



**HAL**  
open science

## Automating Video File Carving and Content Identification

York Yannikos, Nadeem Ashraf, Martin Steinebach, Christian Winter

► **To cite this version:**

York Yannikos, Nadeem Ashraf, Martin Steinebach, Christian Winter. Automating Video File Carving and Content Identification. 9th International Conference on Digital Forensics (DF), Jan 2013, Orlando, FL, United States. pp.195-212, 10.1007/978-3-642-41148-9\_14 . hal-01460607

**HAL Id: hal-01460607**

**<https://inria.hal.science/hal-01460607v1>**

Submitted on 7 Feb 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

## Chapter 14

# AUTOMATING VIDEO FILE CARVING AND CONTENT IDENTIFICATION

York Yannikos, Nadeem Ashraf, Martin Steinebach and Christian Winter

**Abstract** The massive amount of illegal content, especially images and videos, encountered in forensic investigations requires the development of tools that can automatically recover and analyze multimedia data from seized storage devices. However, most forensic analysis processes are still done manually or require continuous human interaction. The identification of illegal content is particularly time consuming because no reliable tools for automatic content classification are currently available. Additionally, multimedia file carvers are often not robust enough – recovering single frames of video files is often not possible if some of the data is corrupted or missing. This paper proposes the combination of two forensic techniques – video file carving and robust hashing – in a single procedure that can be used for the automated recovery and identification of video content, significantly speeding up forensic investigations.

**Keywords:** Automated forensic procedures, video file carving, robust hashing

## 1. Introduction

The amount of contraband image and video files has increased drastically over the past decade. As a result, forensic investigators need sophisticated tools to help recover and analyze image and video content from evidentiary sources. However, the identification of illegal content is a hard problem, and no reliable tools for automatic content classification are currently available. Thus, each multimedia file has to be inspected manually during a forensic investigation.

File carving is an advanced technique for recovering deleted data from storage devices. It works independently of the underlying file system,

enabling the recovery of deleted data that has not already been overwritten.

Most audio and video files (e.g., MPEG files) are streams of data unlike WAV files, images and non-multimedia data. These streams consist of many frames and the corresponding frame headers can be used as the starting points for file carving. This makes it possible to extract individual chunks of multimedia streams when portions of the streams are already overwritten.

After multimedia data is recovered, it is necessary to conduct a time-consuming inspection and search for evidentiary data. Content classification techniques using blacklisting/whitelisting and robust hashing are often used by forensic investigators. Since blacklisting/whitelisting approaches are mostly based on cryptographic hashing, they have high false negative rates. On the other hand, robust hashing is a promising approach for detecting copies of multimedia data after content-preserving changes such as lossy compression or format conversion.

This paper proposes the combination of two forensic techniques – video file carving and robust hashing – in a single procedure that can be used for the automated recovery and identification of video content. It focuses on carving video files encoded using the MPEG video format. The procedure identifies, extracts and decodes single intracoded frames (I-frames) of MPEG video data from raw data sets in a robust manner that allows the partial recovery of frames with missing or corrupted data. Robust hashes of all the decoding results (i.e., image data) are generated and then compared with the hashes in a reference database (blacklist) to identify similarities with known illegal content. This procedure facilitates automated searches and recovery of video content, significantly speeding up forensic investigations.

## 2. File Carving

File carving is the process of extracting file fragments from a byte stream without file system information. This technique is mostly used to recover data from unallocated space on a hard disk. A file carver typically uses file header and footer information to identify the beginning and end of a file. Everything that lies in between a file header and footer is assumed to belong to a single file, which is then “carved” out and restored. Common file carvers that use header and footer information include PhotoRec [5], Scalpel [11] and Foremost [13]. However, if the file to be recovered is fragmented, i.e., it is not stored sequentially, but is split up into two or more parts and distributed throughout the storage area, file carving becomes difficult.

A study by Garfinkel [3] noted that, although hard disks typically exhibit low levels of fragmentation, the larger files tend to be fragmented. Since multimedia video files (e.g., AVI and MPEG files) and high resolution images (e.g., from digital cameras) are usually large, a higher percentage of them should be fragmented. In fact, Garfinkel discovered that about 17% of MPEG files and 20% of AVI files are fragmented.

Several approaches have been proposed to handle file fragmentation during file carving. One approach is bi-fragment gap carving [3], a technique that validates file contents with gaps in between them. However, this technique only works for files with no more than two fragments and with small gaps.

