2011 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) December 7-9, 2011

Bitstream Size Suppression for DCT-based Information Hiding Method

KokSheik Wong and Simying Ong
Faculty of Comp. Science & Info. Technology,
University of Malaya, Malaysia.
Email: {koksheik@, simying@siswa.}um.edu.my

Kiyoshi Tanaka
Faculty of Engineering,
Shinshu University, Japan.
Email: ktanaka@shinshu.ac.jp

Xiaojun Qi
Computer Science Department,
Utah State University, USA.
Email:xiaojun.qi@usu.edu

***Abstract*—Although ScaScraIH [1] is able to scalably scramble image/video and offer scalable carrier capacity for reversible data embedding [2], the product of ScaScraIH suffers from bitstream size increment of $\sim 9.17\%$ on average. The main cause of bitstream size increment in ScaScraIH is storing the number of nonzero coefficients in each 8×8 block which is utilized during the descrambling process. This paper proposes two techniques in suppressing bitstream size increment for ScaScraIH. The first technique exploits the distribution of nonzero DCT coefficients in the image to construct a scanning order, aiming to shorten the distance between nonzero coefficients in a block. For the second technique, combination of predictive and entropy coding is used to encode the number of nonzero coefficients. Experiments are conducted by using standard test images to verify the effectiveness of both techniques in suppressing bitstream size increment. On average, bitstream size increment is significantly suppressed to $\sim 0.91\%$, with some of the processed images assuming smaller bitstream size than its original counter part.**

I. Introduction

Encryption and external information insertion are the two main disciplines in information hiding [3]. Encryption is the translation of plaintext message into unintelligible ciphertext [4]. It plays very important roles, ranging from encrypting our daily password to encrypting highly confidential data such as military image, copyrighted multimedia content, medical image, etc. However, it requires high complexity when multimedia content such as image, video and audio are treated and encrypted in the bitstream level [5]. Thus, partial encryption is applied in most of the information hiding schemes to ensure format compliance as well as to preserve bitstream size.

Some representative partial encryption schemes for DCT compressed image/video include modification on intra prediction mode [6], intra-block shuffle and sign flipping. However, these partial encryption (hereinafter scrambling) schemes suffer from leakage of information of the original content. For example, Li et. al [7] proposed NZCA (nonzero counting attack) that generates a rough sketch of the original image without prior knowledge on the scrambling algorithm used. To counter NZCA, FLBS (Full Inter-Block Shuffle) [7] is proposed to scramble all the AC coefficients from the same subband for destroying the information on the number of nonzero coefficients in a block. However, when the AC coefficients are scrambled within the respective subbands (as in FLBS), correlations exploited by the zigzag ordering is completely destructed. This causes a significant increase in the bitstream size, which is a classical problem in exisiting scrambling schemes in the compressed domain [4].

External information insertion methods embed additional information such as metadata, fingerprint [4], watermark [8], etc. into a host content for various purposes, including quick content retrieval, legal distribution of copyrighted content, claim of ownernership, etc. Recently, reversible information hiding (some authors use the terms lossless, restorable, invertible, etc.) has received much attention because of its attractive feature that can perfectly restore the original content after extraction of payload. This is particularly useful in applications where any form of permanent distortion in the content is not permitted.

In recent years, some joint techniques of scrambling and external information insertion are proposed. Zhang proposed an information hiding method in encrypted images where the inserted information can be extracted without decrypting the encrypted image, and vice versa [9]. Joint work is also proposed for broadcasting paid video content [4]. Specifically, a video is scrambled prior to transmission and a unique fingerprint of the buyer is inserted into the decoded video prior to actual display. Both techniques proposed in [4] and [9] are not reversible. The joint work in ScaScraIH [1] addressed this issue to completely restore the original content and offer scalable carrier capacity [2]. However, it suffers from bitstream size increment.

In this work, two techniques are proposed to suppress bitstream size increment in ScaScraIH [1], [2]. The first technique exploits the distribution of nonzero DCT coefficients in the image to construct a scanning order, aiming to shorten the distance between nonzero coefficients in a block. The second technique combines predictive and entropy coding to encode the number of nonzero coefficients. Experiments are then conducted by using standard test images to verify the effectiveness of both techniques in suppressing bitstream size increment.

