ENGINEERING Information Technology & Electronic Engineering
https://www.academax.com/EITEE; https://www.engineering.org.cn
https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=11344065
E-mail: eng.itee@zju.edu.cn

Engineering Journals
Available on IEEE Xplore®

Research Article
https://doi.org/10.1631/ENG.ITEE.2025.0140

Image fragment carving based on DCT semantics and a adjustment factor

Binglong LI[1✉], Shilong YU[1✉], Yong ZHAO[2], Yifeng SUN[1], Chaowen CHANG[1], Qingxian WANG[1]

[1]*Information Engineering University, Zhengzhou 450001, China*
[2]*SDIC Intelligent (Xiamen) Information Co., Ltd., Xiamen 361000, China*

Abstract: The recovery of evidence from fragmented image files is a prominent research focus in the field of file carving. To address image fragment reassembly, this paper analyzes the Joint Photographic Experts Group (JPEG) image structure and proposes a fragment connection weighting algorithm based on discrete cosine transform (DCT) semantic features, along with a weight adjustment factor that leverages image compression characteristics. By integrating these components, the algorithm effectively determines the fragment sequence in JPEG files, and a practical carving algorithm is designed. Experiments conducted on disk and memory demonstrate that the adjustment factor-based algorithm outperforms the DCT-only method in identifying true best matches (reducing false positives). Disk experiments achieve an average carving precision of 94.4%, surpassing existing methods, while memory experiments validate the feasibility of the approach, along with a theoretical analysis of failure scenarios caused by interference from software such as Windows Picture Viewer.

Key words: Discrete cosine transform; Fragmented image files; Memory media

Unedited

1 Introduction

File carving is an important technique for evidence extraction in digital crime forensic investigations (Ramli et al., 2021b; Guzhov and Wirth, 2025; Lu et al., 2025). However, due to the inherent block-based storage characteristics of hard disk media, file operations such as addition, deletion, and modification can easily lead to file fragmentation, posing significant challenges for evidence recovery. With the development of anti-forensic techniques, digital criminals often conceal evidence by deleting files or even storing them in fragmented forms before law enforcement personnel can obtain image file evidence, thereby exacerbating the degree of file fragmentation on hard disk storage media. Additionally, memory forensics technology is receiving increasing attention because physical memory contains a large amount of digital crime evidence, and research indicates that data fragmentation in memory is even more severe than on disk (van Baar et al., 2008). In digital forensic investigations, it is necessary to find different types of crime-related evidence such as emails, documents, system logs, and multimedia image files from both disk media and memory media. Among these, the extraction of evidence from fragmented image files is currently a hot research topic in the field of fragmented file carving (Ali et al., 2018; Mullan et al., 2019; Fabrizzio et al., 2020; Shi et al., 2023).

Currently, several fragment carving methods have been proposed to extract fragmented image files from disk storage media (Ramli et al., 2021a). The key to image fragment carving methods is the fragment connection weighting algorithm, and most of the current fragment connection weighting algorithms, such as sum of differences (SoD) and Euclidean distance (ED), focus on using the color difference between boundary pixels to determine the best candidate fragment (Tang et al., 2016; Wu et al., 2019). These methods have limitations: the minimum color difference between fragments only indicates the weakest semantic relationship between two rows of pixel data at the fragment edges, but it cannot necessarily determine that these two fragments have a genuine connection (Tang et al., 2016).

Furthermore, existing image fragment carving methods rarely target physical memory media. Extracting and carving image data from physical memory are extremely important for reconstructing digital crime scenes. Research shows that the file fragments within memory images are highly fragmented,

✉ lbl2017@163.com
✉ ysl1327244332@163.com
Binglong LI, http://orcid.org/0000-0003-3421-5432
Shilong YU, http://orcid.org/0000-0001-6339-4308
CLC number: TP309.3

Received Nov. 16, 2025; Revision accepted Mar. 6, 2026; Crosschecked Mar. 16, 2026

© The Authors 2026. Published by Zhejiang University Press Co., Ltd. This is an open access article distributed under the terms of the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

making it impossible to reconstruct fragmented files using image file carving methods designed for disk media (van Baar et al., 2008). Wu et al. (2019) proposed an image fragment carving algorithm using memory structure chains; this method primarily uses file metadata left by the operating system on memory media for file reconstruction and does not study the connection between image fragments in memory.

Currently, many researchers use discrete cosine transform (DCT) to address challenges in the image domain, such as image fusion, texture recognition, and object recognition (Bharati et al., 2004). However, its application remains limited in the specific area of image fragment carving, particularly within the context of fragment connection weighting algorithms. The Joint Photographic Experts Group (JPEG) image encoding process uses DCT to convert image data—for instance, the intensity values of 64 pixels within an 8×8 block—into corresponding DCT coefficients, which are subsequently stored on the storage medium. While the DCT values for different pixel blocks exhibit variations across different regions of an image, they consistently demonstrate localized characteristics. Consequently, we analyze the features of the DCT data generated during the image decoding process and introduces a novel semantic weighting algorithm for fragments based on DCT. Furthermore, to mitigate scenarios where DCT semantic features between fragments appear plausible but the underlying DCT data syntax may be inconsistent or invalid, we design a DCT semantic weight adjustment factor. This factor is derived from the compression properties inherent in JPEG image data entropy encoding and the relationship between the lengths of raw DCT data and their corresponding JPEG entropy-encoded data, thereby enhancing the inter-fragment semantic weight. Experimental results on a set of 2000 fragments demonstrate that the inter-fragment DCT semantic weighting algorithm, incorporating an adjustment factor, successfully identifies the single correct candidate connecting fragment. In contrast, relying solely on the inter-fragment DCT semantic weight produces multiple false candidate fragments. The algorithm's performance is evaluated through experiments on both disk media and memory media. Results from disk media indicate that the fragment connection weighting algorithm accurately identifies genuine candidate fragments and facilitates successful image data carving. Moreover, when compared to other algorithms, it achieves an average carving accuracy of 94.4%. Experiments on memory media confirm the feasibility of performing image fragment carving within such environments and include a theoretical analysis explaining instances where the carving process may fail.

2 Related works

2.1 JPEG file format

The JPEG was a group jointly established in 1986 by the Consultative Committee for International Telephony and Telegraphy (CCITT) and the International Organization for Standardization (ISO). This group is responsible for developing coding standards for static digital images. Committed to standardization, the group developed and refined a digital image compression encoding method for continuous-tone, multilevel grayscale still images, known as the JPEG algorithm, which was established as the international general standard for digital image compression (Kornblum, 2008). The JPEG standard defines four compression-encoding modes: sequential mode, progressive mode, lossless mode, and hierarchical mode. In sequential mode, JPEG file storage is completed through a single scan for image encoding and decoding; in progressive mode, encoding and decoding require multiple scans, with the effect progressing from coarse to fine; in hierarchical mode, the image is encoded and decoded at multiple spatial resolutions. Sequential mode is by far the most commonly used JPEG image encoding mode. The encoding and decoding processes for this mode are shown in Fig. 1.