Pal and Memon [10] developed a graph-theoretic approach for file carving that employs hypothesis testing for fragmentation point detection. The approach is effective at recovering fragmented JPEG files. However, models and weighting techniques have to be developed for all the file types of interest.

## 2.1 Video File Carving

Most of the available carving strategies and implementations are not robust enough to handle multimedia files with several missing blocks. This is because missing blocks make it practically impossible to decode the data.

Since audio and video data are usually stored and transmitted as streams, the stream properties can be leveraged to support more robust data recovery and analysis. Multimedia streams comprise frames, each of which has a header followed by the actual data content. A frame header usually contains enough information to allow the calculation of the frame size, which helps determine the starting position of the next frame. This facilitates multimedia file parsing, fragmentation detection and partial recovery.

Relatively little work has been done on carving methods for video file recovery. Yoo, *et al.* [18] have developed an AVI file carving approach that works by parsing AVI file format information. They have also proposed similar approaches for MP3 and WAV files, and for NTFS compressed files. However, they assume that all the media files are stored sequentially and they only consider data cut-offs, disregarding file fragmentation with two or more fragments.

The Netherlands Forensics Institute has developed Defraser, an open source tool for video data carving [14]. Defraser can carve multimedia files such as MPEG-1/2, WMV and MP4. It can also extract and view single frames of MPEG-1 data. However, Defraser relies on a valid

MPEG-1 structure and is unable to handle small occurrences of fragmentation and missing data in specific file areas.

### 3. Content Identification

The identification of illegal multimedia content is an important task for forensic investigators. The typical approach is to manually inspect all the multimedia data found on a seized storage device. Automated approaches are necessary because manual inspection is very time consuming and is prone to the accidental overlooking of evidence and exonerating content.

Blacklisting and whitelisting are often used to filter known illegal and legal content. Blacklisting involves the filtering of illegal, harmful content using a reference database called the blacklist. Whitelisting involves the filtering of harmless content (e.g., operating system files) to reduce the amount of content to be investigated. To reduce the size of a reference database, an identifying hash of fixed length is generated for each file and stored instead of the file itself. Typically, cryptographic hash functions are used to generate the file hashes.

If blacklists and whitelists are available, a seized storage device can be automatically searched for known content – for each file found on the storage device, a hash is generated and compared with the hashes in the blacklist and whitelist. In this way, illegal content as well as harmless content can be identified. After this is done, it is necessary to inspect all the remaining files. This is by far the most time consuming part of a forensic investigation. When new harmful and harmless files are identified during the inspection, their hashes are stored in the blacklist and whitelist, respectively.

#### 3.1 Robust Hashes

Unlike cryptographic hashes, “perceptual” or “robust” hashes are specifically designed for multimedia data. Robust hashes are generated from multimedia data properties based on human perception (e.g., brightness, form and contrast). Robust hashing provides an automated mechanism to decide if two multimedia files (with different cryptographic hashes) are perceived as being identical or very similar to each other. Several researchers have used robust hashing on image and video data [2, 9, 19] and on audio data [1, 6]. Lejsek, *et al.* [7] have used robust hashing to identify illegal image and video content. Their approach uses a combination of local image descriptors based on SIFT and multidimensional NV-trees to identify video data. Experiments indicate that the approach is robust to video modifications such as mirroring, scaling

and cropping. However, since the content to be identified is assumed to be available, their approach is not applicable to the preliminary investigation of storage media.

We have recently proposed an improved version [15] of the block-based video hash introduced by Yang, *et al.* [17]. This block-based video hash was chosen because it strikes the right balance between low computational complexity and good detection results.

## 4. Multimedia Content

Advanced methods for multimedia content recovery and identification/classification require the decoding of multimedia data. Knowledge about the specific structures of multimedia file formats is vital to these methods. Since this paper focuses on the recovery and identification of MPEG-1 video data fragments, we briefly describe the structure of the MPEG-1 video format.

An MPEG-1 video stream is essentially a sequence of still images called frames. The video stream is divided into several layers (Figure 1). The first layer is the sequence layer, which denotes a number of video sequences, each starting with a sequence header and ending with a sequence end code. The sequence header stores information necessary for decoding the actual picture data (e.g., resolution information). An arbitrary number of groups of pictures (GOPs) exist between the sequence header and sequence end code. The GOP also starts with a header followed by a number of pictures (frames). The first frame of each GOP is always an I-frame; the others are B-frames and P-frames.