II. Review of ScaScraIH [1], [2]

Let $G(x,y)$ be the (x,y)-th DCT coefficient block in a JPEG image G, and let $G_{i,j}(x,y)$ be the (i,j)-th coefficient in $G(x,y)$ where $1 \leq i,j \leq 8$. Here, $1 \leq x \leq M$ and $1 \leq y \leq N$ where $M \times N$ is the dimension of G. The

978-1-4577-2166-3/11/$26.00 ©2011 IEEE

ZRL (zerorun-level) pairs in each block are recorded and en-queued into the list L_α to undergo permutation later. Since ScaScraIH treats each ZRL pair as a whole (without changing the number of zerorun), there is no bitstream size increment due to scrambling. However, to completely reconstruct the original image, the number of nonzero coefficients $\zeta(x,y)$ in $G(x,y)$ is recorded in the form of ZRL pair and stored in the list L_β. In particular, the ZRL pair becomes (zerorun, level) $= (0, \zeta(x,y))$ if $\zeta(x,y) > 0$ or $(1,1)$ to signify no nonzero coefficient is in the block. These newly introduced ZRL pairs are en-queued to the list L_β in which case L_β will be appended to L_α and permuted altogether using a secret key κ to form the list L_κ.

The scrambled image G' is constructed by de-queueing $n \times \mu$ ZRL pairs from L_κ and putting them into n blocks where $\mu = |L_\kappa|/(M \times N)$ and $n \in \mathbb{N}$. These coefficients are arranged in specific ways determined by VQD (virtual queue decomposition) to encode the payload. In particular, if a n-tuple $(x_1, x_2, ..., x_n)$ denotes a specific bitstring, the number of nonzero coefficients in $G(x, y+i)$ are forced to be x_i for $x_i \geq 0$, $\sum x_i = n \times \mu$, and $0 \leq i \leq n-1$. The carrier capacity of VQD is scalable in which case higher carrier capacity is achieved when larger n is considered. Refer to [2] for more information.

The original image and the embedded payload can be perfectly reconstructed from the image processed by ScaScraIH using the same key κ. Although ScaScraIH allows scalable distortion to the perceptual fidelity of the image as well as offering scalable carrier capacity, it causes significant bitstream size increment in its product. The main cause to this problem is that new ZRL pairs are introduced to the bitstream to record the number of nonzero coefficients $\zeta(x,y)$ in $G(x,y)$.

III. Methodology

In this work, two techniques are proposed to suppress bitstream size increment in ScaScraIH [1], [2].

A. Adaptive Scanning Order

In this section, we propose a novel image dependent scanning order. First, a 63-bin histogram is built to capture the statistics of all 63 AC subbands in all 8 x 8 blocks in an image. The number of nonzero coefficients in each AC subband is recorded. The bins in histogram are then sorted in a decreasing order according to their frequencies (i.e., the number of times the subband corresponding to the histogram bin assumes a nonzero value). In particular, the proposed technique creates a generic scanning order which is optimal for the image itself. Bins of higher frequency are considered first in the scanning path because it is likely that those AC subbands will assume nonzero value. $S7$ in **Fig. 1** shows a representative scanning order constructed for the image Lena (quality factor set to 80).

In addition to the proposed adaptive scanning order, more gain (in terms of compression) is expected by integrating other scanning orders. Seven additional scanning orders (as shown in **Fig. 1**) are adopted from exiting work of Itoh [10] and Pan [11]. An exhaustive search approach is then taken. Every block of the image is scanned by using all eight scanning orders and the resulting Huffman codewords are recorded. The scanning order which produces the least number of bits (with respect to the resulting Huffman codewords) will be selected as the scanning order for that particular block. The distribution of the selected scanning orders is recorded in **Table I**. It is obvious that the proposed adaptive scanning order (denoted by $S7$) is, by far, the mostly selected order, followed by the default zigzag scanning order (denoted by $S2$) as suggested in JPEG standard [12] for most of the cases. Bitstream size

1	2	3	4	11	12	13	14
5	6	9	10	18	17	16	15
7	8	20	19	27	28	29	30
21	22	25	26	31	32	33	34
23	24	35	36	43	44	45	46
37	38	41	42	47	48	49	50
39	40	51	52	57	58	59	60
53	54	55	56	61	62	63	64

