. on Image Processing*, vol. 13, no. 8, pp. 1147-1156, 2004.

[2] B. G. Mobasseri, R. J. Berger, M. P. Marcinak and Y. J. Naikraikar, "Data embedding in JPEG bitstream by code mapping," *IEEE Trans. Image Processing*, vol. 19, no.4, Apr. 2010.

[3] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-lsb data embedding," *IEEE Trans. on Image Processing*, vol. 14, pp. 253-266, 2005

[4] L. Kamstra and H. Heijmans, "Reversible data embedding into images using wavelet techniques and sorting," *IEEE Trans. on Image Processing*, vol. 14, no. 12, pp. 2082-2090, Dec. 2005.

[5] S. Kang, H. J. Hwang and H. J. Kim, "Reversible watermarking using an accurate predictor and sorter based on payload balancing," *ETRI Journal*, vol. 34 pp.410-420, Jun. 2012.

[6] H. J. Kim, Apparatus and method for image compression using lossless data hiding scheme, Korea Patent Application No. 10-2013-0001285, 2013

[7] H. J. Kim, V. Sachnev, Y. Q. Shi, J. Nam, and H. G. Choo, "A novel difference expansion transform for reversible data hiding," *IEEE Trans. on Information Forensics and Security*, vol. 3, no. 3, pp. 456-465, 2008

[8] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354-362, 2006.

[9] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y.-Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Trans. on Circuits and Systems for Video Technology*, 2009, 19, pp. 989-999

[10] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 721-730, 2007.

[11] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.13, no. 8, pp. 890-896, 2003.

[12] Z. Qian, X. Zhang, "Lossless data hiding in JPEG bitstream," *Journal of Systems and Software*, vol.85, no. 2, pp. 309-313, Feb. 2012.

[13] Int. Telecommunication Union, CCITT Recommendation T.81, Information Technology - Digital Compression and Coding of Continuou-stone Still Images - Requirements and Guidelines 1992.

[14] J. Fridrich, M. Goljan, Q. Chen, and V. Pathak, "Lossless data embedding with file size preservation," *Proc. El SPIE, Security and Watermarking of Multimedia Contents*, San Jose, 2004, vol. 5306, pp. 354-365.

[15] H. Liu, F. Shao, and J. Huang, "A MPEG-2 video watermarking algorithm with compensation in bit stream," *Proc. DRMTICS*, 2005, pp. 123-134.