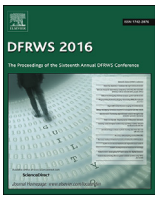
Digital Investigation 18 (2016) S108–S117

Contents lists available at ScienceDirect

Digital Investigation

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



DFRWS USA 2016 — Proceedings of the 16th Annual USA Digital Forensics Research Conference

Recovery of heavily fragmented JPEG files

Yanbin Tang [a], Junbin Fang [b], K.P. Chow [a], S.M. Yiu [a, *], Jun Xu [c], Bo Feng [d], Qiong Li [e], Qi Han [e]

[a] Dept. of Computer Science, The University of Hong Kong, China
[b] Dept. of Optoelectronic Engineering, Jinan University, China
[c] Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
[d] Department of Computer Science, Stony Brook University, USA
[e] School of Computer Science and Technology, Harbin Institute of Technology, China

Keywords:
Photo forensics
JPEG file carving
Fragmentation point
Color similarity
Fragmented JPEG file

A B S T R A C T

File carving from damaged file system plays an important role in file recovery for identifying evidence in digital forensics. In this paper, we focus on JPEG file carving, with an emphasis on heavily fragmented cases. The difficulty lies on how to order fragmented pieces into a complete picture without sufficient decoding information. We provide a framework to tackle this problem, which consists of the following key components: (i) a new similarity metric (CED) to evaluate if two data blocks are consecutive in the same JPEG file and a fragmentation point detection algorithm based on CED; and (ii) an overall recovery algorithm to reconstruct the JPEG file from fragmented pieces. The proposed framework was verified on an image dump from a SD card of a digital camera. The results were compared to Adroit Photo Forensic (APF), a commonly used photo carving tool. In our experiments, our tool can automatically recover 97% fragmented JPEG files (versus 79% by APF).

© 2016 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Introduction

There is a substantial number of crime cases involving computers and multimedia files, and it is not uncommon for suspects to erase the files or even destroy the file system before being seized by the law enforcement officer Garfinkel (2007). Reconstructing the photos from deleted file fragments – JPEG file carving – becomes an important technique to recover the digital evidence in these cases.

A number of JPEG carving techniques have been proposed (Sencar and Memon, 2008; Pal and Memon, 2009; Poisel and Tjoa, 2013) to reconstruct the deleted photos from binary data blocks obtained from a storage device. A common idea is to check consecutive blocks to determine if a fragmentation point is found. Then, we try to identify the beginning data block of the next fragment. The core technical problem is how to determine whether two data blocks are consecutive in the same JPEG file.

All existing tools are based on the assumption that the boundary of two consecutive data blocks should have similar colors. Similarity measures such as the sum of differences (SoD) or Euclidean distance (ED) are used to capture the differences of the colors of pixels on the boundary. However, this assumption is only valid for smooth regions. The differences of color values may differ a lot in other parts of the photo, such as a sharp region. Using SoD or ED, which mainly capture the absolute differences of color values, may not be able to correctly identify the fragmentation points in the sharp region and may cause the failure of the subsequent carving steps.

* Corresponding author.
E-mail addresses: ybtang@cs.hku.hk (Y. Tang), junbinfang@gmail.com (J. Fang), chow@cs.hku.hk (K.P. Chow), smyiu@cs.hku.hk (S.M. Yiu), jxu@ir.hit.edu.cn (J. Xu), bofeng@cs.stonybrook.edu (B. Feng), qiongli@hit.edu.cn (Q. Li), qi.han@hit.edu.cn (Q. Han).

http://dx.doi.org/10.1016/j.diin.2016.04.016
1742-2876/© 2016 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

We observe that although these differences fluctuate in different areas of the photos, they exhibit a locality property. That is, the variation pattern of color values within a small region is very similar. By taking advantage of this property, we propose a new metric, called Coherence of Euclidean distance (CED), and develop a more effective and robust fragmentation point detection algorithm to solve the problem. Also, to improve the efficiency of carving, instead of carving the file block by block as in some existing algorithms (Memon and Pal, 2006, De Bock and De Smet, 2016, APF, 2013), our algorithm groups JPEG data blocks into JPEG pieces (see the definition below) first and then takes each piece as the minimum data unit for processing. The following terminologies are frequently used in the paper. A *block*/*cluster* is the minimum allocation unit in a file system. A *fragment* or a *piece* has one or more consecutive blocks which belong to the same file. A *fragmented file* is one with two or more fragments. A *fragmentation point* is the last block in a fragment, i.e., where the fragmentation occurs. A *segment* has one or more consecutive fragments which belong to several files. An example illustrating these terms is shown in Fig. 1. In the figure, each square is a block and we show an example for consecutive blocks found in storage media.

The rest of this paper is organized as follows. Section 2 talks about related works. The proposed metric CED and the fragmentation point detection algorithm will be introduced in Section 3, followed by the piece-based JPEG photo recovery algorithm (Section 4). Section 5 shows the experimental results and Section 6 concludes the paper.

Related work

For file recovery, there are a number of popular forensic tools (e.g. Encase, FTK, The Sleuth Kit). Most of them focus on recovering files stored sequentially in harddisks and cannot handle fragmented files. With the knowledge of specific file formats or specific markers, some open source solutions (e.g. Foremost (2010), Scalpel by Richard and Roussev (2005), Multimedia File Carver by Poisel (2012)) can recover sequentially stored file even if filesystem metadata is not available due to deletions, or format, operations. The idea is to locate the file header and the footer (by searching for specific binary strings), then extract all data blocks in-between to recover the file. However, these methods do not work well for fragmented files. Fragmentation can become a serious problem for large files when the available free space is low. This is particularly pressing for removable media, such as flash memory for digital cameras and mobile phones, which typically employ the FAT file system, or use file system wear-leveling.

