SPEEDING UP THE KNOWLEDGE-BASED DEBLOCKING METHOD FOR EFFICIENT FORENSIC ANALYSIS

[1]Yanzhu Liu, [2]Xiaojie Li and [2]Adams Wai Kin Kong*

[1]Institute of Automation, Chinese Academy of Science, China
[2]School of Computer Engineering, Nanyang Technological University, Singapore
Emails: yanzhu.liu@ia.ac.cn, xli16@e.ntu.edu.sg and adamskong@ntu.edu.sg

ABSTRACT

To restore skin features, e.g. skin marks, in JPEG compressed images for criminal and victim identification, Tang et al. proposed a knowledge-based (KB) deblocking method. Experimental results showed that it outperforms other deblocking methods designed for generic images. However, it is computationally demanding. In this paper, two hash functions, bitwise ℓ_1-minimization, a hybrid searching procedure and a parallel scheme are developed to speed it up. Experimental results demonstrate that the proposed computational techniques speed up the KB method more than 150% on average.

Index Terms**—**Skin images, forensics, skin marks, JPEG, biometrics

1. INTRODUCTION

Recently, skin marks and blood vessels have been proposed to identify criminals and victims in evidence images [1-2]. These features are small and very sensitive to image quality. Evidence images are very likely compressed by the JPEG method because it is widely installed in digital cameras. A knowledge-based (KB) method has been specially designed for removing JPEG blocking artifacts in skin images, so as to restore the original clarity of the images [3]. Experimental results demonstrated that this method can effectively improve the quality of skin images and perform better than other deblocking methods designed for generic images [5-7]. The KB method is extremely slow because it searches optimal uncompressed blocks in a large skin image database to replace the compressed blocks in evidence images. It may take several hours to process one image with 8M pixels. Even though more computing resources are given, the processing time cannot be reduced significantly because the KB method is not parallel. Thus, it is essential to speed up the KB method for legal cases with a lot of evidence images.

Before presenting the proposed computational techniques to speed up the KB method, a set of notations and a summary of the KB method are given. A target block is a block that is currently processing. A frequency neighborhood (FN) is the quantized discrete cosine transform (QDCT) coefficients in the to-be-processed blocks labeled as blocks 5-8 in Fig. 1(a). A spatial neighborhood (SN) refers to the 18 pixels in the processed blocks adjacent to the target block. Fig. 1(a) illustrates these terms. In this paper, the QDCT coefficients in a block are regarded as a row vector $q_e = [q_1 \cdots q_r]$ where q_i is a QDCT coefficient and $r < 64$ (8×8), because the quantized high frequency coefficients are always zero and can be ignored. The subscript e denotes that the vector is from an input image (evidence image). The row vector is called index vector, which is illustrated in Fig. 1(b). An FN is also represented by a row vector $f_e = [c_5\ c_6\ c_7\ c_8\ a_{5,1} \cdots a_{5,r_5}\ \ a_{6,1} \cdots a_{6,r_6}\ \ a_{7,1} \cdots a_{7,r_7}\ \ a_{8,1} \cdots a_{8,r_8}]$, where c_i is a quantized DC coefficient, a_{ij} is a quantized AC coefficient, $j \in \{1 \cdots r_i\}$ and $i \in \{5 \cdots 8\}$. $f_{ec} = [c_5\ c_6\ c_7\ c_8]$ and $f_{ea} = [a_{5,1} \cdots a_{5,r_5}\ \ a_{6,1} \cdots a_{6,r_6}\ \ a_{7,1} \cdots a_{7,r_7}\ \ a_{8,1} \cdots a_{8,r_8}]$ are defined to represent quantized DC coefficients and the quantized AC coefficients separately. The SN is also represented as a row vector s_e.