(a) S0

1	2	3	7	11	13	18	26
4	5	8	10	15	20	28	33
6	9	12	16	21	29	35	41
14	17	22	23	30	36	42	46
19	24	25	31	37	43	47	51
27	32	38	44	48	52	55	56
34	39	45	49	53	57	59	61
40	50	54	58	60	62	63	64

(b) S1

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

(c) S2

1	4	6	14	19	27	34	40
2	5	9	17	24	32	39	50
3	8	12	22	25	38	45	54
7	10	16	23	31	44	49	58
11	15	21	30	37	48	53	60
13	20	29	36	43	52	57	62
18	28	35	42	47	55	59	63
26	33	41	46	51	56	61	64

(d) S3

1	5	7	21	23	37	39	53
2	6	8	22	24	38	40	54
3	9	20	25	35	41	51	55
4	10	19	26	36	42	52	56
11	18	27	31	43	47	57	61
12	17	28	32	44	48	58	62
13	16	29	33	45	49	59	63
14	15	30	34	46	50	60	64

(e) S4

1	4	12	13	36	37	52	53
2	5	11	14	35	38	51	54
3	10	15	25	34	39	50	55
6	9	16	26	33	40	49	56
7	17	24	27	32	41	48	57
8	18	23	31	42	47	58	61
19	22	28	30	43	46	60	63
20	21	29	44	45	59	62	64

(f) S5

1	2	3	6	7	8	19	20
4	5	10	9	17	18	22	21
12	11	15	16	24	23	28	29
13	14	25	26	27	31	30	44
36	35	34	33	32	42	43	45
37	38	39	40	41	47	46	59
52	51	50	49	48	58	60	62
53	54	55	56	57	61	63	64

(g) S6

1	2	4	6	18	24	30	39
3	5	7	10	15	26	31	33
9	8	11	14	21	29	34	35
12	13	16	20	25	37	41	42
17	19	23	27	32	45	46	44
22	28	36	38	40	49	53	50
43	47	54	51	48	56	58	52
57	61	62	60	55	64	63	59

(h) S7

Fig. 1. Scanning Orders

TABLE I
STATISTICS OF SELECTED SCANNING ORDERS IN TERMS OF THE NUMBER OF BLOCKS

Images	Airplane	Baboon	Boat	Lake	Lenna	Peppers
$S0$	200	253	756	209	261	223
$S1$	390	291	362	328	512	407
$S2$	679	702	562	729	665	615
$S3$	488	571	389	549	399	623
$S4$	162	496	65	216	103	166
$S5$	240	360	101	203	162	280
$S6$	135	225	372	140	250	237
$S7$	*1802*	*1198*	*1489*	*1722*	*1744*	*1545*

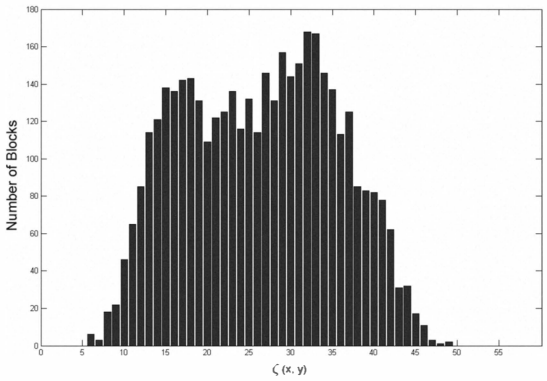

Fig. 2. Distribution of $\zeta(x, y)$ in Baboon

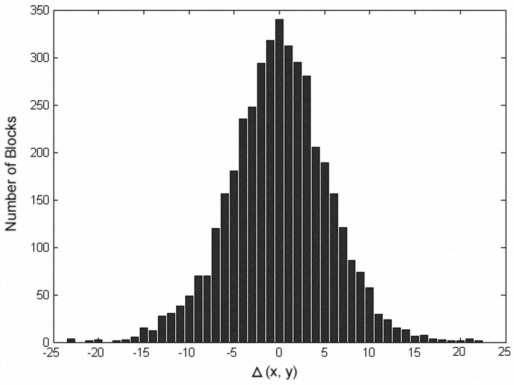

Fig. 3. Block Distribution of $\Delta(x, y)$ in Baboon

of a JPEG compressed image can be trimmed down by using adaptive scanning order because the zerorun length between two consecutive nonzero coefficients is decreased.