Cohen (2007) introduced a "semantic carving method" to reconstruct a file by taking advantage of unique markers and metadata information (e.g. in pdf and zip files) stored in fragments of the same file. However not all file types contain such information, and the method is not applicable to JPEG files. Garfinkel (2007) provides a solution for bifragmented files (files with only two fragments) with an identifiable header and footer. It is a brute-force approach, which generates all combinations of data blocks between the header and footer and to verify them using a JPEG renderer. Since there are many files with more than two fragments, the practicality of Garfinkel's algorithm is limited. Karresand and Shahmehri (2008) proposed a method to reassemble a special type of JPEG file with "restart markers", which is designed to handle bit stream error due to file corruption or unreliable transmission. In practice, most JPEGs do not have such restart markers. Even files with restart markers are not guaranteed to have one in every fragment, so the method can still fail.

Memon et al. formulated the problem of reassembling JPEG fragments as a Hamiltonian path problem and proposed several greedy heuristics to solve the problem (Shanmugasundaram and Memon, 2003; Pal et al., 2003; Memon and Pal, 2006). In their approach, each data block is considered a node, and the similarity between two blocks is the value of the edge between two nodes. Both Memon and Pal (2006) and Pal et al. (2008) use SoD as the metric for computing the similarity of two blocks and the tool Adroit Photo Forensic (APF) was developed based on this measure. APF has become a well-known and useful tool for image carving in forensics investigation. As shown in our experiments, the performance of APF degrades when the file is heavily fragmented or if the fragmentation point occurs at some sharp regions in the photo.

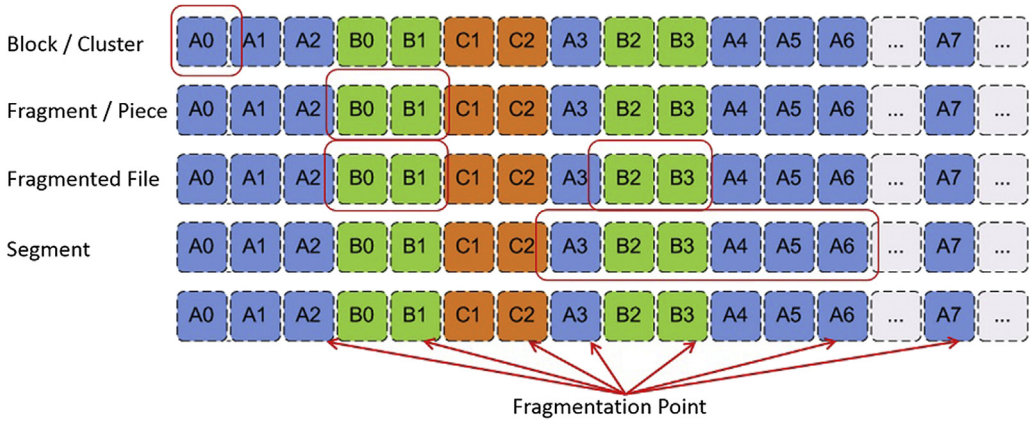


Fig. 1. An example for illustration of terminologies.

Guo and Xu (2011) proposed the use of a neural network algorithm to predict the next fragment to recover JPEG files. As discussed in their paper, it does not work well if the positions of fragments are not consecutive. Ying and Thing (2011) proposed an inequality-based fragmented file carving algorithm. However, it requires the identification and extraction of the fragments belonging to the same file first. In reality, fragments from multiple JPEG files are mixed together. De Bock and De Smet (2016) proposed a new approach to reconstruct fragmented JPEG files. However, it is also based on data block as the minimum unit of file carving and still cannot efficiently solve the problem of finding the start of next data block when a fragmentation point exists in-between.

In summary, most of the existing JPEG carving algorithms use SoD or ED as the matching metric, and as shown in the next section, the effectiveness of these metrics may be an issue. Also, these approaches are block-based algorithms, and are not scalable to large-scale investigation when fragments of multiple files exist on the storage media.

The proposed metric (CED) and fragmentation point detection

In this section, we first review two commonly used similarity metrics for JPEG file carving, namely Sum of Differences (SoD) and Euclidean Distance (ED). Then, we show how to use CED to detect fragmentation points.

Review of SoD and ED

Based on the characteristics of smooth contents in images, two commonly used similarity metrics: Sum of Differences (SoD) and Median Edge Detection (MED), are used to evaluate whether two fragments (blocks) are adjacent in most of the existing methods (e.g. Memon and Pal, 2006). SoD is applied in APT due to its linear computational complexity. For a fragmented image, SoD is calculated as the sum of differences between the RGB values of the pixels on the boundaries of two blocks. The normalized formula for SoD is:

$$SoD = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - y_i\right| \tag{1}$$

where x_i is the RGB value of the pixel from one fragment, and y_i is the RGB value of the corresponding pixel from the other fragment, and n is the number of pixel pairs involved in the boundaries.

