Identification and recovery of video fragments for forensics file carving

Khawla Alghafli
Department of Electrical and Computer Engineering,
Khalifa University, Abu Dhabi, UAE
khawla.alghafli@kustar.ac.ae

Thomas Martin
Department of Computing and Mathematics,
Manchester Metropolitan University,
T.Martin@mmu.ac.uk

***Abstract*—In digital forensics, file carving of video files is an important process in the recovery of video evidence needed for many criminal cases. Traditional carving techniques recover video files based on their file structure. However, these techniques fail in cases where the file is split into several fragments, especially if some of the fragments were overwritten. In this paper, we present a method for identification and recovery process of video fragments if the video Codec specifications were overwritten. It consists of two parts which are detector and validators. The detector looks for sequences of bytes that could be video fragments in forensics image. The validator decides to accept or reject that a given fragment is a part of a video file. Based on the proposed method we implement a prototype which is called VidCarve. We have conducted several experiments to evaluate the proposed method with current video carving tools. Experimental results show that the discussed method can identify video fragments with high rates of precision and recall. The overall performance rate can produce forensically sound evidence and play a vital role in the process of recovery of digital evidence in many criminal cases.**

***Keywords*—Digital forensics, data recovery, file carving, fragmented video files, fragment identification.**

I. Introduction

Over the last few decades, the number of digital devices in use has increased. These devices are involved in most aspects of peoples' daily lives, many crimes have adapted to take advantage of these devices. These crimes include child abuse, counterfeiting, domestic violence, identity theft and economic fraud [1]. Digital forensics is the process of preservation, acquisition, examination, analysis and presentation of digital evidence from any digital device [2]. The aim of digital forensics is to find the evidence of computer crimes that can be accepted in court.

Data recovery is a primary element of digital forensics. There are cases where digital forensics researchers and practitioners need to recover files from a system which has a corrupted, overwritten or an unknown file system. File carving is a data recovery technique that recovers files from storage media based on the file content and structure without using file system metadata [3].

File carving is a powerful technique in digital forensic cases because it offers the flexibility of being able to recover information stored on digital media independent of the type of the underlying file system. Video content available in digital storage media of digital devices is widespread and growing exponentially. These devices include mobile phones, cameras, and surveillance systems. Many criminal cases include the recovery of deleted or corrupted video files. The aim of this research is to recover video files from unallocated disk space. We will make the use of the file content and structure rather than relying on the file system metadata to identify and reassemble a video file's blocks.

Operating systems allow users of the digital devices to create, modify, access and delete files. The important tasks of the file system management module in the operating system are storing of the files in the system and locating them we they are in need. The storing of the file can be done in two ways depending on the available free space. They can be stored either in contiguous memory locations or non-contiguous memory locations (the latter being called a fragmented file). There are many techniques available that can recover files that were stored in contiguous memory locations, but it is still a challenging problem in the area of file carving to recover fragmented files. In this research we will focus on solving the problem of recovering video file fragments without making use of file system metadata.

II. Previous work

The primary concept of file carving is that it recovers files without using the file system metadata. Many different file types have fixed patterns that always occur at the start and/or end of the file: so called headers and footers. The first carver which was published in the literature was Scalpel [4]. This carver requires a database of headers and footers for specific file types. The carver then retrieves files by searching for the pattern of a header and marks it as the starting point of a file. It then searches for the pattern of the corresponding footer. The entire sequence of bytes between the header and the footer is carved as a file. In the cases where some file types do not contain a footer, it uses the file size to determine the end of file. In the specific case of video files, if the file is not fragmented then Scalpel will likely be able to carve it as most video files have known headers. However, the main issue with the Scalpel is that it assumes that the file is not fragmented i.e. the bytes of the file exist in sequential order between the header and the footer. Whereas in practice, the actual file may not exist in contiguous, sequential order. If that is the case, the carved file will not be completely correct.