Nevertheless, if eight scanning orders are considered, $log_2(8) = 3$ bits per block must be spent on coding these side information to indicate which scanning order is used. By using prefix coding (eg., Huffman), the number of bits needed to encode the side information can be further compressed into ~ 2.3 bits per block (versus actually 3 bits per block) through simulations.

B. Predictive Coding

In [1], the number of nonzero coefficients in each block is coded in raw format, (i.e., $(0, \zeta(x, y))$ if $\zeta(x, y) > 0$ and $(1, 1)$ otherwise). This is inefficient because JPEG coefficients VLC (variable length coding) is defined in symmetrical range. For example, coefficients of category 1 consist of -1 and 1 while coefficients of category 2 consist of -3, -2, 2, and 3, and so forth. ScaScraIH ignores the negative range in VLC coding because all the $\zeta(x, y)$ are positive integers. Furthermore, since all the ZRL pairs are in $(0, \zeta(x, y))$ format, the number of bits needed to encode a ZRL pair depends on $\zeta(x, y)$. When $\zeta(x, y)$ increases (AC category also increases under the same zerorun), the number of bits needed to encode the corresponding ZRL pair also increases. Bitstream size can become very large if most of the $\zeta(x, y)$ concentrate on large values. The histogram of $\zeta(x, y)$ is shown in **Fig. 2** for Baboon (quality factor set to 80). It is obvious that $\zeta(x, y)$ is relatively large. It should be noted that, in [1], shorter codewords are assigned to rarely occurred cases (i.e, $(0, \zeta(x, y))$) while longer codewords are assigned to more probable cases. In particular, the cases of $1 \leq \zeta(x, y) \leq 6$, which generate the shortest codewords among all the positive values, are ignored. As a result, a significant number of bits are needed to encode L_β.

Predictive method is adopted in this paper to fully utilize both positive and negative ranges in VLC coding. Predictive method is feasible here because it is found, through empirical studies, that the number of nonzero coefficients $\zeta(x, y)$ in each block is approximately the same as the number of nonzero coefficients in its neighboring blocks. That is, it is found experimentally that $\zeta(x, y) \sim \zeta(x \pm \delta x, y \pm \delta y)$ for small δx and δy. For simplicity, we predict that a block $G(x, y)$ will have the same number of nonzero coefficient as its left neighbor $G(x, y-1)$. Instead of storing the raw value, we are storing the difference. As a preprocessing step, the number of nonzero coefficients $\zeta(x, y)$ in each block is first computed, and the average $\overline{\zeta}$ is calculated. Let $\Delta(x, y)$ be the prediction error for $G(x, y)$, and is computed as follows for all x:

$$\Delta(x, y) = \begin{cases} \zeta(x, y) - \overline{\zeta} & \text{if } y = 1 \\ \zeta(x, y) - \zeta(x, y-1) & \text{otherwise} \end{cases} \quad (1)$$

More importantly, $\Delta(x, y)$ is distributed around zero in a symmetrical range as shown in **Fig. 3**. Shorter codeword (regardless of its zerorun length) can be utilized to encode $\Delta(x, y)$ of higher frequency, and longer codeword is utilized to encode $\Delta(x, y)$ of lower frequency. For the worst case scenario, we need to utilize 127 cases since $|\Delta(x, y)| \leq 63$. Based on Eq. (1), $\Delta(x, y) = 63$ when the current block has 63 nonzero coefficients in it while the block to its left has none. The case $\Delta(x, y) = -63$ is just the opposite. In the assignment of codeword, both positive and negative codewords are fully utilized to achieve the most efficient way of coding these errors. These codewords are captured as ZRL pairs and then en-queued into the empty spaces in DCT coefficients blocks. In other words, the entries in L_β are updated to utilize this representation scheme.