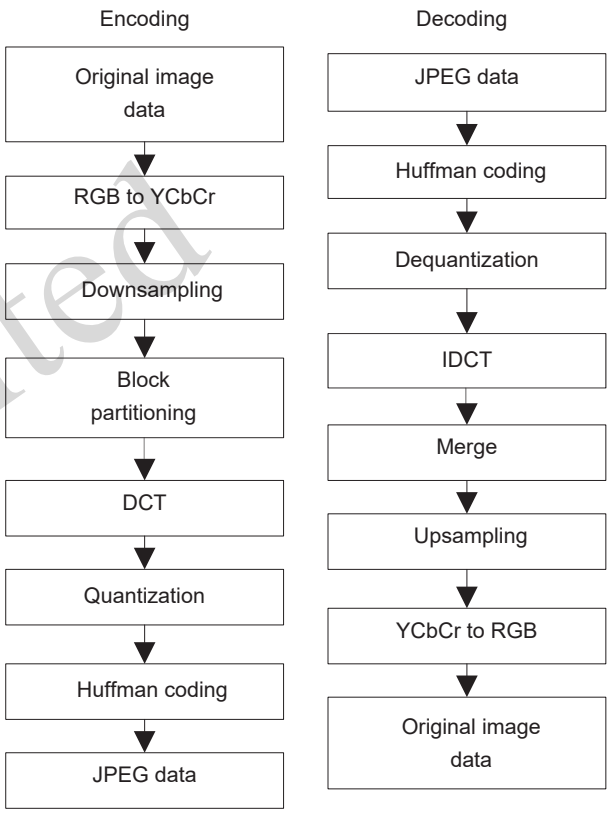

Fig. 1 JPEG encoding and decoding process (RGB: red/gree/blue; IDCT: inverse discrete cosine transform. YCbCr is a color model that separates an image into three components: luminance, blue-difference chroma, and red-difference chroma)

As shown in Fig. 1, the encoding and decoding processes of the JPEG compression algorithm are inverse processes. The encoding process, for example, mainly consists of six steps (Dabbaghchian et al., 2010). Step 1 involves color space conversion, that is, converting RGB model data representing color to YCbCr model data. The JPEG file interchange format (JFIF) file format uses the YCbCr color space for encoding color information. This color model consists of three color components: Y, Cb, and Cr. The Y component (luminance) approximates brightness information, while the Cb and Cr chrominance components approximate color information. Step 2 is downsampling. To exploit the human eye's higher sensitivity to luminance changes compared to chrominance changes,

Unedited

the chrominance components can be subsampled to have different resolutions. When chrominance subsampling is performed, the chrominance resolution is reduced by a factor of two, either horizontally, vertically, or both horizontally and vertically. Step 3 comprises image block partitioning. This step involves dividing the image into non-overlapping small blocks of size 8×8. During block partitioning, the Y, Cb, and Cr components are divided into 8×8 blocks and processed separately. Step 4 is DCT, which applies a two-dimensional DCT to each small block to obtain transform coefficients in the frequency domain. Step 5 involves data quantization, where the DCT coefficients are quantized to discard the least important information. Step 6 is Huffman coding, where the quantized transform coefficients are reordered via a zigzag scan within each transform block and passed through the Huffman coding algorithm to generate the JPEG data. JPEG data are essentially compressed data stored on the medium, generated through the aforementioned six steps: color space conversion, downsampling, block partitioning, DCT, quantization, and Huffman coding. For decoding, the above steps are performed in reverse order.

The JPEG standard specifies the JFIF file format as the storage format for JPEG data streams on storage media. A JFIF file consists of a series of markers or marker segments used to distinguish and identify image data and its related information. The JFIF file header, through relevant markers, defines image details, compression mode, quantization tables, Huffman tables, and other information needed for decoding. The data between the Start of Scan (marked as FFDA) and the end of the file primarily consists of the image pixel data stream. Within this data stream, information can be divided into consecutive segments of minimum coded unit (MCU) data. MCUs are arranged from left to right and top to bottom in the data stream. Each MCU is further divided into several data units, the size of which is always 8×8. JPEG divides the image into three sub-images: Y, Cr, and Cb. After color space conversion, downsampling, block partitioning, DCT, quantization, and Huffman coding, the encoded data are stored. Within each MCU, the data order is Y, Cr, Cb. If a color component has multiple data units, they are arranged from left to right and top to bottom(Singh and Gupta, 2016).

From a file carving perspective, JPEG data are divided into file header data and image pixel data. The file header data contains image-related information, and the image pixel data stream consists of entropy-encoded compressed data blocks (MCUs). Each MCU consists of a certain number of blocks from each color component. The size of an MCU is determined by the sampling factors of the different color components, namely the horizontal and vertical sampling factors for the Y, Cr, and Cb components. Common MCU sizes are typically 8×8, 16×8, or 16×16 pixels, with corresponding color component sampling ratios of $1:1:1$, $2:1:1$, or $4:1:1$. Here, $1:1:1$ means $(1 \times 1):(1 \times 1):(1 \times 1)$, $2:1:1$ means $(2 \times 1):(1 \times 1):(1 \times 1)$, and $4:1:1$ means $(2 \times 2):(1 \times 1):(1 \times 1)$. Denoting the maximum horizontal sampling factor among the three components as $H_{\max}$ and the maximum vertical sampling factor as $V_{\max}$, the width of a single MCU matrix is $H_{\max} \times 8$ pixels, and the height is $V_{\max} \times 8$ pixels. Each MCU is divided into several 8×8 data units, and the number of data units per MCU is $H_{\max} \times V_{\max}$. Thus, an 8×8 MCU consists of one Y, one Cb, and one Cr color component each; a 16×8 MCU consists of four data units, specifically two Y, one Cb, and one Cr; a 16×16 MCU consists of six color units, specifically four Y, one Cb, and one Cr.

2.2 Current state of image fragment carving technology

File carving has become one of the important techniques for evidence acquisition in digital crime forensic investigations. Currently, many popular forensic tools, such as EnCase, Forensic Toolkit (FTK), and The Sleuth Kit (TSK), include file carving capabilities(Richard and Roussev, 2005). However, most of these forensic tools are only effective for contiguously stored files on hard disk storage media and have limitations in the context of fragmented file carving.

The key issue in fragmented file carving lies in determining the fragment connection weighting algorithm. both the algorithm itself and the process for calculating weights depend on the file type that the file carver should recover. Shanmugasundaram and Memon (2003) proposed a fragmented file carving method based on a contextual statistical model, which primarily uses semantic relationships (such as text) between document fragments to determine connection weights. However, this method is unsuitable for fragmented carving files with structured formats such as JPEG. Garfinkel (2007) proposed a bifragment gap carving (BGC) algorithm and used a JPEG validator for verification; yet, this algorithm is limited to file carving divided into two fragments. Memon and Pal (2006) proposed applying a greedy algorithm for fragment reassembly, suggesting three mechanisms, namely, pixel matching (PM), DCT, and median edge detection (MED), to identify connecting fragments. This method calculates fragment connection weights based on edge pixels and only provides experimental results for bitmap (BMP) file fragments. To overcome the limitation on the number of fragments in the BGC algorithm, Pal et al. (2008) proposed a greedy algorithm optimized method based on sequential hypothesis testing; however, this method still uses the DCT method based on pixel values to calculate the connection weights between adjacent fragments. Wu et al. (2019) analyzed the impact of similarity patterns on fragment connectivity, suggesting that horizontal similarity is almost unhelpful, and proposed an improved MED method based on left and right diagonal pixels. Tang et al. (2016) proposed a new similarity measure method named cumulative ED (CED), based on the principle of local similarity between two rows of pixels at fragment edges. All these methods judge the connection relationship between fragments based on pixel similarity semantic. However, when processing fragments containing invalid information, these fragment connection weighting algorithms may incorrectly identify them as fragments with the best pixel color semantics. Although this might not trigger syntax errors, it inevitably leads to carving errors.