In the picture layer, the header is followed by slices containing the actual picture data. The picture data itself is organized in a macroblock layer with a header, four luma blocks with brightness information and two chroma blocks with color information.

MPEG-1 files may have a video stream or video and audio streams. In the latter case, the two streams are multiplexed as a transport stream (e.g., used for network streaming) or a program stream (e.g., used for storage purposes).

## 5. Challenges

According to the Cyber Forensic Field Triage Process Model [12], it is important to increase the speed at which forensic analysis is performed. This requires efficient forensic tools that involve as little human interaction as possible. Automated procedures can significantly reduce the workload of forensic investigators.

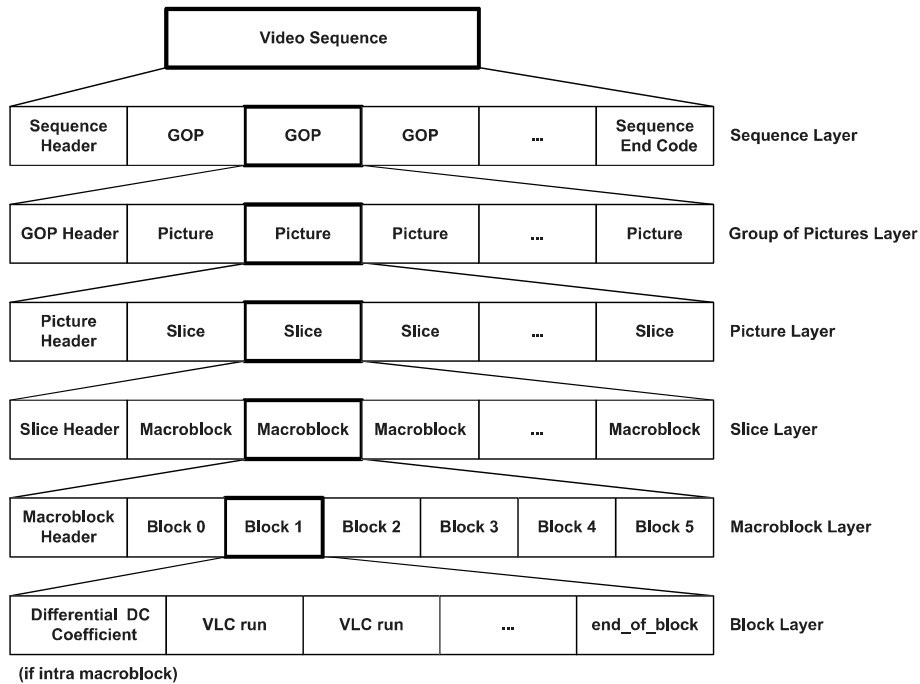


Figure 1. MPEG-1 video stream organization (from [16]).

Although several file carving approaches exist, the main challenge is dealing with fragmentation. Most file carvers are not robust enough to handle even small amounts of fragmentation, or they merely assume that all the data found between a header and footer belongs to a single file. Currently available video file carvers allow single frame carving as well as decoding and visualization, but often fail if just a few bytes of a frame are corrupted or missing.

At this time, law enforcement primarily uses cryptographic hashing to identify known multimedia content. However, this approach cannot identify content that has been (even slightly) modified, for example, through compression or cutting. For these reasons, a considerable amount of time is spent to manually inspect the various types of unidentified multimedia content that are encountered during a forensic investigation.

## 6. Video Frame Recovery and Identification

Our automated procedure for video file recovery and identification requires little or no interaction with a forensic investigator. Although we use the MPEG-1 as an example format for the recovery and decoding

steps, the steps can be applied to any video format with components similar to I-frames.

The automated procedure incorporates three steps that are performed in sequence:

- 1. Robust Recovery:** This step involves searching a storage device or hard disk image, and reconstructing complete video files or single frames of video data found on the device. Specific properties of video file formats are engaged to support robust recovery.
- 2. Robust Decoding:** This step involves the decoding of video content. Techniques such as resolution correction and resolution identification are used to enable decoding even when some of the data is corrupted or missing.
- 3. Robust Identification:** This step involves the calculation of robust hashes of video content (single frames) to identify unchanged content and slightly modified content. The reference databases are also searched for known robust hashes.

## 6.1 Robust Recovery

The robust recovery of video content should apply file carving in a more advanced manner than traditional header/footer identification. Therefore, we designed a video frame carving approach that analyzes video frame information in order to extract and decode single I-frames. If the resolution information required for decoding is not present or cannot be found, a resolution identification/correction approach is applied to decode the available data.