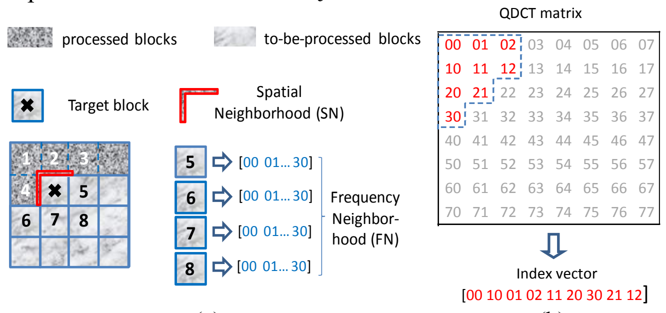


Fig. 1 Illustration of target block, frequency neighborhood, spatial neighborhood and index vector.

The KB method uses information from a given database to remove blocking artifacts and restore skin features. This

[*] All her work was done in the Nanyang Technological University.

database is produced by a set of uncompressed skin images and a JPEG quantization table. The original images are first compressed and then the QDCT coefficients (index vector), the FN and the SN of each block, which are respectively denoted as q_{di}, f_{di} and s_{di}, are extracted. The quantized DC and AC coefficients in f_{di} are denoted as f_{dci} and f_{dai}, respectively. The subscript d indicates that the vectors are generated from images in the database and the subscript $i \in \{1, \cdots, N\}$, where N represents the total number of blocks in the database, is an index to denote different blocks.

The KB method removes blocking artifacts in Y, U and V components separately. Input compressed images are processed block by block in a raster-scan order – from left to right and from top to bottom. Given a target block in a compressed image, the KB method first searches for uncompressed blocks in the skin image database, whose index vectors are the same as the index vector from the target blocks. Mathematically, this operation is to construct the set,

$$G(q_e) = \{\, i \mid \|q_e - q_{di}\|_1 = 0 \} \qquad (1)$$

where $\|\bullet\|_1$ represents ℓ_1 – norm. $G(q_e)$ is an index set and its cardinality is often very large, especially for U and V components. Then, the FN of the target block is used as a search query and ℓ_1 – norm is also used as a measure to reduce the size of $G(q_e)$. The remaining uncompressed blocks form the set

$$G'(f_e) = \{\, i \mid \|f_e - f_{di}\|_1 \le \|f_e - f_{dj}\|_1 \ \forall j \in G(q_e) \wedge i \in G(q_e) \} \qquad (2)$$

Note that the cardinality of $G'(f_e)$ can be greater than one. Finally, the optimal uncompressed block in the database for the target block defined by

$$j = \arg\min_i \|s_e - s_{di}\|_1 \quad s.t. \quad i \in G'(f_e), \qquad (3)$$

is obtained and the pixels in the target block are replaced with the optimal uncompressed block. Eqs. 1, 2 and 3 refer to respectively the first, second and third steps in the KB method.

To speed up the KB method, a number of computational techniques are proposed in this paper. Section 2 presents two hash functions for a new database structure. They reduce the computational complexity of Eq. 1 to O(1). Section 3 gives the bitwise ℓ_1–minimization to speed up the computation of Eq. 2 for U and V components. Section 4 offers a parallel scheme for multi-core computers. Section 5 compares the original KB method and the proposed implementation. Section 6 offers some conclusive remarks.

2. HASH FUNCTIONS FOR INDEX VECTORS AND A NEW DATABASE STRUCTURE

Given an uncompressed image database and a quantization table, the images are transformed to the YUV domain and compressed by the JPEG method. Then, index vectors (q_{di}), frequency neighborhoods (f_{di}) and spatial neighborhoods (s_{di}) are extracted. The information from U and V components is stored in a two-level directory structure. The first hash function, which maps an input index vector to the first level folder name, is defined as

$$h_1(q_e) = \sum_{i=1}^{r} s_i \times 2^{i-1}, \qquad (4)$$

where $s_i = 1$ if $q_i \ne 0$ and $s_i = 0$ if $q_i = 0$. Note that $q_e = [q_1 \cdots q_r]$. The second hash function, which computes the second level folder name, is defined as