Therefore, some scholars have studied fragmented file carving methods based on the syntactic features of JPEG files. Karresand and Shahmehri (2008) proposed an image fragment carving algorithm based on "restart markers," aiming to han-

Unedited

dle bitstream errors caused by file corruption or unreliable transmission. In practice, most JPEG files do not have such restart markers. Moreover, even files with restart markers cannot guarantee a marker in every image fragment; thus, this method may still fail. Mohamad and Deris (2009) studied the detection of damaged data within the define Huffman table (DHT) segment of JPEG files and proposed a fragmentation point detection method based on DHT length features. Sencar and Memon (2009) used DHT structure information to propose a fragmentation detection algorithm based on bit patterns, but this method cannot handle the problem of fragment detection for files with the same DHT structure (e.g., pictures taken by the same camera).

Li Q et al. (2011) proposed an image damage localization algorithm using the libjpeg library. This approach does not rely on local pixel similarity; instead, it detects potential damage locations (fragmentation points) within images by identifying abnormal distribution characteristics of direct current (DC) and alternating current (AC) values in the DCT domain. However, this algorithm is incapable of determining whether a given image fragment originates from a specific file. Uzun and Sencar (2020) introduced an isolated file fragment reconstruction technique based on DCT domain features, specifically, a robust segmentation method for JPEG entropy-encoded data segments. This method aims to address the challenge of image file carving in scenarios where file header fragments are missing or corrupted. Separately, de Bock and e Smet (2015) developed a novel approach for reconstructing fragmented JPEG files. Their method relies primarily on detecting MCU parsing syntax errors and pixel errors within the JPEG image stream to identify fragment end positions. A key limitation, however, is its inability to locate the start position of the subsequent image fragment. For cases involving missing file header fragments or absent critical file header information, Wu et al. (2018) proposed an image width estimation method founded on MCU analysis. Furthermore, Durmus et al. (2017) addressed the loss of file header fragments by introducing a fragment extraction and reconstruction technique based on camera hardware fingerprint features, specifically camera sensor noise patterns. A significant constraint of this method is its dependence on prior knowledge of the specific camera's hardware fingerprint characteristics. Azhan et al. (2022) presented an image fragment feature recognition algorithm using error level analysis. This technique can identify distinctive error pattern features within 8×8 pixel blocks of image fragments; however, the researchers did not extend their investigation to the connection relationships between different fragments. Collectively, while the aforementioned research efforts attempt to leverage characteristics inherent in the JPEG compressed image data stream, none of them use image DCT domain features to specifically investigate the problem of fragmented file carving.

In recent years, memory forensics has become a primary technique for malware detection, and extracting image data based on physical memory images has also become a key technique for obtaining evidence in digital investigations. However, research on obtaining evidence based on physical memory images is relatively scarce. van Baar et al. (2008) analyzed the data fragmentation problem in memory images, and their results show that the degree of fragmentation in memory is much higher than that on hard disk media. Li BL et al. (2021) proposed an image data carving method based on memory structure metadata, but did not study the connection relationships between image fragments on memory media. Currently, no research results have been reported on image data carving methods specifically designed for physical memory images.

Therefore, departing from traditional fragment connection weighting algorithms that rely on pixel differences, this paper designs a novel inter-fragment semantic weighting algorithm based on DCT. Furthermore, to prevent scenarios where DCT semantic features between fragments appear coherent while the underlying DCT data syntax may be inconsistent, we develop a DCT semantic weight adjustment factor. This factor is constructed by leveraging the compression characteristics of JPEG image data entropy encoding and the length relationship between DCT data and JPEG entropy-encoded data, thereby enhancing the semantic weighting between fragments.

3 Fragment connection weighting algorithm based on DCT semantics and adjustment factor

3.1 DCT semantic feature extraction

The JPEG encoding process uses the DCT to encode image pixel data (e.g., an 8×8 pixel block) into corresponding DCT coefficients. Within the DCT domain, each coefficient $F(u, v)$ has a linear correspondence with all spatial pixel values. Let $f(x, y)$ represent the spatial pixel values. Then, $F(u, v)$ is obtained by applying DCT to transform $f(x, y)$ into the frequency domain for an 8×8 block. DCT is defined as follows:

$$\begin{aligned} F(u,v) =& \frac{1}{4}c(u)c(v)\sum_{y=0}^{7}\sum_{x=0}^{7} f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right) \\ & \cdot \cos\left(\frac{(2y+1)v\pi}{16}\right), \\ & u = 0, 1, \cdots, 7,\ v = 0, 1, \cdots, 7, \end{aligned} \tag{1}$$

where $c(u)$ and $c(v)$ are the normalization coefficients of DCT, defined as

$$c(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0, \\ 1, & u = 1, 2, \ldots, 7, \end{cases}$$

$$c(v) = \begin{cases} \frac{1}{\sqrt{2}}, & v = 0, \\ 1, & v = 1, 2, \ldots, 7. \end{cases}$$

For the specific case where $u = 0$, $v = 0$, Eq. (1) simplifies to

$$F(0,0) = \frac{1}{8}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y). \tag{2}$$

Note that DCT can quickly and effectively remove redundancy between adjacent pixels and exhibits energy compaction characteristics. In the DCT coefficient matrix, the coefficient at the top–left corner is called the DC coefficient, and the remaining coefficients are called AC coefficients. The DC coefficient shows the average intensity of the 8×8 pixel block. From Eq. (1), it can be seen that the DC coefficient is closely related to the average of the 64 pixel values in the 8×8 block

Unedited

in the pixel domain. The value of each AC coefficient reflects the change in grayscale values in different directions. Furthermore, AC coefficients closer to the top–left corner reflect low-frequency grayscale changes, while those in the bottom–right corner reflect high-frequency grayscale changes. That is, most of the image information is concentrated in the DC coefficient and the nearby low-frequency spectrum, whereas the high-frequency spectrum farther away from the DC coefficient contains almost no image information or may even contains only noise. Therefore, using DCT coefficients as features for image blocks is reasonable.

Therefore, we use the zigzag scanning method (as shown in Fig. 2) to convert the DCT coefficients into a row-wise sequence, which serves as the feature vector for the corresponding 8×8 pixel block. To reduce dimensionality, the DCT coefficient feature vector sequence is truncated by a truncation factor k. The resulting sequence serves as the feature descriptor for an image block.

$$\boldsymbol{a}_{i,j} = (a_1, a_2, \cdots, a_k),\ k = \lfloor f_{\mathrm{t}} \cdot 8^2 \rfloor, \tag{3}$$

where $\boldsymbol{a}_{i,j}$ represents the feature vector of an 8x8 image block at row i, column j in the image. f_t denotes the DCT truncation factor, a hyper-parameter in the range $(0, 1]$ that controls the dimensionality of the feature vector by determining the retention ratio of the original 64 DCT coefficients. Considering the spatial correlation between adjacent image pixel blocks, we can use their DCT coefficient feature vectors to calculate the connection similarity between adjacent pixel blocks. We define Eqs.(4) and (5) to represent the connection similarity between adjacent pixel blocks as follows:

$$\begin{aligned} S_{\mathrm{v}}(\boldsymbol{a}_{i,j}, \boldsymbol{b}_{i+1,j}) &= |\boldsymbol{a}_{i,j} - \boldsymbol{b}_{i+1,j}| \\ &= |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_k - b_k|, \end{aligned} \tag{4}$$

$$\begin{aligned} S_{\mathrm{h}}(\boldsymbol{a}_{i,j}, \boldsymbol{c}_{i,j+1}) &= |\boldsymbol{a}_{i,j} - \boldsymbol{c}_{i,j+1}| \\ &= |a_1 - c_1| + |a_2 - c_2| + \cdots + |a_k - c_k|, \end{aligned} \tag{5}$$

where $S_{\mathrm{v}}(\boldsymbol{a}_{i,j}, \boldsymbol{b}_{i+1,j})$ represents the connection similarity between vertically connected blocks; and $S_{\mathrm{h}}(\boldsymbol{a}_{i,j}, \boldsymbol{c}_{i,j+1})$ represents the connection similarity between horizontally connected blocks. Smaller values of S_{v} and S_{h} indicate a greater degree of connection between the corresponding image blocks.