The algorithm for recovering and decoding video frames has the following steps:

- 1.** Locate all the I-frames by searching the input data for I-frame headers and verifying the headers. If no I-frame headers are found, stop.
- 2.** Starting with the located I-frame headers, search the input data backwards for corresponding sequence headers within a specific threshold range.
- 3.** If a sequence header is found, extract the resolution information from the sequence header; this information is needed for the subsequent decoding of I-frames.
- 4.** Read the data following the I-frame header; this is typically optional and/or I-frame slice data. Ignore the optional data and ver-



ify the slice data. Ignore the missing and invalid slice data parts; instead, read the data that follows within a specific threshold range until additional slice data is found.

5. Read the following data and verify the structure until the next frame start code, GOP start code or sequence end code is found.
6. If the MPEG stream was identified as a system stream, demultiplex the I-frame data; otherwise, skip this step.

## 6.2 Robust Decoding

The recovered I-frames must be decoded to allow further processing such as automated content identification. Typically, decoding proceeds without any problem if all the information required for decoding is available. For an I-frame, this includes the resolution information, which is stored in its corresponding sequence header. If the specific sequence header is not available (e.g., due to fragmentation or overwriting), the correct resolution needed for decoding has to be found in some other manner.

Since a robust decoding should be able to handle cases where I-frame resolution information is missing, we propose the use of a resolution identification and correction procedure. This involves decoding frames with commonly used resolutions and subsequently measuring the pixel row differences of the decoded frames to automatically identify the correct decoding results. All the decoding results should be stored in a widely-supported image file format such as JPEG.

The robust decoding algorithm has the following steps:

1. Decode all the I-frames with their corresponding resolution information obtained from the sequence headers.
2. For each I-frame with missing resolution information:
  - (a) Decode the I-frame using each unique resolution previously extracted from the sequence headers.
  - (b) Decode the I-frame using the most common resolutions for the specific video format (e.g., source input format (SIF)).
  - (c) Apply the resolution correction
3. Store each decoded result as a JPEG file.

The first two steps of the algorithm help recover and decode frames of video files that are fragmented or are partially overwritten.

**Resolution Correction.** During a video frame recovery process, the sequence headers containing resolution information may be missing for certain frames. This could occur because the headers were overwritten with other data or they could not be located due to fragmentation. For these frames, the following automated resolution identification/correction algorithm is applied:

1. **Identify Suitable Resolutions for Frame Decoding:** Initially identify the suitable resolutions for decoding, e.g., chosen from the most commonly used resolutions of the specific video format or from resolutions of frames found in the same raw data set.
2. **Decode Frames Using Suitable Resolutions:** Decode each frame using the previously determined suitable resolutions (individually chosen for each frame).
3. **Correct Resolution Identification for Incorrect Decoding Results:**
  - (a) Divide each decoding result into blocks of  $16 \times 16$  pixels.
  - (b) Discard the green blocks that appear at the end of a decoding result.
  - (c) Determine all possible resolutions for the remaining  $16 \times 16$  pixel blocks.
  - (d) Create new decoding results using these resolutions.
  - (e) Measure the global pixel block row difference as follows:

$$\Delta(r) = \frac{\sum_{i=1}^{\frac{n-1}{b}} \sum_{j=1}^m \delta_v^2(bi, bi+1, j)}{1 + \sum_{i=1}^{\frac{n-1}{b}} \sum_{j=1}^m c(bi, bi+1, j)}$$

for each decoding result  $r$  with  $n, m$  dimensions of  $r$  ( $n$  rows,  $m$  columns);  $b$  pixel block size (here,  $b = 16$ );  $p(i, j)$  pixel value for pixel at row  $i$ , column  $j$ ; vertical pixel difference  $\delta_v(i, i', j) = |p(i, j) - p(i', j)|$  and

$$c(i, i', j) = \begin{cases} 1 & \text{if } \delta_v(i, i', j) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- (f) Determine the result  $r$  with the smallest  $\Delta(r)$  as the correct decoding result.

### 6.3 Robust Identification

Robust identification is the final step of the automated procedure for processing the recovered frames. In general, the recovered video frames can be processed via automated classification or identification of their content. However, many classification approaches are somewhat limited and yield high false positive and/or false negative rates. An example is an approach that uses the amount of skin tones in an image to infer that the image shows human skin. Therefore, we propose the use of robust hashing to identify known content in an automated manner. Our robust block hashing method [15] allows for efficient content identification; it can help divide large sets of recovered frames into known and unknown content. Cryptographic hashing can also be used for automated content identification, but it lacks robustness.