Most file systems try to store files in contiguous clusters. However, fragmentation is inevitable as files are created, modified and deleted in storage media. The literature includes several attempts to recover image files. One of the approaches that can be used to recover fragmented image files is the Bifragment Gap Carver [3]. This approach works on the scenario where the files are fragmented on two fragment and the file has a known header and footer. To carve a file in this scenario, the carver must be able to identify the additional data between the first and second fragments. All combinations of the gap between the first and second fragment are tested until the validation test of a file pass. This approach can be applied to recover fragmented video files by testing all combinations of the gap between first and second fragment to the video decoder. If the decoder decodes the data successfully, it means we set the correct gap. However, this approach has a limitation is that it cannot carve nonlinear fragmentation. It assumes the sequence of the fragments is the same in the original file. Another limitation is that it will not carve files that are split into more than two fragments.

Pal and Memon proposed a general framework that can perform recovery of fragmented files without restrictions in the number of the fragments per file [5]. This framework consists of three main stages: preprocessing, collation and reassembly. The framework can be applied to the carving of video files where methods and algorithms for each phase are need to be proposed to handle video fragments.

Also, Na et al. proposed a frame-based technique to recover video fragments (a frame being the smallest meaningful unit of a video file) [6]. Their technique consists of two main steps: identification and connection of video frames. The video frames were identified based on the start code, such as 0x00000001 for H.264 bitstreams. Then, the identified frame is validated by decoding it using the corresponding decoder and decoding header. The process of connecting valid frames relies on the information on frame length that is stored in a STSZ box of the video container. In the cases where the STSZ box was corrupted or overwritten, they proposed a method to connect two set of frames according to the cluster size. If the two connected sets were not verified by the decoder, they again expand the end of the first set or start of the second set according to the cluster size until the new set is verified by the decoder. They prevented unlimited expansion by using a threshold. The main limitation of their approach was that in the case of STSZ box was corrupted or overwritten, they cannot connect all the video fragments. Their proposed method only connects two fragments. As a result, it will return a considerable number of playable fragments. These fragments need more efficient reassembling techniques to produce playable video files rather than playable fragments.

Park and Lee proposed a procedure for DVR fragment forensics [7]. The procedure consists of five main steps: preprocessing, classification, reassembly, extraction and post-processing. The classification step consists of snipping the decoding information and the video frames in the same manner that were proposed by Na et al. [6]. The video files are constructed based on metadata stored in the video frames, such as timestamp, GPS location, Camera number or speed. They present a way to sort the extracted video frames based on the above properties. These frames will be converted to images and these images will be used to construct a playable video file. The metadata of the video frames that were proposed to be used to construct video files are not always available in the the video frames. Also, these metadata could be identical between hundreds of video frames.

Lewis developed a defragmentation algorithm and tool for H.264/AVC bitstreams that takes as input the forensics image and user specified bitstream syntax description [8]. The syntax description is used to generate a syntax checking parser. The role of the parser is to determine if vectors of data in the image are part of the bitstream, through the use of the decoder parameters. Finally, they used a special depth-first search to reassemble validated H.264 bitstream fragments in the correct sequence. This will produce a complete and error free video file with the knowledge of initial and final block index in the bit streams. They chose the permutation that maximizes the number of the bitstreams that can be decoded without error. The disadvantage of their approach is that the time cost is high. The algorithm has to parse many disk blocks when searching for a sequence of blocks that yield a valid bitstream. Their objective is to construct one chain that produces an error free video file. Their approach decoded many permutations of recovered bitstreams that yielded an error free chain. This needed a high number of decoding trials. Moreover, in real cases, if there are any overwritten or missing parts the chain cannot be constructed. Also, the forensics image may contain multiple video bitstreams, so the objective should be to produce multiple chains of the available video bitstreams.

Beside these, Casey and Zoun explained several practical lessons that were learned from extensive working experience with the problem of file carving of fragmented video files [9]. They provided recommendations for several trade-offs that can be considered in the development of new video carving tools. They mentioned that there is no single approach that has ability to carve fragmented video files for all cases. The forensic analyst should choose the most effective approach for a given case.