While computing SoD is straightforward and efficient, SoD is not always a good metric in identifying fragmentation points.[1] For example, as shown in Fig. 2, there are three individual pixels, R_0 (black), R_1 (gray) and R_2 (blue), which have different RGB values, and therefore the SoD values of (R_0, R_1) and (R_0, R_2) are:

$$SoD_{(R_0,R_1)} = (|0-30| + |0-30| + |0-30|)/1 = 90$$

$$SoD_{(R_0,R_2)} = (|0-0| + |0-0| + |0-90|)/1 = 90$$

In this example, both $SoD_{(R_0,R_1)}$ and $SoD_{(R_0,R_2)}$ are 90, meaning that pixels R_1 and R_2 should have the same color similarity to R_0 using the measurement of SoD. However, it is obvious that R_0, R_1 and R_2 are different colors and R_1 (gray) is more similar to R_0 (black) than R_2 (blue). If the pixels, R_1 and R_2 from two fragments, are compared to pixel R_0 to evaluate the matching likelihood using SoD, it is difficult to distinguish which one should be more similar.

Besides SoD, Euclidean Distance (ED) is another similarity metric commonly applied in device independent color space to measure color difference (Sharma, 2002). ED is similarly computed from the RGB values of the pixels across the boundary. Lower ED value means higher similarity.

$$ED = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2}$$

In Li et al. (2012), ED is applied for file carving and the experiment shows that more reliable results can be obtained than using SoD. For example, by using ED to compare the color similarity of Fig 2, R_1 (gray) is more similar to R_0 (black) than R_2 (blue).

$$ED_{(R_0,R_1)} = \sqrt{(0-30)^2 \times 3}\Big/1 = 30\sqrt{3}$$

$$ED_{(R_0,R_2)} = \sqrt{\left((0-0)^2 + (0-0)^2 + (0-90)^2\right)}\Big/1 = 90$$

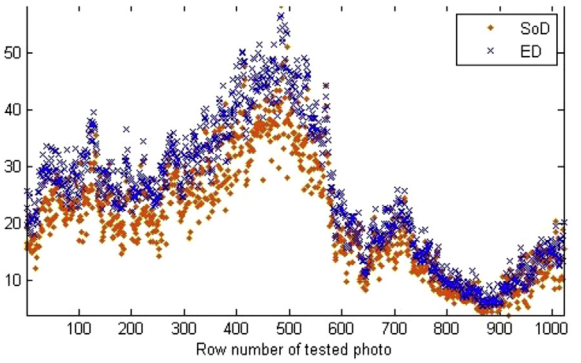
Nevertheless, both SoD and ED focus on the color similarity between the boundary adjacent pixels rather than the integrity and the smoothness of the whole image. Both matching metrics are limited to the pixels from just one row on the boundary. When fragments have similar color spaces or similar content, it will be harder to match the fragmentation boundaries correctly and the possibility of false concatenation will increase. Another serious problem is that the angles, the lines, and the sharp areas in image content are very likely to be falsely identified as fragmentation points since the object and color in those areas vary significantly, e.g., the boundaries of buildings, windows, forest or branch of trees.

Using picture "7.jpg" from 2007 DFRWS Carving Challenge as an example (Fig. 3), the values of SoD and ED between any two adjacent rows are calculated. The results are plotted against the row number in Fig. 4(a). Both SoD and ED vary considerably (from 5 to 55) across the whole picture, and exhibit high correlation in their respective measures of the similarity of adjacent rows. Both the values of SoD and ED are large in high-frequency areas of the image, i.e., from row 20 to row 590, where the image contents vary significantly due to the branches and leaves of the tree. If the image happens to be fragmented around row 400, it is likely that these two parts will not be considered as consecutive fragments. The same problem exists in the other high-frequency areas.

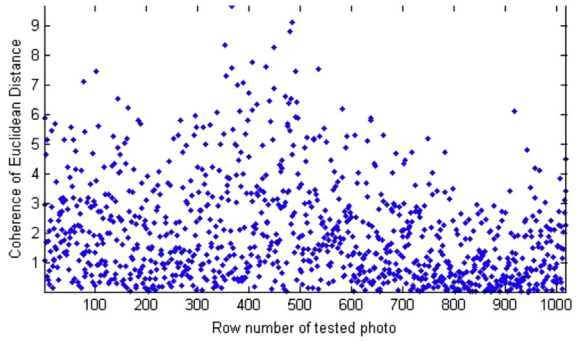
[1] The specific examples we used in the paper are to help readers to understand our observations/ideas, and should not be treated as evidence that SoD/ED is always a bad measure.

Fig. 2. RGB values for black color R_0 (0,0,0), gray color R_1 (30,30,30) and blue color R_2 (0,0,90). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 3. Photo **7.jpg** from DFRWS 2007 challenge.



(a) SoD and ED values versus row number.



(b) CED values versus row number.

Fig. 4. Values of different similarity metrics of **7.jpg**.

Clearly, each fragment can only belong to one file. Therefore, if a fragment is wrongly incorporated into a file, the error will be propagated and amplified in the subsequent carving steps for all related images.

New metric – coherence of Euclidean Distance (CED)

We now propose a new similarity metric to improve the accuracy and the reliability for adjacent fragments. In general, similar RGB values only occur in smooth areas of a picture, and the values may fluctuate drastically in sharp areas. Despite this fluctuation, the *variation pattern* of RGB values in a small range is similar and stable, regardless of whether the region is in a smooth or a sharp area. As an example, consider three consecutive rows; the variation between the first and the second row is similar to that between the second and the third row, both in smooth and sharp areas. As shown on Fig. 4(a), although the SoD and ED values from row 400 to 500 are higher than the values of other areas, they are close to each other within this range. By taking advantage of this characteristics, we introduce a new matching metric, Coherence of Euclidean Distance (CED), which defined as follows:

$$CED = \left| ED_{boundary} - ED_{nearby} \right| \tag{3}$$

The $ED_{boundary}$ is calculated from the RGB values of the pixels on both sides of the boundary using Equation (2). The ED_{nearby} is calculated from the RGB values of the pixels at the rows next to the boundary using the same equation. The pixels involved in the computation are illustrated on Fig. 5. Rather than evaluating the similarity between two adjacent row's pixels, CED focuses on the variation pattern to provide a more resilient method. Conceptually, $ED_{boundary}$ indicates the similarity between a partially recovered file and a fragment under evaluation. ED_{nearby} provides a guideline (locality information) on what $ED_{boundary}$ value would represent a possible adjacent fragment.