$$h_2(q_e) = q'_1 q'_2 \cdots q'_r, \qquad (5)$$

where q'_i is a null character if $q_i = 0$ and $q'_i = q_i$ if $q_i \ne 0$. In each of these second level folders, there is only one file storing all FNs and each line in this file is one FN. Note that each line is distinctive, meaning that no duplicated FN is stored in the same file. For each FN, there are two corresponding files in the same folder: one stores all the SNs with the same FN and the other stores corresponding uncompressed pixels. h_1 and h_2 provide two strings as two folder names that are uniquely defined by the index vector. Clearly, there is no numerical comparison to be performed for locating the folder, which stores FNs and SNs with the same index vector as q_e. We can use h_1 and h_2 to implement Eqs. 1 for U and V components because their index vectors have only few non-zero coefficients.

The compression ratio of Y component is generally low and therefore, it has numerous different index vectors. To avoid generating file fragments, a single file is used to store all FNs. In this file, all FNs with the same q_{di} are saved continuously. Furthermore, a mapping structure, which is implemented by a MySQL table, is used to record the FN and SN positions in the file. When a search is performed, the input index vector is first used as a key to obtain the first and last byte location of FNs in the file. Based on these two values, the corresponding FNs are retrieved from the file. Similarly, SNs and uncompressed pixels are stored in two files.

3. A BITWISE ℓ_1–NORM FOR U AND V FREQUENCY NEIGHBORHOODS

The KB method uses ℓ_1 – norm as a measure to find optimal FNs i.e., $\arg\min_i \|f_e - f_{di}\|_1$, where $i \in G(q_e)$. In each comparison, ℓ_1–norm performs m-1 additions, m subtractions and m absolute operations, where $m = 4 + \sum_{i=5}^{8} r_i$ is the length of f_e and f_{di}. It is time consuming when the cardinality of $G(q_e)$ is large. Thus, a binary representation of FNs is proposed and a bitwise ℓ_1–norm is designed for this representation.

Before discussing the bitwise representation for FNs, a simple example is first introduced. Given two integers 3 and

4 and using Table 1 to encode them, their bitwise representations are respectively b_3^5=[1 1 1 0 0] and b_4^5=[1 1 1 1 0], where the superscripts are the order of the coding table indicating the number of bits used to represent one integer and the subscripts are the integers of the bitwise representations. Table 1 is an order five coding table. Clearly, $\sum_{i=1}^{5} b_{3i}^5 \otimes b_{4i}^5 = \|4-3\|_1$, where $\otimes$ represents a bitwise XOR. From Table 1, it can be easily observed that $\sum_{i=1}^{5} b_{ki}^5 \otimes b_{ji}^5 = \|k-j\|_1$, where $0 \le j,k \le 5$. The count function of the dynamic_bitset class in the C++ library is used to perform the summation. Thus, ℓ_1–norm can be implemented in two operations, one bitwise XOR and one bit count. This encoding method is modified from the generalized IrisCodes [8-9], which have been used to support high speed iris and palmprint matching [10].

Given a set of integers, $p_1, \ldots, p_u$, directly encoding them may require more bits. A more effective approach is to encode r_i where $r_i = p_i - \min_j(p_j)$. The minimum order of the coding table is $\max(p_j) - \min(p_j)$. A coding table with an order Δ can be employed. It can be represented by a matrix $\Omega = [\omega_{ij}]$, where $1 \le i \le \Delta$, $1 \le j \le \Delta + 1$, and ω_{ij} is defined by

$$\omega_{ij} = \begin{cases} 0 & if\ i \ge j \\ 1 & otherwise \end{cases}. \tag{6}$$