Our robust block hashing method is used for the robust identification of MPEG-1 frames. Therefore, all the decoding results stored as JPEG files after the robust decoding procedure are further processed by detecting and removing horizontal or vertical black bars caused by letterbox, pillarbox or windowbox effects. Then, the corresponding robust block hashes are calculated, which are subsequently used to query a reference database (e.g., with blacklisted content). If the same or a similar robust block hash is found in the reference database, the corresponding I-frame is classified as showing illegal content.

## 7. Experimental Evaluation

This section describes the experimental results obtained for the robust recovery, robust decoding and robust identification steps of our procedure.

### 7.1 Recovery (MPEG-1 Frame Carving)

For our experiments, we prepared ten data sets, each consisting of a raw byte stream with random data. Five MPEG-1 files with only video data were chosen (i.e., “elementary stream” MPEG-1 files), and five files with multiplexed video and audio data were chosen (i.e., “system stream” MPEG-1 files). The MPEG-1 files had a frame rate of 25 or 29.9 fps, bit rates ranging from 0.58 to 4.09 Mbps, file sizes between 3 MiB and 46 MiB; and contained between 29 and 302 I-frames.

For each data set, we took one of the MPEG-1 files, split a randomly (non-zero) sized portion of the file into chunks of predefined size  $s$ , and put these chunks into the data set in a non-consecutive order. The first group of five data sets  $G_{\text{elem}}$  comprised the elementary stream MPEG-1

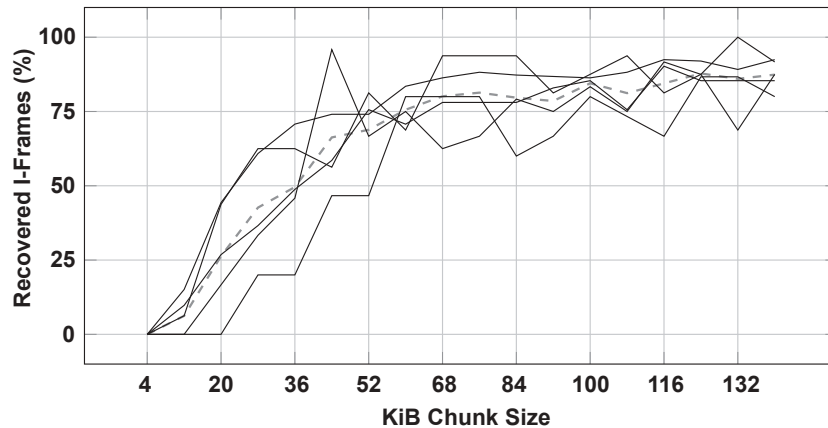


Figure 2. Percentage of completely recovered I-frames per chunk size for  $G_{elem}$ .

files and the second group of five data sets  $G_{sys}$  comprised the system stream MPEG-1 files. No MPEG-1 file was used twice. The resulting data set essentially corresponds to a hard drive image filled with random data and a single fragmented MPEG-1 file with  $s$ -sized fragments. The data set preparation was performed using standard Unix tools.

After the data sets were prepared, our MPEG-1 video frame carving prototype was used to search each data set for I-frames to be reconstructed. The number of I-frames that were completely or partly reconstructed was recorded. After each test run, the chunk size  $s$  was increased, and ten new data sets were prepared by filling them with random data and using the same set of MPEG-1 files. Then, the next test run was started. Eighteen different chunk sizes ranging from 4 to 140 KiB were used; thus, each data set was tested eighteen times.

The experimental results in Figures 2 and 3 show that the number of completely recovered I-frames grows with increasing chunk size, which is to be expected. Figure 2 shows that, for a mean chunk size of at least 36 KiB, nearly 50% (49.57%) of the I-frames could be completely recovered from  $G_{elem}$ . On the other hand, Figure 3 shows that a mean chunk size of 44 KiB was necessary to completely recover at least 50% (50.25%) of the I-frames from  $G_{sys}$ .