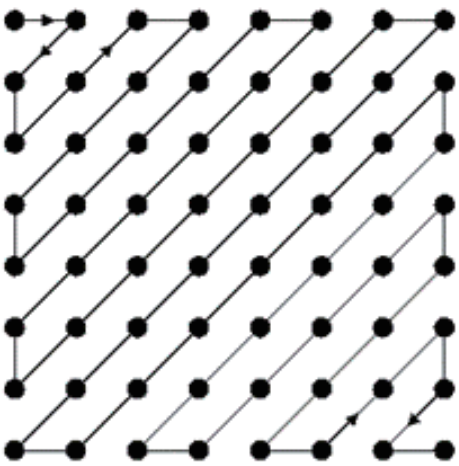

Fig. 2 Zigzag scanning order

Conversely, larger values suggest a lower likelihood of adjacency. This paper explores the connection similarity between two fragments, focusing primarily on the connection similarity between vertically adjacent blocks, as defined in Eq. (4).

Subsequently, we conduct a statistical analysis on the DCT coefficients of 2000 JPEG image fragments. The results reveal that the variance of high-frequency AC coefficients is merely 1.2% of that of low-frequency coefficients, showing no significant difference from the distribution of random noise (t-test, $p > 0.05$. p denotes the p-value, indicating no statistical significance at the 0.05 level). Note that while high-frequency information may contain local details in texture–dense images, severe quantization distortion in fragmented scenarios is prone to causing false matches, so it is not incorporated into the feature vector for the time being. For specific high-resolution distortion-free images, the high-frequency feature dimension can be extended by adjusting the truncation factor k.

3.2 Adjustment factor construction

The DCT coefficients at the fragment edges are generated by decoding the encoded data of the fragment edges. Each DCT coefficient block represents 8×8 pixels, and each pixel requires 3 bytes for storage. As shown in Fig. 1, image data are compressed and encoded data are generated through steps such as color space conversion, block partitioning, quantization, and Huffman coding. During the decoding process, each DCT coefficient block shows one-to-one correspondence to a data string of a specific length at a certain position in the JPEG image data. Suppose there are two DCT coefficient blocks $\boldsymbol{a}_{i,j}$ and $\boldsymbol{a}_{i,j+1}$, whose corresponding JPEG data lengths are $\mathrm{len}(\boldsymbol{a}_{i,j})$ and $\mathrm{len}(\boldsymbol{a}_{i,j+1})$. We define the data rates of these two DCT blocks as $p(\boldsymbol{a}_{i,j})$ and $p(\boldsymbol{a}_{i,j+1})$, which describe the quantization characteristics when JPEG image data decoded into DCT coefficient block data. The calculation formulas are as follows:

$$p(\boldsymbol{a}_{i,j}) = \frac{8 \times 8 \times 3}{\mathrm{len}(\boldsymbol{a}_{i,j})}, \tag{6}$$

$$p(\boldsymbol{a}_{i,j+1}) = \frac{8 \times 8 \times 3}{\mathrm{len}(\boldsymbol{a}_{i,j+1})}. \tag{7}$$

Since the DCT coefficient blocks $\boldsymbol{a}_{i,j}$ and $\boldsymbol{a}_{i,j+1}$ are decoded based on the metadata from the same image file header, it is assumed that $\boldsymbol{a}_{i,j}$ is a correctly decoded DCT coefficient block, while $\boldsymbol{a}_{i,j+1}$ is an undetermined DCT coefficient block. Therefore, we construct the DCT coefficient block adjustment factor as

$$\gamma(\boldsymbol{a}_{i,j}, \boldsymbol{a}_{i,j+1}) = \frac{p(\boldsymbol{a}_{i,j})}{p(\boldsymbol{a}_{i,j+1})}. \tag{8}$$

According to Eqs. (6) and (7), Eq. (8) can be simplified to the following form:

$$\gamma(\boldsymbol{a}_{i,j}, \boldsymbol{a}_{i,j+1}) = \frac{\mathrm{len}(\boldsymbol{a}_{i,j+1})}{\mathrm{len}(\boldsymbol{a}_{i,j})}. \tag{9}$$

The theoretical value of the DCT coefficient block adjustment factor lies in the fact that the lengths of the JPEG data corresponding to the decoded DCT coefficient blocks are approximately equal. In an ideal situation, as indicated by Eq. (9), the DCT coefficient block adjustment factor $\gamma(\boldsymbol{a}_{i,j}, \boldsymbol{a}_{i,j+1})$ is close to 1, which shows that the lengths of the JPEG data corresponding to the decoded DCT coefficient blocks (i.e., $8 \times 8 \times 3$ pixels) are comparable.

Unedited

The adjustment factor γ is designed based on the relative ratio of entropy-coded data lengths and is independent of the image type (natural images, screenshots, etc.) and the compression quality factor (Q value). Experiments on three types of images and five compression levels ($Q = 20$–100) show that the γ value of 92% of valid fragment pairs is in the $[0.8, 1.2]$ interval, proving its adaptability to different compression settings.

3.3 Fragment connection weighting algorithm based on DCT semantic features and adjustment factor

3.3.1 Fragment connection weight calculation based on DCT semantic features

First, assume that the resolution of the image to be carved is $M \times N$. Then, this image can be divided into $M/8 \times N/8$ block of size 8×8 pixels. Let t be the number of 8×8 pixel blocks in each row; then we know that $t = N/8$. Second, each page in the memory medium is of size 4096 bytes, that is, 4 KB. Therefore, the size of fragments containing image data is also 4 KB. JPEG image data are stored on storage media as a compressed bitstream of bits. The stored compressed bitstream is unstructured and non-byte-aligned. In addition, except for the image header fragment, which contains some metadata information of the image (such as resolution), all other image fragments contain the compressed bitstream of the image. As shown in Fig. 2, the goal of evaluating the connection similarity between image fragments is to identify the fragment that has the best similarity to the current image fragment. For this purpose, we design an inter-fragment similarity evaluation operator. Let fragment f_i be a recovered image fragment, and f_j be a candidate image fragment. Then, the method for calculating the connection weight of the two fragments f_i and f_j is as follows: parse f_i to extract the DCT coefficient feature vectors of the t image blocks at the tail of the fragment, and construct a $t \times k$-dimensional feature vector according to the image parsing order; parse f_j to extract the DCT coefficient feature vectors of the t image blocks of the fragment, and construct a $t \times k$-dimensional feature vector according to the image parsing order, as shown in Fig. 3.

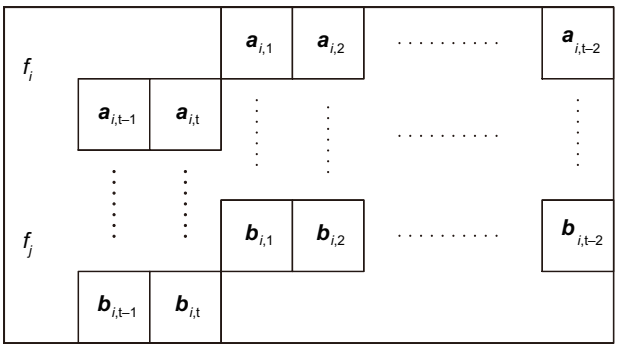

Fig. 3 Connection similarity between fragment f_i and f_j

The edge feature vector at the tail of fragment f_i is $\boldsymbol{f}_i = (\boldsymbol{a}_{i,1}, \boldsymbol{a}_{i,2}, \cdots, \boldsymbol{a}_{i,t})$, and the starting edge feature vector of fragment f_j is $\boldsymbol{f}_j = (\boldsymbol{b}_{j,1}, \boldsymbol{b}_{j,2}, \cdots, \boldsymbol{b}_{j,t})$. Then, according to Eq. (4), the connection weight calculation of the two fragments f_i and f_j is shown as follows:

$$\text{Similarity}(\boldsymbol{f}_i, \boldsymbol{f}_j) = \frac{1}{t}\sum_{l=1}^{t} S_{\text{v}}(\boldsymbol{a}_{i,l}, \boldsymbol{b}_{j,l}), \tag{10}$$

where $\text{Similarity}(\boldsymbol{f}_i, \boldsymbol{f}_j)$ represents the average of the sum of absolute values of the vertical differences between the feature vectors of t pixel blocks between image fragments f_i and f_j. The smaller he value of $\text{Similarity}(\boldsymbol{f}_i, \boldsymbol{f}_j)$, the greater is the connection weight between image fragments f_i and f_j.