Using the previous notations, the FNs can also be represented as a matrix F_d, where each row is $[f_{dci}\, f_{dai}]$. For the sake of convenience, let F_{dc} and F_{da} be two matrixes and each of their rows stores respectively f_{dci} and f_{dai}. Since the range of the quantized DC coefficients in F_{dc} is larger than the range of the quantized AC coefficients in F_{da}, they are processed separately. By making use of the property of the coding tables, $r_{dcj}(x)$ is defined as

$$r_{dcj}(x) = F_{dcj}(x) - \min_x(F_{dcj}(x)), \tag{7}$$

where F_{dcj} is the j^{th} column of F_{dc} and $F_{dcj}(x)$ is its element, is encoded. Since the range of the quantized AC coefficients is very short, the global minimum of all quantized AC coefficients is used to define

$$r_{daj}(x) = F_{daj}(x) - \min_{j,x}(F_{daj}(x)), \tag{8}$$

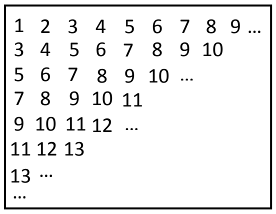
where F_{daj} is the j^{th} column of F_{da}. Once $r_{dcj}(x)$ and $r_{daj}(x)$ are encoded, the FNs are represented by a set of bits for comparison.

4. A PARALLEL SCHEME

To make use of multi-core processors, the KB method should be parallelized. The first and second steps do not use information from any other blocks to process a target block. Thus they can be parallelized directly. The last step searches optimal blocks using information from left, left-top, top and top-right processed blocks. In addition to data dependence, it is noted that even in the same image, some blocks have exactly the same QDCT coefficients and FNs. Considering these factors, a parallel algorithm is given in Fig. 2.

```
Input: Compressed Image
Output: Deblocked Image
Algorithm:
1. Extract QDCT coefficients and FNs of each block in the input
   image in parallel.
   1.1. Identify the unique QDCT coefficient blocks.
   1.2. Read in corresponding FNs from database in parallel.
2. Perform parallel searches to identify nearest FNs.
3. Perform parallel search according to the computing order
   illustrated in Fig. 3 to identify optimal SN and obtain
   corresponding uncompressed pixels.
4. Return the deblocked image.
```

Fig. 2 Pseudo code for parallel search.



```
1  2  3  4  5  6  7  8  9 ...
3  4  5  6  7  8  9  10
5  6  7  8  9  10 ...
7  8  9  10 11
9  10 11 12 ...
11 12 13
13 ...
...
```

Fig. 3 The order of spatial neighborhood computation

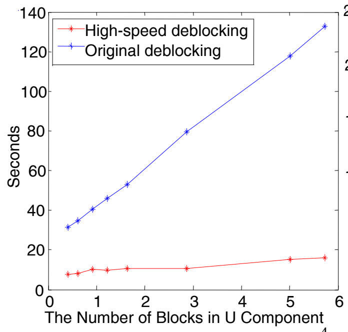
Table 1 Order 5 coding tables for the bitwise ℓ_1 – norm.

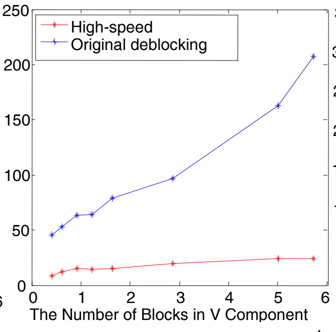
Input values	0	1	2	3	4	5
Bit 1	0	1	1	1	1	1
Bit 2	0	0	1	1	1	1
Bit 3	0	0	0	1	1	1
Bit 4	0	0	0	0	1	1
Bit 5	0	0	0	0	0	1