Figure 2 shows the percentage of completely recovered I-frames per chunk size for the five data sets in  $G_{elem}$  containing chunks of elementary stream MPEG-1 video files (mean values shown as a dashed line). Note that the highest mean rate of 87.8% completely recovered I-frames from  $G_{elem}$  was obtained for a chunk size of 124 KiB, which is roughly equal to the size of a GOP with two additional average-sized I-frames.

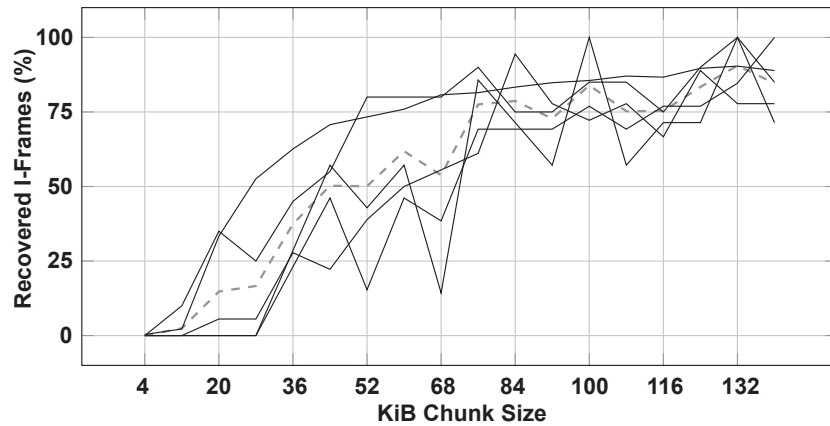


Figure 3. Percentage of completely recovered I-frames per chunk size for  $G_{\text{sys}}$ .

Figure 3 shows the percentage of completely recovered I-frames per chunk size for the five data sets in  $G_{\text{sys}}$  containing chunks of system stream MPEG-1 video files (mean values shown as a dashed line). Note that the highest mean rate of completely recovered I-frames was 90.55% for a chunk size of 132 KiB.

The different results obtained for  $G_{\text{elem}}$  and  $G_{\text{sys}}$  are due to the different file structures. The MPEG-1 files constituting the system stream contain more header information as well as audio data. Therefore, larger chunk sizes are needed to obtain better carving results.

The results show that I-frames from fragmented MPEG-1 files even with relatively small chunks (fragments) can be recovered completely or partially. Starting with a chunk size of 28 KiB, 20% or more of the I-frames were recovered completely or partially. Note that both the completely and partially recovered I-frames (which cause incorrect decoding results) are useful to evaluate the robust hashing method.

We also compared the performance of our MPEG-1 video frame carving implementation with Defraser [14], which works in a similar manner. For the comparison, we prepared six MPEG-1 video files (25 or 29.9 fps, 1.13 to 3.05 Mbps bit rate, 6 MiB to 46 MiB file size) and cut them into chunks of 4, 8, 16, 32, 64, 128 and 256 KiB. The chunks were then placed on a 1 GiB hard disk image file that was previously filled with random data.

Our prototype implementation and Defraser were able to recover 92% of the 361 I-frames. Of the recovered frames, both Defraser and our implementation were able to recover 82% I-frames completely and 18% partially. However, because Defraser relies on finding intact sequence headers, it was unable to recover I-frames with corrupted or missing

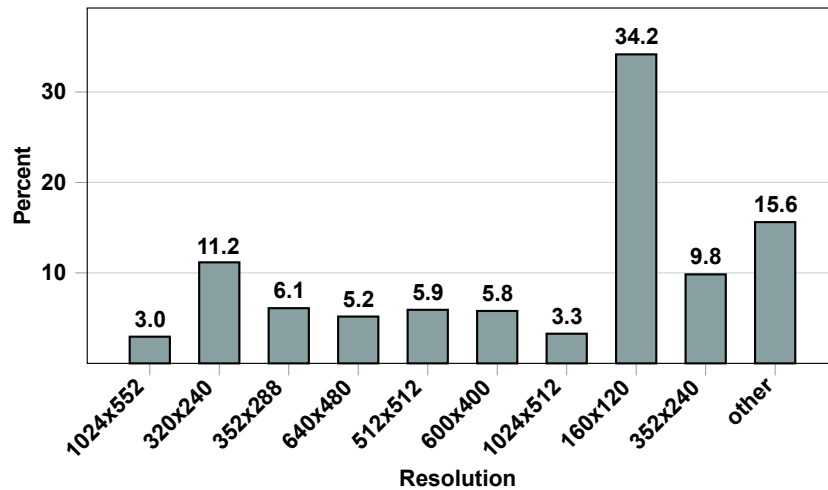


Figure 4. Resolution distribution of 5,489 randomly-downloaded MPEG-1 video files.

sequence headers. For such cases, we used the frame resolution correction procedure, which is described below.