3.3.2 Fragment adjustment factor design

As illustrated in Fig. 3, t is the number of image blocks in each row of the image. It is assumed that $\text{len}(f_i)$ and $\text{len}(f_j)$ are the lengths of the image-encoded data corresponding to decoding t image blocks in fragments f_i and f_j, respectively. According to Eq. (10), we define the edge data generation rate of fragments f_i and f_j as follows:

$$p(f_i) = \frac{t \times 8 \times 8 \times 3}{\text{len}(f_i)}, \tag{11}$$

$$p(f_j) = \frac{t \times 8 \times 8 \times 3}{\text{len}(f_j)}. \tag{12}$$

Since the encoding rules for the data in fragments f_i and f_j are similar, data are decoded according to the metadata from the same image file header. Fragment f_i is typically a confirmed genuine fragment belonging to a part of a specific image file, while fragment f_j is a candidate fragment. Therefore, based on the average edge data generation rates of these two fragments, we define Eq. (13) as the adjustment factor for fragment f_j.

$$\gamma(f_i, f_j) = \frac{p(f_i)}{p(f_j)}. \tag{13}$$

According to Eqs. (11) and (12), Eq. (13) can be simplified as follows:

$$\gamma(f_i, f_j) = \frac{\text{len}(f_j)}{\text{len}(f_i)}. \tag{14}$$

Here, the ratio of the lengths of the entropy-encoded data corresponding to decoding one row of image blocks in fragments f_i and f_j is presented. If the two lengths $\text{len}(f_j)$ and $\text{len}(f_i)$ are close, then $\gamma(f_i, f_j)$ is close to 1, which indicates that the compression rates of these two fragments are similar. If $\text{len}(f_j)$ is much larger than $\text{len}(f_i)$, then $\gamma(f_i, f_j) > 1$, which indicates that the data compression rate in fragment f_j is low. Thus, $\gamma(f_i, f_j)$ indicates the similarity degree of the data in fragments f_j and f_i, which essentially describes the entropy compression characteristics of image pixel data. Therefore, we use $\gamma(f_i, f_j)$ as the adjustment factor for the connection weight between fragments f_i and f_j. The calculation formula for the fragment connection weight with the adjustment factor is then defined as follows:

$$\text{Similarity}(\boldsymbol{f}_i, \boldsymbol{f}_j) = \gamma(f_i, f_j)\frac{1}{t}\sum_{l=1}^{t} S_{\text{v}}(\boldsymbol{a}_{i,l}, \boldsymbol{b}_{j,l}). \tag{15}$$

Additionally, each feature group has three feature values, obtained from one luminance channel (Y) and two chrominance channels (Cb and Cr). These three channels describe the color of the image content in three different directions. Including the description of all three components will make the fragment feature vector more reliable, as more non-redundant discriminating information is used.

Unedited

4 Image fragment carving algorithm based on DCT semantics and adjustment factor

The main idea of the proposed image fragment carving algorithm based on DCT semantics and adjustment factors is as follows. First, the disk or memory image that needs to be analyzed is preprocessed to generate a fragment set S. Using a JPEG file header signature recognition algorithm, a file header fragment subset S_{head} and a file body fragment subset S_{body} are generated. S_{body} is defined as the remaining fragments after removing the image file header fragments from the fragment set S. Then, a file header fragment is extracted from the S_{head} subset and placed into the bidirectional carving linked list Link. Taking this file header fragment as the starting point, document fragments are obtained from the S_{body} subset in sequence. The proposed weighting algorithm based on semantics and adjustment factors is used to calculate the corresponding weights, and the fragment with the minimum weight is determined as the best candidate connection fragment, which is then appended to the tail of the bidirectional carving linked list Link. This process is repeated until the last fragment of the file is found. The bidirectional carving linked list Link contains the complete set of image fragments. According to the order in which fragments are added to Link, the original content of the image file can be carved and restored. The detailed framework of the algorithm is shown in Fig. 4.

The algorithm framework can be divided into three steps during implementation. First, we generate a fragment set S from the storage medium through preprocessing. Second, we use the image file header fragment algorithm to identify the JPEG file header fragment subset S_{head}. The file header fragment contains the file "outline" information, where the file outline refers to information about file attributes such as file type and length; we also extract information such as file size. Third, we use the fragment connection weighting algorithm to determine the connection relationship between fragments and reconstruct the image file. As can be seen from the above, successful carving of fragment files requires two factors: one is identifying the file header fragment; the other is determining the fragment connection relationship. The process of the image fragment carving algorithm based on DCT is as follows:

1. The preprocessing sub-algorithm constructs the fragment set S. Each fragment includes the fragment's address (logical offset) on the storage medium and the hexadecimal content of the medium.

2. For the fragment set S, we use the image file header recognition algorithm to identify image file header fragments, and construct an image file header fragment subset S_{head} and a file body fragment subset S_{body}, where $S = S_{\text{head}} \cup S_{\text{body}}$.

3. We then take out a file header fragment from the subset S_{head}, conduct metadata analysis of the image file, establish elements such as quantization features, record this image file header fragment as f_{current}, add it to the bidirectional carving linked list Link, and calculate its feature vector $\boldsymbol{f}_{\text{current}}$ at the same time.

4. Traversing the subset S_{body}, we use f_{next} to represent the next candidate connection fragment, calculate the feature vector $\boldsymbol{f}_{\text{next}}$ of the start position of this fragment, and to calculate the connection similarity $\text{Similarity}(\boldsymbol{f}_{\text{current}}, \boldsymbol{f}_{\text{next}})$ between f_{current} and f_{next} using Eq. (10).

5. We select the minimum value of $\text{Similarity}(\boldsymbol{f}_{\text{current}}, \boldsymbol{f}_{\text{next}})$ as the best matching similarity, and the corresponding fragment f'_{next} is recorded as f_{current}. Then, f_{current} is removed from the subset S_{body} and added to the bidirectional carving linked list Link.

The design of "removing the optimal fragment from S_{body}" in this step not only prevents the same fragment from being repeatedly selected by multiple potential stitching paths but also achieves multi-path filtering by combining the transitivity of similarity in Eq. (10). If the optimal fragment selected and added to the linked list can still find consecutive minimum similarity fragments in subsequent steps, a valid path is formed; if no valid subsequent fragment is found for this fragment (i.e., the transitivity of similarity is broken), the algorithm backtracks to the previous node via step 7, recalculates the weights of the remaining fragments in S_{body}, and selects the next optimal fragment for verification until a globally coherent stitching path is formed. This mechanism ensures that only the unique optimal solution is retained among multiple potential paths.