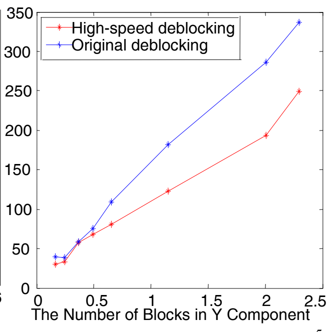
5. EXPERIMENTAL RESULTS

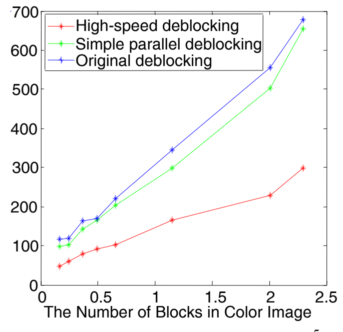
In the experiments, the skin image database containing 5,662 images was used to evaluate the speed of the proposed algorithm and the KB method, and the images in the database were compressed by the JPEG method with the Independent JPEG Group (IJG) quantization tables with quality factors (QF) of 50. 400 skin images with size of 128x128, 128x192, 192x192, 129x256, 256x256, 256x448, 448x448 and 448x512 pixels were examined. Each size has 50 images. In each test, the median time of processing 50 images with the same size was used as a performance index. Originally, the KB method was implemented in Matlab. For fair comparison, C++ was used to re-implement it. The experiments were performed on a 2.93 GHz PC with 4 cores and 4 GB memory, running Windows 7.

Fig. 4 plots computation time against size of input images in terms of the number of blocks. The number of blocks in U component (Fig. 4(a)) and V component (Fig. 4(b)) is only 25% of those in Y component because of the JPEG down sampling operation in these two components. Figs. 4(a) and (b) show that the proposed algorithm is significantly faster than the original KB method for U and V components. More precisely, for U and V components from



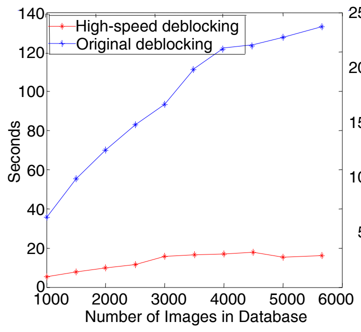


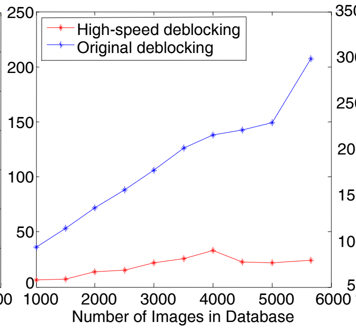


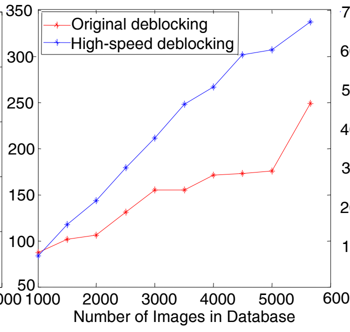


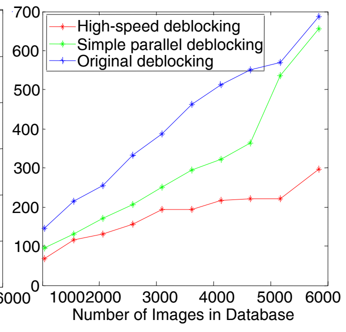
(a) the numbers on x-axis in 10^4 (b) the numbers on x-axis in 10^4 (c) the numbers on x-axis in 10^5 (d) the numbers on x-axis in 10^5

Fig. 4 The relationship between computation time and input image size in terms of number of blocks. The quality factor of input compressed images is 50. (a) U component, (b) V component, (c) Y component and (d) color images. (a color figure)









(a) (b) (c) (d)