As a comparison, the CED values of the "7.jpg" are also plotted versus row number in Fig. 4(b). As evidenced by the graph, CED values of "7.jpg" exhibit much lower variability across the rows of the image, compared to the values of SoD and ED in Fig. 4(a). CED seems to provide a more reliable

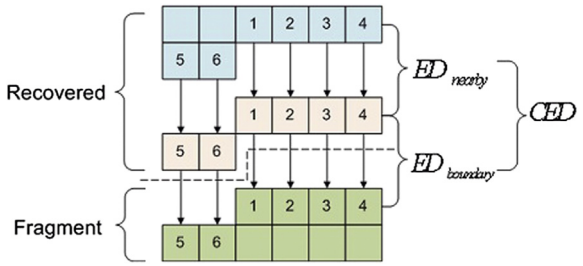


Fig. 5. Illustration of computing CED.

measure for identifying fragmentation points, even in the high-frequency areas of an image.

Fragmentation point detection using CED

We now illustrate how to use CED to detect fragmentation points. For ease understanding, we use the same example (the "7.jpg" in 2007 DFRWS Carving Challenge) as a test sample. This picture has a resolution of 720 × 1024 and the file size is 263,680 bytes with the size on disk of 266,240 bytes (65 blocks/clusters or 512 sectors on harddisk).

To simulate fragmented JPEG file, the original "7.jpg" is sliced into 4 pieces, numbered from 0 to 3, with every piece containing a random number of data blocks. To make the fragmented file displayable in Windows Photo Viewer, the file header and footer are kept in their original positions. Thus, the positions of the other two pieces (i.e. #1, #2) are switched to generate a fragmented image. As shown in Fig. 6, the permuted sequence is {#0, #2, #1, #3} in this test.

The values of SoD, ED and CED for the fragmented JPEG file were calculated and plotted versus the row number of the fragmented image. As shown in Fig. 7, the curve for CED values has 3 clear peaks. The positions of the peaks are precisely correlated to the row number of the 3 fragmentation points in the image.[2]

Fig. 6. The fragmented image of **7.jpg** in 2007 DFRWS Carving Challenge.

For SoD and ED, the curves are quite different, as shown in the left two subplots in Fig. 7. First, both the values of SoD and ED fluctuate significantly over the image. Second, based on the values of SoD and ED, it is quite difficult to identify the fragmentation points. For example, the values of SoD and ED for the fragmentation point between pieces #2 and #1 (at row 936) are relatively low, compared to the values of SoD and ED for the first fragmentation point and for the rows from 1 to 450. The same situation occurs around the fragmentation point between pieces #1 and #3. Therefore, two fragmentation points may be falsely eliminated using SoD or ED values.

We also tested the image (Fig. 6) with APF, which uses SoD as the similarity metric. The recovered image is shown in Fig. 8 – the recovery is not successful, as some pieces cannot be recovered. In particular, APF can precisely match the boundary between pieces #0 and #1, while it fails to match the boundaries between pieces #1 and #2 and between #2 and #3 due to the false elimination of fragmentation points. In the following section, we perform an evaluation to compare the reliability of CED relative to SoD and ED.

JPEG piece-based photo recovery

Based on the proposed metric, CED, we design a JPEG photo recovery algorithm based on JPEG pieces instead of JPEG data blocks. The flow diagram of the whole process is shown in Fig. 9. There are 6 stages in the process.

(1) *Identification of JPEG fragments.* During the first stage, we identify the data blocks in the forensic dump under investigation that are JPEG fragments.

(2) *Grouping of JPEG fragments into JPEG segments.* This step is performed by finding the standard markers of JPEG header and footer in the identified JPEG fragments and grouping the consecutive JPEG fragments between two markers as JPEG segments, to be processed and decoded as a whole in the next stage.

(3) *Decoding of JPEG segments.* The decoding converts the JPEG-encoded data blocks into RGB pixel image segment. The techniques for decoding JPEG segment without JPEG header information will be discussed in subsection 4.3.

(4) *Detection of fragmentation points* refers to the process of identifying fragmentation points. If no fragmentation point exists in the decoded image, the corresponding JPEG segment will be treated as an inseparable data

[2] In other cases, a fragmentation point may generate a pair of peaks when the boundary of two fragments does not occur exactly at the left-most pixel, but it would be easy to figure out the exact boundary based on the peaks.