## 7.2 Decoding (Resolution Correction)

A post-processing step is required to find the correct resolution when the resolution information of individual frames is missing. The post-processing was done as follows:

- Identification of Resolutions for Frame Decoding:** First, we identified the most commonly used resolutions of the MPEG-1 video format by collecting 5,489 different MPEG-1 videos from random sources using Google. Next, we extracted the resolutions that were used. We observed that nearly 85% of the video files used one of nine resolutions, including the SIF resolutions. Note that, according to the MPEG specification, MPEG-1 video data should be encoded using one of three SIF resolutions:  $352 \times 240$ ,  $352 \times 288$  or  $320 \times 240$  pixels. Figure 4 shows the resolution distribution of the MPEG-1 files. We believe that, together with the most commonly used MPEG-1 video resolutions, all the resolutions of frames found in the same raw data set would be suitable for individual frame decoding and subsequent resolution identification.
- Frame Decoding and Resolution Correction:** Using the resolutions that were determined to be suitable, we decoded each frame several times, each time with a different resolution. The results were stored as JPEG files. For each frame, we chose the decod-



(a) Original frame with incorrect resolution.



(b) Same frame with corrected resolution.

Figure 5. Frame decoding results before and after resolution correction.

ing result with the least amount of green blocks and applied our resolution correction approach. To evaluate the resolution correction approach, we used a test set of ten sample frames, where each frame was stored using 30 different resolutions. Upon applying the resolution correction, we were able to automatically produce decoding results with correct resolutions for about 80% of the JPEG files. Figure 5 shows an example of a frame decoding result with incorrect resolution and one with corrected resolution.

### 7.3 Identification (Robust Hashing)

With regard to content identification, we used our prototype implementation as reported in [15] to evaluate robust block hashing with a

test set containing simulated illegal images. Specifically, the test set contained 4,400 images of a cheerleader team, the notable characteristics being a large quantity of skin colors, similar poses in all the images, and the presence of one or more persons in all the images. The images were randomly divided into two equal-sized groups. One group was used as a test set to be recognized and was added to a robust hash database that already contained 88,000 robust hashes of other images. The second group of images was used as a test set to evaluate the false positive rate.

All 4,400 images were scaled down by a factor to ensure that the larger edge was only 300 pixels long; this corresponded to an average size reduction of 25% compared with the original images. Next, the images were horizontally mirrored and stored with a JPEG quality factor of 20, producing small images of low quality. This procedure simulates the strong changes that can occur during image conversion.

The two groups of images could be distinguished rather well using the Hamming distance between robust hashes as a feature to distinguish between known and unknown images. With a Hamming distance of 32 as the threshold, a false positive rate of 1.1% and a false negative rate of 0.4% were obtained. When using a Hamming distance of eight as the threshold, the false negative rate increased to 10% while the false positive rate dropped to 0%. Using both the Hamming distance and a weighted distance as proposed in [15] yielded even better results: for a Hamming distance of eight and a weighted distance of sixteen, the false positive rate dropped to 0% and the false negative rate fell to just 0.2%.

We also used our implementation to measure the time needed to create a robust hash of an image and to look up the hash in a database. We used the same image set of 4,400 images and the same robust hash database (88,000 plus 2,200 hashes). Our experiments revealed that each image was processed in roughly 10 ms, including robust hash generation and database lookup. We also observed that the speed could be increased up to factor of four when hard drive caching was used. These results are comparable with those obtained when computing cryptographic hashes (e.g., SHA-1) of the image files.

## 8. Conclusions

Combining robust video file recovery, robust frame decoding and robust video content identification into a single automated procedure can speed up forensic investigations. The prototype implementation described in this paper can recover and decode single MPEG-1 I-frames in the presence of small amounts of fragmentation. Additionally, ro-



bust block hashing can help recognize known image material, such as recovered and decoded I-frames, with low error rates and computational overhead.

The prototype implementation provides good results with regard to recovery, decoding and identification. Most video file carving approaches do not work well in the presence of fragmentation. Additionally, unlike our prototype, most carving tools do not have integrated I-frame decoders. The Defraser tool [14] can decode and visualize I-frames, but fails if just a few bytes of data (e.g., sequence header) needed for decoding are not available. Further research is necessary on reliable fragmentation point detection for multimedia file formats. Pal, *et al.* [10] have proposed a promising approach for text and image recovery, especially for JPEG files. However, fragmentation point detection approaches have yet to be developed for a wide range of video and audio file formats.