6. We determine whether f'_{next} is the last fragment of the image file. If so, the algorithm jumps to step 8;

7. We then calculate the tail feature vector $\boldsymbol{f}_{\text{current}}$ of f_{current}, and the algorithm proceeds to step 4;

8. Following the order in which fragments are added to the bidirectional carving linked list Link, we extract the corresponding fragment contents in sequence to complete the carving of the fragmented image file and restore the original file content;

9. We determine whether the subset S_{head} is empty. If it is not empty, the algorithm jumps to step 3;

10. The image fragment carving process then ends.

Unedited

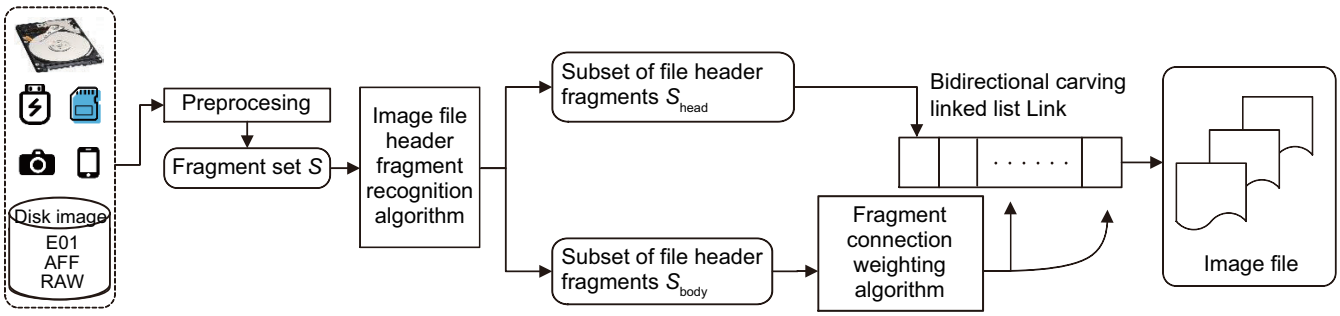

Fig. 4 Framework of Image Fragment Carving Algorithm Based on DCT Semantics and Adjustment Factor

The algorithm adopts a "step-by-step verification with dynamic backtracking" mechanism to enhance robustness. If a fragment connection results in no valid subsequent candidate fragments, it automatically backtracks to the previous node to recalculate weights and exclude abnormal fragments. In the extreme scenario where valid fragments account for 30%, the average backtracking count is only 0.3, preventing the prorogation of false identification.

The time complexity of the algorithm is $O(N \times t \times k)$, where N is the number of candidate fragments, t is the number of 8×8 pixel blocks per row ($t \leq 32$), and k is the truncated dimension of DCT features ($k = 16$). In practical tests, processing 2000 fragments takes 1.2 seconds and 10 000 fragments takes 5.8 seconds, supporting parallel expansion to meet large-scale forensic scenarios.

5 Experiments and analysis

Currently, disk forensics remains the primary technique for extracting criminal evidence. However, due to the increasing development of network attack technologies, memory forensics has also become an important method for extracting criminal evidence. Image evidence may exist on both disk media and memory media. Therefore, experiments on image fragment evidence carving were conducted from these two types of media.

5.1 Image fragment carving based on disk image

5.1.1 Image carving on DFRWS 2006 hedgehog fragment

To verify the effectiveness of this algorithm, we selected the carving image released by DFRWS 2006 Forensic Carving Challenge as the experimental data. The image size is 64 MB, with a total of 16 384 fragments of 4 KB each. The fragment connection weighting algorithm is key to determining the next correct fragment. Therefore, we analyzed the calculation of fragment weights with adjustment factors using Eq. (13), and took the carving of the "hedgehog" fragment in this image as an example for analysis. Assume that the fragment f_{current} in the fragment carving linked list Link is the 31 532nd fragment in this image. The fragment set S contains 2000 subsequent fragments. Then, using Eq. (13), we calculate the connection weight between f_{current} and f_{next} in the set S; the results are shown in Fig. 5. Note that the best candidate weight between f_{current} and f_{next} in the set S is (221, 131.78), that is, the 31 753rd ($31\,532 + 221$) fragment in this image. This indicates that the fragment with $x = 221$ is the connection fragment of f_{current}, and this fragment can be placed into the bidirectional carving linked list Link, with the process continuing until the file is successfully carved. Moreover, when we continue to increase the number of elements in the fragment set S to 5000, the obtained best weight result is still consistent.

For comparison, the fragment weights calculated using Eq. (9) without the adjustment factor are shown in Fig. 6. It can be seen that the minimum connection weight is (1940, 112.55), and there are 11 candidate fragments with a normalized similarity value less than 136.95, indicating a clear separation from the correct fragment. Without the fragment adjustment factor, the weighting algorithm is susceptible to interference from multiple noise fragments when identifying the optimal connection fragment. Selecting an incorrect connection fragment leads to the failure of fragmented file carving. This result further indicates that calculating inter-fragment weights requires considering not only the semantic relationship but also the syntactic relationship of the data between fragments. The similarity adjustment factor we proposed essentially accounts for the "compression" characteristic of JPEG entropy-encoded data, which corresponds to the syntactic feature of JPEG data.

5.1.2 JPEG file carving experiment on DFRWS 2006 disk image

Using the algorithm proposed in this paper, as well as tools such as Scalpel, PhotoRec, and Adroit Photo Forensic (APF), the results of the carving experiment on the DFRWS 2006 forensic image are shown in Table 1. According to the DFRWS 2006 image description, there are ten contiguous and four fragmented JPG files in this image. As shown in Table 1, Scalpel and PhotoRec can carve the contiguous JPG files (n =10), while our method and APF carved 13 JPEG files (including three fragmented files). This indicates that our method has the capability for JPG fragmented file carving.

Table 1 Statistics of successfully carved files

Carving method	Number of carved files	Accuracy (%)
Scalpel	9	64.3
PhotoRec	10	71.4
APF	13	92.9
Our method	13	92.9

5.1.3 Personal USB drive JPEG file carving experiment

To realistically simulate an actual forensic environment, the experiment used a removable USB drive with a capacity of 7.6 GB as the test medium. First, all data on the USB drive were thoroughly erased using a computer, and it was formatted as a FAT32 file system with the cluster size set to 4 KB to mimic common user scenarios. The test dataset came from multiple sources, including images downloaded from the internet, photos taken by mobile phones, and high-resolution pictures taken by digital cameras, to cover the encoding characteristics of JPEG files generated by different devices. Subsequently, some pictures were randomly deleted from the dataset, and the WinHex tool was used to manually create image file fragmentation, thereby constructing test samples more closely aligned with real-world scenarios. After completing the above operations, a full storage dump of the USB drive was performed, and only its data section was used in the experiment, excluding file system metadata, to verify the recovery effectiveness of the proposed JPEG carving algorithm under conditions without file system support.