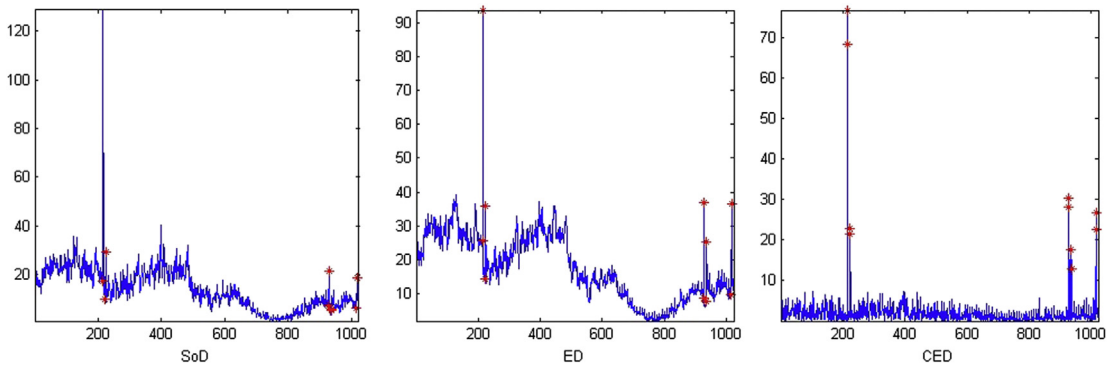


Fig. 7. Comparison of various similarity metrics for the fragmented image.

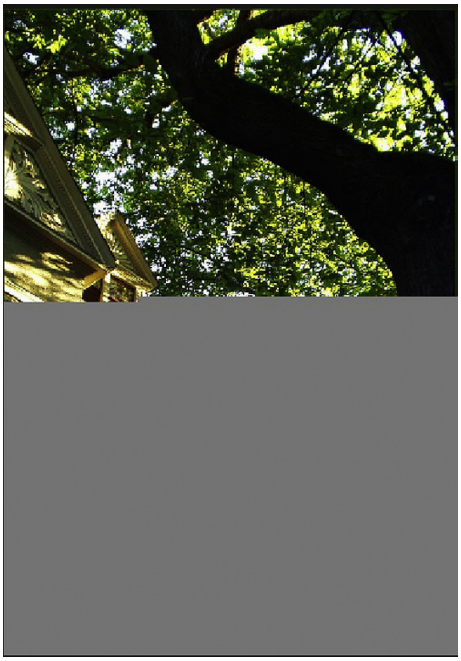


Fig. 8. The recovered image of "7.jpg" using APF.

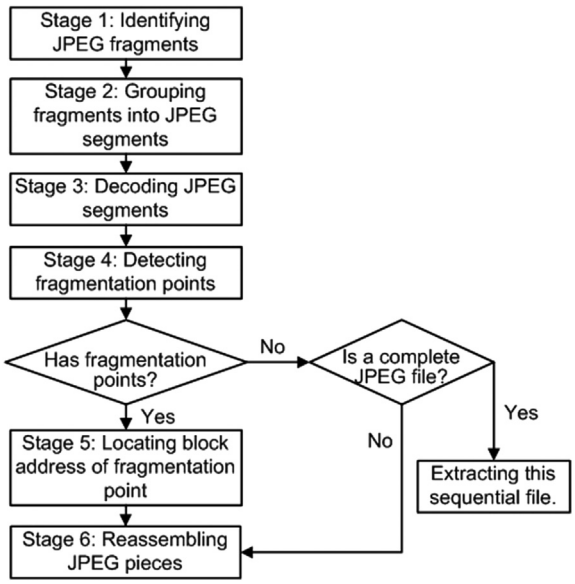


Fig. 9. The flow diagram of the proposed JPEG recovery algorithm.

piece in the subsequent steps. Otherwise, the row number of the fragmentation point is recorded and delivered to the next stage to get the corresponding address of the breaking point of the data blocks in physical storage.

(5) *Locating block addresses of fragmentation points* refers to the translation of the row numbers of the fragmentation points identified at Stage 4 to the physical block addresses of the breaking points of the corresponding data blocks. The JPEG segment will be divided into multiple JPEG pieces according to the block address of the breaking points. Each JPEG piece contains data from consecutive data blocks which belong to a single file.

(6) *Reassembly of JPEG pieces.* During the final stage, the algorithm attempts to put together the JPEG pieces to form valid, renderable images. We employ graph theory and heuristics to accomplish the task. We apply a look ahead technique to improve the efficiency and the accuracy of connecting adjacent JPEG pieces.

Stage 1: identifying JPEG fragments

There would be many different types of files stored in the same storage device. The first step is to identify all blocks that are parts of JPEG files. There are already some well-known approaches for solving this problem (e.g. McDaniel and Heydari, 2003; Li et al., 2005; Veenman, 2007). We can simply use existing tools such as Oscar (Karresand and Shahmehri, 2006) in this step.

Stage 2: grouping fragments into JPEG segments

In this stage, we sort the JPEG fragments according to their physical block addresses. Consecutive fragments are

put in the same group (segment) unless hitting a header or footer (see the example below). Fragments grouped in the same segment may still belong to different files and each segment will be decoded as a whole in the next stage. As an illustration, consider the scenario described by Table 1. There are 4 JPEG headers/footers in 351 consecutive JPEG data blocks. The first JPEG segment starts from block #0 to block #153, and the other two segments include blocks from #154 to #312 and blocks from #313 to #350, respectively. Data blocks starting from a JPEG header to a JPEG footer have a high chance being a complete JPEG file stored sequentially. JPEG headers and footers can be identified by special markers – '0xFFD8' for the header and '0xFFD9' for the footer.