IV. Discussions

Note that the proposed image dependent scanning order is applicable in all DCT-based compression standard for reducing bitstream size (i.e., gain in compression). However, this decrease in bitstream size is achieved at the expense of storing the scanning order as side information, which can be larger than the gain achieved. Refer to [10] for more detailed dicsussion on the trade-off. Nevertheless, the proposed scanning order is verified to be efficient and it contributes the most to the bitstream size suppression (refer to **Table I**). Furthermore, the number of scanning orders can be adjusted

TABLE II
BITSTREAM SIZE SUPPRESSED BY USING ADAPTIVE SCANNING ORDER

Images	Before (Bytes)	After (Bytes)	Gain (%)
Airplane	38727	36951	+4.59
Baboon	78687	76232	+3.12
Boat	49062	45890	+6.47
Lake	52223	50330	+3.62
Lena	37937	36121	+4.79
Peppers	39423	37533	+4.79

TABLE III
BITSTREAM SIZE SUPPRESSED BY USING BOTH ADAPTIVE SCANNING ORDER AND PREDICTIVE CODING METHODS

Images	Before (Bytes)	After (Bytes)	Gain (%)
Airplane	38727	39397	-1.73
Baboon	78687	78574	+0.14
Boat	49062	48570	+1.00
Lake	52223	52938	-1.37
Lena	37937	38584	-1.71
Peppers	39423	40124	-1.78

based on usability preference. For example, if less scanning orders are considered, the side information can be reduced but the compression ratio will also be compromised. Nevertheless, the carrier capacity in [1] can be scalably increased by using larger n (i.e., more blocks), which justifies the use of more scanning orders to better suppress the bitstream size increment.

In ScaScraIH, this side information will be embedded as part of the payload and hence reducing the effective carrier capacity. In addition, the introduction of predictive coding instead of coding the raw values also decreases the carrier capacity of ScaScraIH in two ways. First, the polarity of $\zeta(x, y)$ was exploited in ScaScraIH to encode information (e.g., positive $\rightarrow$ '1', and negative $\rightarrow$ '0'), but it is no longer available since both the positive and negative cases of each codeword are utilized. Secondly, in case the distribution of $\Delta(x, y)$ is not symmetric around zero, we need to explicitly code (as part of the payload) the mapping of VLC to $\Delta(x, y)$. Without maximizing the coding efficiency, in this work, we assume that the distribution is centered at zero, and the negative case (i.e., -1) is considered first before the positive case (i.e., 1).

Although predictive coding reduces the number of bits needed to code the number of nonzero coefficients in each block, it inevitably causes bitstream size increment since new ZRL pairs (i.e., foreign to the image) are introduced into the JPEG stream. The room made by using different scanning orders will be occupied by these ZRL pairs. As a result, we will further study other forms of error prediction functions, including the ones in JPEG-LS [13], to reduce the bitstream size increment.

V. EXPERIMENTS AND RESULTS

Two experiments are conducted to verify the effectiveness of the proposed techniques in suppressing bitstream size. Six standard images of size 512×512 pixels are used in the following experiments. These standard images are Airplane, Baboon, Boat, Lake, Lena and Peppers of 8 bits depth. Quality factor is set to 80 unless specified otherwise.

A total of eight scanning orders, including the proposed image dependent scanning order, are used in the experiment. **Table II** records the bitstream size before and after adaptive scanning is implemented. The last column records the gain in compression (i.e., the ratio of *difference in size* to *original size*) after adaptive scanning is implemented. Note that positive gain in the experimental result indicates a reduction in bitstream size. Suppression is achieved for all images, with an average suppression of $\sim 4.56\%$ of the original bitstream size.

TABLE IV
EFFECTIVE CARRIER CAPACITY [BITS] FOR VARIOUS BLOCK SIZES n

n	Airplane	Baboon	Boat	Lena	Peppers
2	8695.6	6126.8	9185.4	8660.6	8758.8
3	11967.9	8191.5	12244.8	11790.6	11891.9
4	13842.0	9526.2	14106.1	13661.7	13749.4
5	15020.2	10322.7	15210.8	14812.7	14885.5

Next, we consider the output bitstream size after applying both adaptive scanning order and predictive coding methods on [1]. The results shown in **Table III** are obtained after scrambling the image and inserting external information (i.e., random sequence of zeros and ones). It is observed that bitstream size increment is suppressed to $\sim 0.91\%$, which is significant when compare to the results in [1] (i.e., increment of $\sim 9.17\%$). Some processed images (e.g., Baboon and Boat) even assume smaller bitstream size than its original counter part. Therefore, we conclude that bitstream increment size of the manipulated image can be suppressed by using the proposed techniques.