III. Background of video file structure

Nowadays, video files are stored in multimedia file containers. This is because a multimedia file container can hold different data types and define how these data types can co-exist. The simple structure of a container can hold interleaved audio and video streams along with their codec specification parameters. More advanced multimedia file container structure can hold more information such as multiple video or audio streams, subtitles and tags. An example of a multimedia file container is the ISO Based Media File Format (ISO BMFF) [10]. The basic structure of a video file is shown in Fig.1.

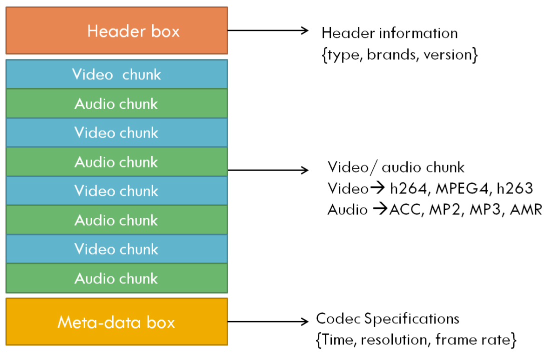

Figure. 1. The structure of video file

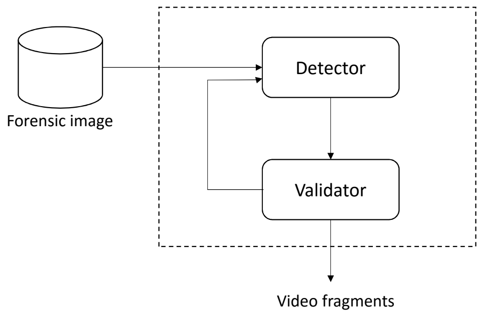

Figure. 2. The process of the identification and Recovery of video fragments

MP4 and 3GPP video files are the most common file formats used to save recorded video in smartphone devices and personal cameras to record video. These two file formats are contained in ISO BMFF, which makes this container a popular multimedia container. ISO BMFF can be defined as the structure that can hold timed media information in a flexible and extensible format that can be easily interchanged, managed and edited [11]. The structure of an ISO BMFF file is object oriented. This means that the file can be broken down into several objects, or boxes as they are called in this standard. The structure of each object can be known from its type. All data in the media file are contained in boxes. No other data stored in a media file are self-contained. The detailed structure of all boxes that ISO BMFF container can contain is illustrated in ISO/IEC 14496-12 standard [11].

All boxes start with a header which contains the size and the type of the box. The size reflects the total size of the box which includes the header, fields, and all contained sub boxes. This simplifies the job of parsing the file. Some of the ISO BMFF boxes are optional and others are mandatory. There are two main mandatory boxes which are file type (ftyp) and movie (moov) boxes. The ftyp box holds header information such as file type, compatible brands and versions. The moov box includes the metadata information of a media file plus codec initialization parameters such as resolution and frame rate.

The media data (mdat) box is not a mandatory box in the ISO BMFF structure. However, it is a very important box because it holds all video and audio streams. The video data is stored in this box as interleaved video and audio chunks of data. Each chunk consists of a number of video/audio samples. According to the ISO BMFF standard [10], a sample is defined as the data that is associated with a single time stamp. Thus, the sample in a video chunk of ISO BMFF corresponds to a Network Abstraction Layer (NAL) unit of H.264. In the H.264 standard, the video frame is stored in a NAL unit.

ISO BMFF specifies how video/audio data can be stored within the mdat box. However, it does not define how data is encoded and how to decode it. To parse the content of this box, we have to know the type of codec and codec parameters that can be used to decode video or audio streams. This information can be found in the moov box.

IV. Identification and Recovery of Video Fragments without using Video Codec Specifications

Identification is the first step in the process of carving video files. When fragmentation occurs, the data related to one file are stored in different non-contiguous memory locations. Header-footer or header-maximum file size carving techniques cannot be applied in this case, nor can file structure based techniques. We propose an identification and recovery process of video fragments illustrated in Fig. 2 . This process is divided into two stages, namely the detector stage and the validator stage. The detector searches the forensic image for sequences of bytes that could be video fragments. The validator makes the decision to accept or reject that a given fragment is a part of a video file.

Sections IV-A and IV-B will discuss the detector and validator respectively in more detail.