Stage 3: decoding JPEG segments

Each JPEG segment containing multiple fragments will be decoded to RGB pixels during this stage. Standard JPEG decoders follows strict rules. An appropriate JPEG header is required to decode the compressed JPEG fragments. If a header exists in a segment, we use it to decode other fragments in the same segment (e.g. Segments #1 and #2 in Table 1). There exist segments without headers (e.g. Segment #3 in Table 1). We can try fitting in every header found in the data dump and try to decode the fragments. We assume that the decoding is successful if no warning message is given. Alternatively, we can follow the technique given in (Sencar and Memon (2009)) to construct a pseudo header.

Stage 4: detecting fragmentation points to identify fragmentation pieces

After the JPEG segments are decoded, CED values of the rows in decoded image segments are computed to evaluate the similarity distribution in the image segment. To improve the reliability of fragmentation point detection, we use local information to determine a dynamic threshold for making the decision. For that purpose, we keep track of the similarity between adjacent areas within decoded image, and retrieve the mean CED value as a local filtering factor. From our experiments, we found that the CED value of fragmentation point is usually several times higher than the mean CED value. The threshold used in this work is calculated based on the statistical results from empirical experiments.

If no significant peak of CED value is found, we conclude that the image segment contains no fragmentation point. The corresponding JPEG segment is considered a sequential piece and all the JPEG data blocks in the piece belong to the same file. The sequential piece is treated as a unit and is processed in Stage 6. Otherwise, if a fragmentation point exists, the corresponding row number will be identified by checking the distribution of CED over the image segment. In the next stage, the JPEG fragments near the fragmentation boundary is further analyzed to precisely locate the block address of the breaking point. After that, the JPEG segment is split into multiple pieces according to the locations of the breaking points.

Table 1
An example: grouping of JPEG fragments.

Block no.	Type	Segment no.
0	JPEG Header 1	# 1
1–153	JPEG data	# 1
154	JPEG Header 2	# 2
155–311	JPEG data	# 2
312	JPEG Footer 1	# 2
313–349	JPEG data	# 3
350	JPEG Footer 2	# 3

Stage 5: locating block address of fragmentation point

Normally, the size of a physical unit in a storage medium is small, such as 4096 bytes per cluster or block. A data block can contain compressed image data of approximately $N \times 1000$ pixels, where N is the height of a MCU (minimum coded unit, commonly consists of 8 or 16 pixels), depending on the compression parameters such as component number and sub-sampling size. If the image changes sharply in color or content, the decoded length of the JPEG data block can be shorter. However, nowadays, more and more device captured pictures hold high resolution for good quality. It is common to have a 10 Megapixels camera with a resolution of 3648×2736 and higher. As a result, an image row does not fit into a data block (usually 4KiB), and therefore the fragmentation boundary may be formed by several data blocks. Thus, when the row number of the fragmentation point is identified at the previous stage, it needs to be further analyzed to locate the breaking point of data blocks. Otherwise, there will be alignment errors during the reassembling stage.

After decoding, we calculate three numbers – the total number of data blocks in the JPEG segment (N_{blocks}), the number of rows in the decoded image segment (N_{rows}), and the row number of fragmentation point (fp_{row}). Since the row number of decoded image is proportional to the size of data, we can compute an approximate block address (fp_{block}) first for fragmentation point using the formula $N_{blocks}/N_{rows} = fp_{block}/fp_{row}$. Then, using binary search, we determine the upper-bound and lower-bound of fp_{block} to make sure that the data blocks around the fragmentation boundary are included.

Next, all the data blocks around fp_{block} are checked one by one to confirm the exact block address of the breaking point. In this phase, the CED value between two adjacent blocks is calculated. If the value of CED suddenly jumps, the corresponding block is the breaking fragmentation point.

Stage 6: reassembly of the JPEG pieces

When all the JPEG pieces are identified, the final task is to reassemble the pieces into JPEG files, i.e., to concatenate the JPEG pieces one by one correctly. We model this as a graph problem: each JPEG piece is represented as a vertex in a graph, and the challenge of reassembling JPEG files can be mapped to the problem of finding the shortest path from the header vertex to the footer vertex. The distance between two vertices is weighted by the likelihood that one piece follows another piece, which can be calculated using

the proposed similarity metric, CED. Therefore, the shortest path should be the permutation of JPEG pieces which has the smallest sum of CED values.

Specifically, we follow these steps. With a file header, we try to find the next correct piece from all available JPEG fragmentation pieces. We use the proposed CED metric to detect any fragmentation points. If a fragmentation point occurs between the header and the testing fragment piece, then this piece under consideration does not belong to this file and the next candidate will be considered. If no fragmentation point is detected, then this testing piece will be reassembled after this file header. The process is repeated until a file footer is found to recover the whole image. If no more correct pieces can be found, then the constructed segment will be treated as a partially truncated file, since the remaining part of this file may be deleted or overwritten.

Experimental results

To evaluate the performance of the proposed JPEG recovery algorithm, we carried out a series of experiments. We tested the reliability of the CED similarity metric relative to that of SoD and ED; we used two different image datasets. We investigated the overall success rate of the proposed JPEG recovery algorithm by using the fragmented JPEG files in a real digital camera storage medium containing 207 high resolution JPEG files. Finally, a specific experiment for heavily fragmented JPEG files is designed to test the performance of the proposed algorithm. As a reference tool, we used APF (version 3.2b). All implementation is done using Matlab v2013b and libjpeg (version 8b) is compiled as the JPEG decoder.

Reliability of CED

In this experiment, we prepared two datasets. The first dataset includes 1000 JPEG files downloaded from an Internet website, www.pdphoto.com. We resized all downloaded images to a reference resolution of 1024×768 and an average size of about 100 KB. The second dataset includes 100 high resolution pictures produced by several popular digital cameras. These images have a higher resolution of 3648×2048 or above, and the average file size is around 6 MB.