Robust hashing clearly improves video content identification mechanisms such as cryptographic hashing, which are prone to fail if the data to be hashed has been manipulated even slightly. Also, block-wise cryptographic hashing as proposed in [4] can help identify known video content, especially single frame fragments that cannot be decoded anymore. However, since the corresponding reference database can be very large, it is necessary to explore approaches for trading-off fragmentation robustness, database size and lookup speed.

## Acknowledgement

This research was supported by the Center for Advanced Security Research Darmstadt (CASED), Darmstadt, Germany.

## References

- [1] E. Allamanche, J. Herre, O. Hellmuth, B. Froba, T. Kastner and M. Cremer, Content-based identification of audio material using MPEG-7 low level description, *Proceedings of the Second International Symposium on Music Information Retrieval*, 2001.
- [2] J. Fridrich and M. Goljan, Robust hash functions for digital watermarking, *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 178–183, 2000.
- [3] S. Garfinkel, Carving contiguous and fragmented files with fast object validation, *Digital Investigation*, vol. 4(S), pp. S2–S12, 2007.
- [4] S. Garfinkel, A. Nelson, D. White and V. Roussev, Using purpose-built functions and block hashes to enable small block and sub-file forensics, *Digital Investigation*, vol. 7(S), pp. S13–S23, 2010.

- [5] C. Grenier, PhotoRec ([www.cgsecurity.org/wiki/PhotoRec](http://www.cgsecurity.org/wiki/PhotoRec)), 2007.
- [6] J. Haitsma, T. Kalker and J. Oostveen, Robust audio hashing for content identification, *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pp. 117–124, 2001.
- [7] H. Lejsek, A. Johannsson, F. Asmundsson, B. Jonsson, K. Dada-son and L. Amsaleg, Videntifier forensics: A new law enforcement service for the automatic identification of illegal video material, *Proceedings of the First ACM Workshop on Multimedia in Forensics*, pp. 19–24, 2009.
- [8] N. Memon and A. Pal, Automated reassembly of file fragmented images using greedy algorithms, *IEEE Transactions on Image Processing*, vol. 15(2), pp. 385–393, 2006.
- [9] J. Oostveen, T. Kalker and J. Haitsma, Visual hashing of video: Application and techniques, *Proceedings of the Thirteenth IS&T/SPIE International Symposium on Electronic Imaging, Security and Watermarking of Multimedia Contents*, vol. 4314, 2001.
- [10] A. Pal, H. Sencar and N. Memon, Detecting file fragmentation points using sequential hypothesis testing, *Digital Investigation*, vol. 5(S), pp. S2–S13, 2008.
- [11] G. Richard III and V. Roussev, Scalpel: A frugal, high performance file carver, *Proceedings of the Fifth Annual Digital Forensics Research Workshop*, 2005.
- [12] M. Rogers, J. Goldman, R. Mislán, T. Wedge and S. Debroya, Computer Forensics Field Triage Process Model, *Proceedings of the Conference on Digital Forensics, Security and Law*, pp. 27–40, 2006.
- [13] SourceForge.net, Foremost([foremost.sourceforge.net](http://foremost.sourceforge.net)).
- [14] SourceForge.net, NFI Defraser ([sourceforge.net/projects/defraser](http://sourceforge.net/projects/defraser)).
- [15] M. Steinebach, H. Liu and Y. Yannikos, Forbild: Efficient robust image hashing, *Proceedings of the SPIE Conference on Media Watermarking, Security and Forensics*, vol. 8303, 2012.
- [16] S. Vimal, Introduction to MPEG Video Coding, Lectures on Multimedia Computing, Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani, India, 2007.
- [17] B. Yang, F. Gu and X. Niu, Block mean value based image perceptual hashing, *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 167–172, 2006.

- [18] B. Yoo, J. Park, S. Lim, J. Bang and S. Lee, A study of multimedia file carving methods, *Multimedia Tools and Applications*, vol. 61(1), pp. 243–251, 2012.
- [19] X. Zhou, M. Schmucker and C. Brown, Video perceptual hashing using interframe similarity, *Proceedings of the 2006 Sicherheit Conference*, pp. 107–110, 2006.