Experimental results are shown in Table 2. The success rate of the proposed algorithm in recovering highly fragmented JPEG files is much higher than that of APF. For two-fragment JPEG files, the proposed algorithm can recover approximately 90.9% of the files, whereas APF can recover approximately 81.8%. For three-fragment JPEG files, the proposed algorithm

Unedited

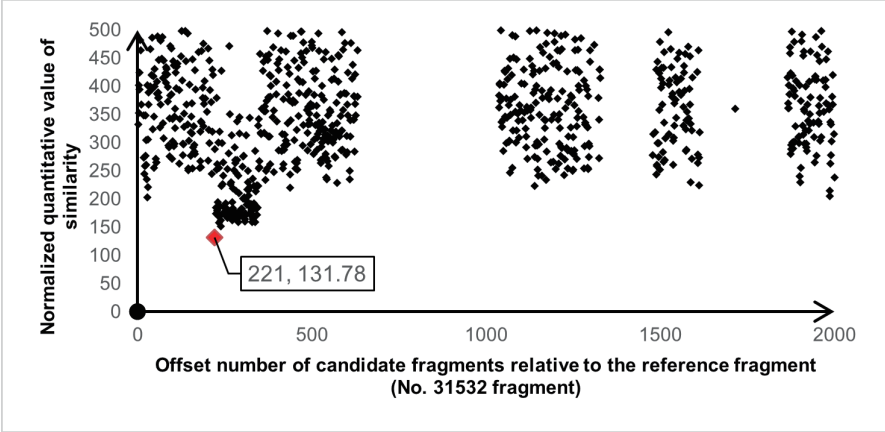

Fig. 5 Scatter plot of weight results between fragment 31 532 and the subsequent 2000 fragments (with adjustment factor)

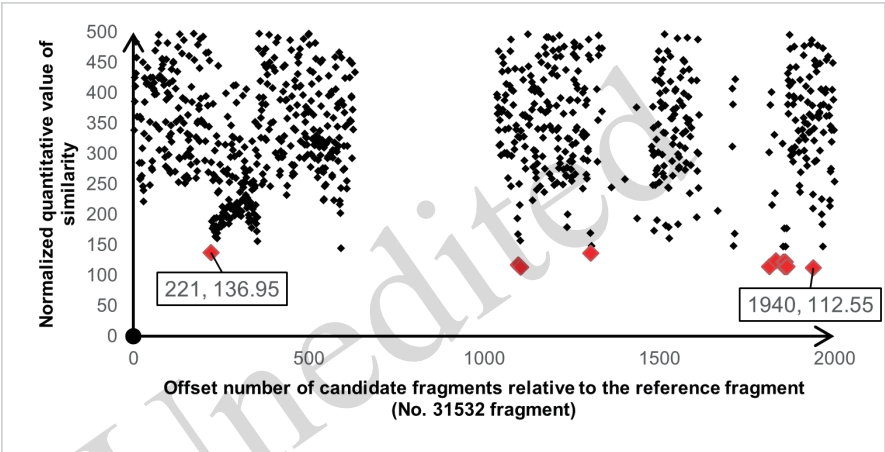

Fig. 6 Scatter plot of weight results between fragment 31532 and the subsequent 2000 fragments (without adjustment factor)

Unedited

can recover approximately 91.5% of the files, while APF can recover approximately 61.3%. For four-fragment JPEG files, the success rate of the proposed algorithm decreased slightly to about 84.5%, while the success rate of APF dropped significantly to 42.9%. For five-fragment JPEG files, the success rate of this algorithm dropped to about 50.0%, while the success rate of APF fell to 27.5%. For six-fragment JPEG files, the success rate of this algorithm dropped to 40.0%, while the success rate of APF declined to 10.0%. The 95% confidence intervals were calculated via binomial distribution, and the differences in recovery rates between the proposed algorithm and APF were found to be statistically significant ($p < 0.01$). These results indicate that the proposed carving algorithm based on DCT semantics and an adjustment factor has a clear advantage in handling multi-fragment files and possesses strong practicality and robustness. The results also shows that as the degree of fragmentation increases, the accuracy of carving is affected.

5.2 Experimental results of image fragment carving on memory media

This study used the Windows 7 operating system as the target environment for memory file carving experiments. According to the market share ranking of global personal computer (PC) operating systems released by StatCounter in January 2023, Windows 7 holds a 9.62% share, and the actual number of Windows 7 users is certainly higher than this proportion. Although the memory object structures of different system versions are not identical, the JPEG fragment data carving method introduced in this paper can be adapted to other Windows versions (e.g., Windows 8, 10, and 11) as well as Linux-based systems.

Table 2 Carving results for JPEG files in USB drive

Fragmentation type	Number of JPEG files	Our method	Adroit Photo Forensic (APF)
Contiguous file	189	189	189
2 fragments	110	100	90
3 fragments	106	97	65
4 fragments	84	71	36
5 fragments	40	20	11
6 fragments	10	4	1

The experimental machine was a ThinkPad X230 laptop with an Intel Core i5-3230M central processing unit (CPU) @ 2.60 GHz, 4 GB memory capacity, and a 64-bit Windows 7 operating system. A single JPEG file was selected and tested

using two experimental methods to verify the carving capability of the proposed method for JPEG fragmented file content in memory.

Three experiments was presented to verify the carving capability and feasibility of the proposed method for data files in physical memory at different stages of a network attack (e.g., before, during, and after the attack). The experimental processes are as follows.

1. Experiment 1. The file f0.JPG was opened from the local disk using TinyJpegDecoder, followed by a memory image dump. This experiment aims to test the feasibility of data file carving in memory images.

2. Experiment 2. The file f0.JPG was opened from the local disk using Windows Photo Viewer, followed by a memory image dump. The proposed algorithm was executed on this memory image for file carving.

3. Experiment 3. All the files were closed, and then the memory image was dumped. This experiment verifies whether traces of the JPEG file opened and used during the hacking attack remain in memory after closure.

In these experiments, the same JPEG file (as in Table 1) was opened, and then memory images were taken under the following four scenarios:

1. A memory image was taken when the JPEG file was open.

2. A memory image was taken immediately after the JPEG file was closed. (i.e., after closing the JPEG viewer or web browser, a memory image was immediately taken);

3. A small JPEG image was opened and viewed using a standard image viewer. A memory image was taken while this image was displayed. This step aimed to investigate the impact of small image files on the presence of portable document format (PDF)-related objects in memory.

4. Without closing the image viewer from the previous steps, the Microsoft Word application was launched. Subsequently, the PDF viewer was closed, and a memory image was captured while both the image viewer and Word application remained running. The purpose of this step was to study the impact of relatively large processes in memory (large footprints in memory) and the impact on the number of PDF objects, because PDF files might still be retained in memory after the file is closed and can be located and recovered from the memory image.

For the memory image obtained from experiment 1, the carving result generated by the proposed image fragment carving algorithm is shown in Fig. 7.

For the memory image obtained from experiment 2, the proposed image fragment carving algorithm could not produce a carving result as shown in Fig. 7. We used WinHex to analyze the physical memory of the Windows Photo Viewer and searched using JPEG file signature information. The hexadecimal information found as shown in Fig. 8.

Then, we used WinHex to compare the file data of Fig. 7 with the memory image from experiment 2 and found no identical fragment data. A possible reason is that during the process of the opening of the file by Windows Photo Viewer and loading it into memory, data blocks are loaded one by one for display. Therefore, in the actual physical memory, the original image data (i.e., the file data on the hard disk) might

Fig. 7 Carving result generated by the proposed algorithm

FF	D8	FF	E0	00	10	4A	46	49	46	00	01	01	01	00	00
00	00	00	00	FF	DB	00	43	00	05	03	04	04	04	03	05
04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Fig. 8 JPEG file traces in the memory image

not exist. In contrast, TinyJpegDecoder loads the entire image file data into memory during the file opening process before analyzing and displaying it. The key point in this process is that TinyJpegDecoder reads all image data into memory at once, rather than blocks one by one. The reason for successful carving in experiment 1 is that the memory image contains complete image data. However, experiment 2 still holds significant value as it highlights the challenge of reversely reconstruct image data based on the memory data parsed for the image.