A false match occurs when we falsely match two non-adjacent fragments as adjacent fragments. To evaluate the reliability of CED in estimating matching likelihood, false match rate (FMR) is introduced and defined as the ratio of mismatched rows to the total rows in a picture. The pictures in the two datasets were analyzed and the average FMR using different metrics were summarized and compared.

The statistics of the false matches using CED, ED and SoD computed from the two datasets are listed in Table 2. For the dataset of 1000 low resolution images, the total number of rows is $768 \times 1000 = 768,000$. Using CED as the similarity metric, there are 132,529 rows that may be falsely matched. Thus, the corresponding FMR is $FMR_{CED} = 132,529 \div 768,000 = 17.26\%$. If SoD or SoD is used as a matching metric, the numbers of false matches increase to 381,112 and 491,775, respectively and a corresponding FMR of 49.62% and 64.32%. There are 46 files for which there is no false match if CED is used. As a comparison, for all the 1000 files, there is at least one false match if we use ED or SoD. This result indicates that CED is more reliable than ED and SoD in color similarity evaluation and matching likelihood estimation.

When the experiment is performed on the dataset containing 100 high resolution images, the performance of CED is much better than ED and SoD. For the 100 high resolution images, the total number of rows is $2048 \times 100 = 204,800$. As shown in Table 2, there are only 1829 false matches using CED, while the number of mismatched rows using SoD and ED are 115,142 and 128,805, separately. The performance of using CED improves substantially from $FMR_{CED} = 17.25\%$ to 0.89% while for the other two metrics, the improvement is much smaller. This shows that CED can take advantage of high resolution to compute a more accurate similarity than the other two metrics. From the results, it can be concluded that the reliability of CED can be further improved with high resolution images since high quality images can provide more pixels and further enhance the stability of the curve for CED values. Since the trend is to have more high quality images, CED should be adopted in future JPEG carving tools.

Carving JPEG files from digital camera storage medium

In this experiment, the devices under test include a digital camera for taking photos and a 4 GB SD card as the storage medium. The digital camera has a resolution of 3648×2736 with an average file size of 5 MB. Initially, the data on the SD card is wiped off on a PC and the SD card is formatted as FAT32 with the cluster size of 4KiB. Then, the dataset for test is prepared by repeatedly taking photos and randomly deleting some of them to simulate a normal user's behavior. After the operations, a storage dump of the SD card is created on a PC. Finally, the file system information of this storage dump is removed and only the data

Table 2
The number of false matches using CED, SoD and ED.

Testing	1000 low resolution			100 high resolution		
Photos	(1021×768)			(3648×2048)		
Similarity Metric	CED	ED	SoD	CED	ED	SoD
False Matches	132,529	381,112	491,775	1829	115,142	128,805
False Matching Rate	17.26%	49.62%	64.03%	0.89%	56.22%	62.89%
Files with False Match	954	1000	1000	59	100	100

segments are kept to test the proposed JPEG recovery algorithm, which will try to recover as many JPEG files as possible without the help of file system information.

The storage dump under test contains 207 JPEG files of about 1 GB in total. Due to the frequent operations of deletion, some of the photos are fragmented (or even heavily fragmented). We identified 120 fragmented JPEG files and 20 of them have a fragmentation point in their JPEG headers. The broken headers of the 20 JPEG files have fewer than 50 consecutive data blocks and do not provide enough decoding information. Therefore, we focus on the carving problems of fragmentation point detection and JPEG pieces matching. We exclude from consideration the 20 JPEG files with broken JPEG header and treat their remnants as noise during the carving analysis and recovery procedures. Note that APF cannot solve the problem of broken JPEG header too.

To summarize, there are 87 sequential files, 100 fragmented files and 20 fragmented files with broken JPEG header in the storage dump under test. In this experiment, the storage dump is analyzed by our proposed algorithm to recover 87 sequential files and 100 fragmented files. As a baseline, we also analyze the storage dump with APF.

As shown in Table 3, 187 files are classified according to the number of pieces. The number of recovered JPEG files (a recovered JPEG file is one that is 100% correct after recovery) by APF and our proposed algorithm are listed in the second column and the third column respectively. For those 87 sequential JPEG files, both APF and our proposed algorithm can easily and successfully recover all of them. For the 100 fragmented JPEG files, the results are different. Our proposed algorithm can recover 97 photos, while APF can recover only 78 photos. In other words, our algorithm has a successful rate of 97%, which is about 20 percentage points higher than that of APF (78%) in this test.

We also manually analyzed the reasons for the failed recovery by our algorithm. One file has a very small JPEG piece which consists of only 3 data blocks. After decoding, this small piece is translated into a tiny image segment which is far shorter than the width of one row in the photo. Therefore, due to the limited number of pixels for computing the CED value, we are not able to connect it back to the original photo. Even worse, this small piece is falsely concatenated to another JPEG file causing the failure of not able to recover this JPEG file as well. For that reason, APF can recover two 4-piece files, while our algorithm can only recover one. In the last case, the content of the photo changes dramatically in the horizontal direction, such as windows or handrails with reflecting lights, and a false peak based on CED values was generated. Thus, the fragmentation points of this file are identified wrongly and the file cannot be carved correctly.

Table 3
Recovery of 187 JPEG files in SD card storage dump.

Category	JPEG files	Proposed algorithm	APF
sequential file	87	87	87
2-piece file	79	78	65
3-piece file	18	17	11
4-piece file	2	1	2
6-piece file	1	1	0

Table 4
Recovery of the 184 heavily fragmented JPEG files.