We would like to stress again that bitstream size suppression is achieved at the expense of sacrificing the effective carrier capacity. **Table IV** records the carrier capacity $\Omega(n)$ for various $n \in \mathbb{N}$ for VQD [2]. When the proposed techniques are utilized, the effective carrier capacity is $\Omega(n) - 4096 \times 3$ bits since there are 4096 blocks in each image, and 3 bits are needed for each block. It is observed that the condition $n \geq 4$ must be satisfied in order to integrate the proposed techniques (except for Baboon) since the side information requires 12288 bits. For the case of Baboon, we have to consider smaller number of scanning orders (eg., 2 or 4) to suppress the number of bits needed to store the side information. In general, the effective carrier capacity can be improved by reducing the number of scanning orders, but at the same time, the effectiveness in suppressing bitstream size increment is also compromised. Thus, this is the trade-off between carrier capacity and bitstream size increment. Nevertheless, without sacrificing the performance in suppressing bitstream size increment, the effective carrier capacity can be improved by considering a larger n, but at the expense of higher computational complexity [2].

VI. Conclusions

In this paper, two techniques were proposed to suppress bitstream size increment in ScaScraIH. First technique reduced the number of zeros between two consecutive nonzero coefficients. Second technique predicted the number of nonzero coefficients in a block and coded only the difference. By integrating the proposed techniques into ScaScraIH, bitstream size increment of the manipulated image was significantly suppressed, on average, from $\sim 9.17\%$ to $\sim 0.91\%$. In our best case scenario, some of the manipulated images assumed smaller bitstream size than its original counter part.

As future work, we will further suppress bitstream size increment in ScaScraIH. We also want to seek for possible applications of the proposed techniques in suppressing bitstream size in other domains.

References

[1] K. Wong and K. Tanaka, "Dct based scalable scrambling method with reversible data hiding functionality," in *2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, march 2010, pp. 1–4.

[2] K. Wong, S. Ong, and K. Tanaka, "Improvement of carrier capacity for scalable scrambling method with reversible information insertion functionality," in *2011 IEEE International Conference on Signal and Image Processing Applications*, 2011, to be published.

[3] S. Katzenbeisser and F. A. Petitcolas, Eds., *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st ed. Norwood, MA, USA: Artech House, Inc., 2000.

[4] D. Kundur and K. Karthik, "Video fingerprinting and encryption principles for digital rights management," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 918 – 932, june 2004.

[5] H. Cheng and X. Li, "Partial encryption of compressed images and videos," *IEEE Transactions on Signal Processing*, vol. 48, no. 8, pp. 2439 –2451, August 2000.

[6] J. Ahn, H. Shim, B. Jeon, and I. Choi, "Digital video scrambling method using intra prediction mode," in *Advances in Multimedia Information Processing - PCM 2004*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3333, pp. 386–393.

[7] W. Li and Y. Yuan, "A leak and its remedy in jpeg image encryption," *International Journal of Computer Mathematics*, vol. 84, pp. 1367–1378, September 2007.

[8] C.-C. Wang and Y.-C. Hsu, "New watermarking algorithm with data authentication and reduction for jpeg image," *Journal of Electronic Imaging*, vol. 17, no. 3, pp. 1–1 – 1–8, 2008.

[9] X. Zhang, "Reversible data hiding in encrypted image," *Signal Processing Letters, IEEE*, vol. 18, no. 4, pp. 255 –258, April 2011.

[10] Y. Itoh, "An adaptive dct coding with geometrical edge representation," *IEICE Transactions on Information and Systems*, vol. E86-D, no. 6, pp. 1087–1094, June 2003.

[11] F. Pan, "Adaptive image compression using local pattern information," *Pattern Recognition Letter*, vol. 23, pp. 1837–1845, December 2002.

[12] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[13] G. K. Wallace, "The jpeg still picture compression standard," *Commun. ACM*, vol. 34, pp. 30–44, April 1991.