Fig. 5 The relationship between computation time and database size. The quality factor of the compressed images is 50. (a) U component, (b) V component, (c) Y component and (d) color images. (a color figure)

an image with size of 448x512 pixels, the proposed algorithm is respectively 736% and 757% faster than the KB method. Fig. 4(c) demonstrates that the proposed algorithm can also effectively speed up the process for Y component. The improvement for Y component is not as great as the improvements for U and V components because U and V components have more duplicated blocks and because some proposed schemes are only applicable to them. To fairly compare the proposed algorithm with the KB method for color images in a computer with a multi-core processor, a simple parallel version of the KB method is implemented, whose performance is labeled as simple parallel deblocking in Fig. 4(d). This simple parallel version simultaneously processes Y, U and V components. Fig. 4(d) shows clearly that the proposed algorithm is significantly faster than the KB method and its parallel version. It was expected that the parallel version can effectively reduce the computation time. However, the KB method requires huge memory and I/O operations, which make the simple parallel process ineffective. These experimental results demonstrate that the proposed algorithm can increase the speed of the KB method by 120% on average.

Ten databases with 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000 and 5662 images were constructed to study the relationship between deblocking time and database size. The 50 testing images with size of 448x512 pixels were compressed using the IJG quantization table with quality factor of 50. Figs. 5(a) and (b) show that when the size of databases increases, computational time of both algorithms increases. However, the increasing rates of the proposed algorithm are much slower than that of the KB method. Figs. 5(c) and (d) show the experimental results for Y component and color images. The increasing rates of the proposed algorithm are also slower than that of the KB method.

6. CONCLUTION

In this paper, an algorithm is proposed to speed up the KB method, which is specially designed to remove blocking artifacts and recover skin features in JPEG compressed images for criminal and victim identification. Two hash functions, a bitwise ℓ_1–minimization and a parallel scheme are developed. Experiments are performed to compare the KB method and the new algorithm. The experimental results show that the proposed algorithm runs significantly faster than the KB method.

7 ACKNOWLEDGMENTS

This work is partially supported by the Ministry of Education, Singapore through Academic Research Fund Tier 2, MOE2012-T2-1-024.

8. REFERENCES

[1] A. Nurhudatiana, A.W.K. Kong, K. Matinpour, S.Y. Cho and N. Craft, "Fundamental statistics of relatively permanent pigmented or vascular skin marks", *International Joint Conference on Biometrics*, pp. 1-6, 2011.

[2] C. Tang, A.W.K. Kong and N. Craft, "Uncovering vein patterns from color skin images for forensic analysis", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 665-672, 2011.

[3] C. Tang, A.W.K. Kong and N. Craft, "Using a knowledge-based approach to remove blocking artifacts in skin images for forensic analysis", *IEEE Transactions on Information Forensics and Security*, vol. 2, pp. 1038–1049, 2011.

[4] D. Sun and W. Chan, "Postprocessing of low bit-rate block DCT coded images based on a fields of experts prior", *IEEE Transactions on Image Processing*, vol. 16, no. 11, pp. 2743-2751, 2007.

[5] A. Foi, V. Katkovnik and K. Egiazarian, "Pointwise shape-adaptive DCT for high-quality denoising and deblocking of grayscale and color images", *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395-1411, 2007.

[6] Y. Luo and R. K. Ward, "Removing the blocking artifacts of block-based DCT compressed images", *IEEE Transactions on Image Processing*, vol. 12, no. 7, pp. 838-842, 2003.

[7] J. Chou, M. Crouse and K. Ramchandran, "A simple algorithm for removing blocking artifacts in block-transform coded images", *IEEE Signal Processing Letters*, vol. 5, no. 2, pp. 33-35, 1998.

[8] A.W.K. Kong, D. Zhang and M. Kamel, "An analysis of IrisCode", *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 522-532, 2010.

[9] J.G. Daugman, "High confidence visual recognition of persons by a test of statistical independence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1148-1161, 1993.

[10] A.W.K. Kong, D. Zhang and M. Kamel, "An analysis of brute-force break-ins of a palmprint authentication system", *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, no. 5, pp. 1201-1205, 2006.