Category	JPEG files	Proposed algorithm	APF
3-piece file	109	96	66
4-piece file	75	61	32

Carving heavily fragmented JPEG files

To further investigate the capability of the proposed algorithm for carving heavily fragmented files, another experiment was performed with an independent dataset including 184 heavily fragmented JPEG files, 99 of which are 3-piece JPEG files and 75 of which are 4-piece JPEG files. In this test, the fragmentation points and fragments were generated randomly by a computer program to simulate the heavily fragmented scenario.

As shown in Table 4, the successful rate of recovering heavily fragmented JPEG files using the proposed algorithm is much higher than that using APF. For 3-piece JPEG files, about 88.1% of the files can be recovered using the proposed algorithm, while APF can recover about 60.6% of them. For 4-piece JPEG files, the successful rate of the proposed algorithm is slightly decreased to about 81.3%, while the success rate of APF drops significantly to 42.7%.

Conclusions

In this paper, we proposed a new JPEG carving algorithm which is able to recover fragmented (even heavily fragmented) JPEG files without using any information in the file system. The performance of our algorithm outperforms APF, one of the commonly used tool for photo forensics. The robustness of our algorithm stems from a new similarity measure (CED) which can be used to identify fragmentation points more accurately. However, there are still a few open problems to be resolved in the future. If the content of the picture contains a sharp change in the horizontal direction which is the fragmentation point, our method fails to identify it. Also, further speeding up our algorithm is also desirable.

Acknowledgments

This work was supported in part by Projects of International Cooperation and Exchanges NSFC (61361166006), China, NSFC/RGC Joint Research Scheme (N_HKU 729/13), Hong Kong, and China State Scholarship Fund (201506785014).

References

APF. Adroit photo forensics 2013. 2013. http://digital-assembly.com/.
Cohen MI. Advanced carving techniques. Digit Investig 2007;4(3):119–28.
De Bock J, De Smet P. Jpgcarve: an advanced tool for automated recovery of fragmented jpeg files. Information forensics and security. IEEE Trans 2016;11(1):19–34.
Foremost. Foremost. 2010. http://foremost.sourceforge.net/.
Garfinkel SL. Carving contiguous and fragmented files with fast object validation. Digit Investig 2007;4(Suppl.):2–12.

Guo H, Xu M. A method for recovering jpeg files based on thumbnail. In: Control, automation and systems engineering (CASE), 2011 international conference on. IEEE; 2011. p. 1–4.
Karresand M, Shahmehri N. Oscar file type identification of binary data in disk clusters and ram pages. In: Security and privacy in dynamic environments. Springer; 2006. p. 413–24.
Karresand M, Shahmehri N. Reassembly of fragmented jpeg images containing restart markers. In: 2008 European conference on computer network defense; 2008.
Li B, Wang L, Sun Y, Wang Q. Image fragment carving algorithms based on pixel similarity. In: Multimedia information networking and security (MINES), 2012 fourth international conference on. IEEE; 2012. p. 979–82.
Li WJ, Wang K, Stolfo SJ, Herzog B. Fileprints: identifying file types by n-gram analysis. In: Information assurance workshop, 2005. IAW'05. Proceedings from the sixth annual IEEE SMC. IEEE; 2005. p. 64–71.
McDaniel M, Heydari MH. Content based file type detection algorithms. In system sciences, 2003. In: Proceedings of the 36th annual Hawaii international conference on. IEEE; 2003. p. 10.
Memon N, Pal A. Automated reassembly of file fragmented images using greedy algorithms. Image Process IEEE Trans 2006;15(2):385–93.
Pal A, Shanmugasundaram K, Memon N. Automated reassembly of fragmented images. In: 2003 IEEE international conference on acoustics, speech, and signal processing4; 2003. http://dx.doi.org/10.1109/ICASSP.2003.1202747. IV–732–5.
Pal A, Memon N. The evolution of file carving. Signal Process Mag IEEE 2009;26(2):59–71.
Pal A, Sencar HT, Memon N. Detecting file fragmentation point using sequential hypothesis testing. digital investigation 2008;5:S2–13.
Poisel R. Multimedia file carving. 2012. https://github.com/rpoisel/mmc.
Poisel R, Tjoa S. A comprehensive literature review of file carving. In: Availability, reliability and security (ARES), 2013 eighth international conference on; 2013. p. 475–84. http://dx.doi.org/10.1109/ARES.2013.62.
Richard III GG, Roussev V. Scalpel: a frugal, high performance file carver. In: DFRWS; 2005.
Sencar H, Memon N. Overview of state-of-the-art in digital image forensics. Algorithms Archit Inf Syst Secur 2008;3:325–48.
Sencar HT, Memon N. Identification and recovery of jpeg files with missing fragments. Digit Investig 2009;6:S88–98.
Shanmugasundaram K, Memon N. Automatic reassembly of document fragments via context based statistical models. In: Computer security applications conference, 2003. Proceedings. 19th annual. IEEE; 2003. p. 152–9.
Sharma G. Digital color imaging handbook. CRC Press; 2002.
Veenman CJ. Statistical disk cluster classification for file carving. In: Information assurance and security, 2007. IAS 2007. Third international symposium on. IEEE; 2007. p. 393–8.
Ying HM, Thing VL. A novel inequality-based fragmented file carving technique. In: Forensics in telecommunications, information, and multimedia. Springer; 2011. p. 28–39.