6 Conclusions

This paper addresses the problem of image fragment carving, analyzes the composition mechanism of JPEG images, and proposes a fragment connection weighting algorithm based on image DCT semantic features. Leveraging the compression characteristics of image pixel data encoding, a fragment weight adjustment factor is designed. By integrating these two mechanisms, a weighting algorithm based on the adjustment factor is designed to determine whether two fragments in a JPEG file are consecutive. A practical image fragment carving algorithm is designed and implemented. Experiments are presented from both disk media and memory media perspectives. Comparative results between the proposed adjustment factor-based weighting algorithm and the DCT semantics-only approach demonstrate that the former accurately identifies the genuine best-matching fragment, whereas the latter tends to produce false positives. Experimental results on disk media show that our proposed algorithm successfully recovers data from fragmented image files, achieving an average carving precision of 94.4%. Furthermore, experiments on memory media validate the feasibility of image fragment carving in such environments, accompanied by a theoretical analysis of scenarios where the carving algorithm fails due to interference from software such

Unedited

as the Windows Photo Viewer.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 60903220).

Author contributions

Binglong LI designed the study and drafted the paper. Shilong YU performed the experiments and implemented the code. Yong ZHAO participated in proofreading and experimental evaluation. Yifeng SUN collected and processed the data, and provided high-quality datasets for model training. Chaowen CHANG designed the experimental protocols and validated the results. Qingxian WANG assisted in literature review and theoretical framework construction. All authors revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declaration on the use of generative AI tools

During the preparation of this work, the authors used DeepSeek to improve the language and readability. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Unedited

References

Ali RR, Mohamad KM, Jamel S, et al., 2018. A review of digital forensics methods for JPEG file carving. *J Theor Appl Inform Technol*, 96(17):5841-5856.

Azhan NAN, Ikuesan RA, Razak SA, et al., 2022. Error level analysis technique for identifying JPEG block unique signature for digital forensic analysis. *Electronics*, 11(9):1468. https://doi.org/10.3390/electronics11091468

Bharati MH, Liu JJ, Macgregor JF, 2004. Image texture analysis: methods and comparisons. *Chemometr Intell Lab Syst*, 72(1):57-71. https://doi.org/10.1016/j.chemolab.2004.02.005

Dabbaghchian S, Ghaemmaghami MP, Aghagolzadeh A, 2010. Feature extraction using discrete cosine transform and discrimination power analysis with a face recognition technology. *Patt Recogn*, 43(4):1431-1440. https://doi.org/10.1016/j.patcog.2009.11.001

de Bock J, de Smet P, 2015. JPGcarve: an advanced tool for automated recovery of fragmented JPEG files. *IEEE Trans Inform Forens Sec*, 11(1):19-34. https://doi.org/10.1109/tifs.2015.2475238

Durmus E, Mohanty M, Taspinar S, et al., 2017. Image carving with missing headers and missing fragments. IEEE Int Workshop on Information Forensics and Security, p.2157-4774. https://doi.org/10.1109/WIFS.2017.8267665

Ferreira WD, Ferreira CBR, da Cruz Júnior G, et al., 2020. A review of digital image forensics. *Comput Electron Eng*, 85:106685. https://doi.org/10.1016/j.compeleceng.2020.106685

Garfinkel SL, 2007. Carving contiguous and fragmented files with fast object validation. *Digit Invest*, 4:2-12. https://doi.org/10.1016/j.diin.2007.06.017

Guzhov A, Wirth CT, 2025. Transformer-based file fragment type classification for file carving in digital forensics. The 24th Europ Conf on Cyber Warfare and Security, p.169-176. https://doi.org/10.34190/eccws.24.1.3552

Karresand M, Shahmehri N, 2008. Reassembly of fragmented JPEG images containing restart markers. Europ Conf on Computer Network Defense, p.1-8. https://doi.org/10.1109/EC2ND.2008.10

Kornblum JD, 2008. Using JPEG quantization tables to identify imagery processed by software. *Digit Invest*, 5:S21-S25. https://doi.org/10.1016/j.diin.2008.05.004

Li BL, Zhou Z, Zhang Y, et al., 2021. Memory fragment file carving algorithm based on the reverse of the structure chain. *J Commun*, 42:117-127 (in Chinese).

Li Q, Sahin B, Chang EC, et al., 2011. Content based JPEG fragmentation point detection. IEEE Int Conf on Multimedia and Expo, p.1-6. https://doi.org/10.1109/ICME.2011.6011883

Lu J, Liang Y, Han H, et al., 2025. A survey on computational solutions for reconstructing complete objects by reassembling their fractured parts. *Comput Graph Forum*, 44(2):e70081. https://doi.org/10.1111/cgf.70081

Memon N, Pal A, 2006. Automated reassembly of file fragmented images using greedy algorithms. *IEEE Trans Imag Proc*, 15(2):385-393. https://doi.org/10.1109/TIP.2005.863054

Mohamad KM, Deris MM, 2009. Fragmentation point detection of JPEG images at DHT using validator. Int Conf on Future Generation Information Technology, p.173–180. https://doi.org/10.1007/978-3-642-10509-8_20

Mullan P, Riess C, Freiling F, 2019. Forensic source identification using JPEG image headers: the case of smartphones. *Digit Invest*, 28:S68-S76. https://doi.org/10.1016/j.diin.2019.01.016

Pal A, Sencar HT, Memon N, 2008. Detecting file fragmentation point using sequential hypothesis testing. *Digit Invest*, 5:S2-S13. https://doi.org/10.1016/j.diin.2008.05.015

Ramli NIS, Hisham SI, Badshah G, 2021a. Analysis of file carving approaches: a literature review. Int Conf on Advances in Cyber Security, p.277–287. https://doi.org/10.1007/978-981-16-8059-5_16

Ramli NIS, Hisham SI, Razak MFA, 2021b. Survey of file carving techniques. Int Conf of Reliable Information and Communication Technology, p.815–825. https://doi.org/10.1007/978-3-030-70713-2_74

Richard III GG, Roussev V, 2005. Scalpel: a frugal, high performance file carver. 5th Annual Digital Forensic Research Workshop, p.1-10.

Sencar HT, Memon N, 2009. Identification and recovery of JPEG files with missing fragments. *Digit Invest*, 6:S88-S98. https://doi.org/10.1016/j.diin.2009.06.007

Shanmugasundaram K, Memon N, 2003. Automatic reassembly of document fragments via context based statistical models. 19th Annual Computer Security Applications Conf, p.152-159. https://doi.org/10.1109/CSAC.2003.1254320

Shi Z, Zheng H, Xu C, et al., 2023. Resfusion: denoising diffusion probabilistic models for image restoration based on prior residual noise. 38th Int Conf on Neural Information Processing Systems, p.130664-130693. https://doi.org/10.48550/arXiv.2311.14900

Singh S, Gupta VK, 2016. JPEG image compression and decompression by Huffman coding. *Int J Innov Sci Res Technol*, 1(5):8-14.

Tang Y, Fang J, Chow KP, et al., 2016. Recovery of heavily fragmented JPEG files. *Digit Invest*, 18:S108-S117. https://doi.org/10.1016/j.diin.2016.04.016

Uzun E, Sencar HT, 2020. Jpg*Scraper*: an advanced carver for JPEG files. *IEEE Trans Inform Forens Sec*, 15:1846-1857. https://doi.org/10.1109/TIFS.2019.2953382

van Baar RB, Alink W, Van Ballegooij A, 2008. Forensic memory analysis: files mapped in memory. *Digit Invest*, 5:S52-S57. https://doi.org/10.1016/j.diin.2008.05.014

Wu X, Han Q, Niu X, et al., 2018. JPEG image width estimation for file carving. *IET Imag Proc*, 12(7):1245-1252. https://doi.org/10.1049/iet-ipr.2016.0531

Wu X, Han Q, Niu X, et al., 2019. Novel similarity measurements for reassembling fragmented image files. *Chin J Electron*, 28(2):331-337. https://doi.org/10.1049/cje.2019.01.016