A. Detector

Compressed video bitstreams are difficult to detected based on the content alone without using the meta-data from the file system or codec specifications. This is because compressed video bitstreams are generated by algorithms that remove redundancy. This raises a problem since it means that the data in the fragment is difficult to distinguish from random noise.

Our detector will identify video fragments without using the codec specifications. This method applies in the cases where codec specifications are overwritten. In other words, in this case there will be no meta-data box to figure out the samples sizes which are usually used to identify the header length. We are going to use the redundancy that is added by the syntax description. Moreover, we are going to recover video fragments based on the format of video NAL unit. This approach assumes that each video fragment consists of two or more video NAL units. Our approach will search for the header of the video NAL unit rather than the header length or NAL unit start code prefix of H.264 bitstream.

The NAL unit header consist of 8 bits which include one bit for forbidden_zero_bit, two bits for nal_ref_idc and 5 bits nal_unit_type. Fig. 3 shows the basic structure of the

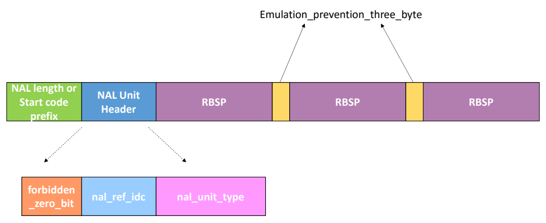

Figure. 3. The structure of H.264/AVC NAL unit

TABLE I
NAL UNITS TYPES OF H.264 DATA STREAM

Content of NAL unit	forbidden_zero _bit (1 bit)	nal_ref_idc (2 bits)	nal_unit _type (5 bits)	NAL unit header (8 bits)
Coded slice of a non-IDR picture	0 b	01 b	00001 b	00010101 b
	0 b	10 b	00001 b	00101001 b
	0 b	11 b	00001 b	00111101 b
Coded slice of an IDR picture	0 b	01 b	00101 b	00011001 b
	0 b	10 b	00101 b	00101101 b
	0 b	11 b	00101 b	01000001 b

H.264/AVC NAL unit. Algorithm 1 presents the Acquisition process of the NAL unit of H.264/AVC. This algorithm is based on NAL unit syntax of H.264 [12]. Table I illustrates the NAL unit types of H.264 stream [12].

Algorithm 1 Acquisition of the NAL unit of H.264/AVC

```
1: if (forbidden_zero_bit || nal_ref_idc || nal_unit_type) = found) then
2:     Set NumBytesInRBSP = 0
3:     for $i = 0, i < NumBytesInNALunit, i++$ do
4:         if $i + 2 < NumBytesInNALunit$ AND $next\_bits(24) == 0x000003$ then
5:             Get $rbsp\_byte[NumBytesInRBSP++]$
6:             Get $rbsp\_byte[NumBytesInRBSP++]$
7:             $i += 2$
8:             Get $emulation\_prevention\_three\_byte$
9:         else
10:            Get $rbsp\_byte[NumBytesInRBSP++]$
11:        end if
12:    end for
13: end if
```

The detector will search for the NAL unit header identifier from the first to the last location in the forensic image in one sequential pass. The search step is one byte since the length of NAL unit header is one byte.

B. Validator

In order to recover fragments from a forensic image, it is important to have a process to validate the recovered bytes. The job of validator is to determine whether a sequence of bytes is part of a video file or not.

Once the detector finds a suspected NAL unit header identifier, it will get the NAL unit size by shifting back four bytes. Then, it will jump to the end of this sample plus four. At this position, there should be the NAL unit header identifier of another video NAL unit. If no such header is found, the conclusion would be that the sequence of bytes was not a video NAL unit as it is rare to a find fragment of only one NAL unit. The validator would reject this NAL unit and the detector would continue the searching process after the location of the rejected NAL header. If a NAL Unit header was found, the validator accepts these two NAL units and we will continue jumping to the next NAL units according to each of the NAL unit lengths. Algorithm 2 describes this process in detail.

Algorithm 2 Identification and recovery of video fragments of H.264/AVC data bitstreams

```
1: Set curr_loc =0
2: while $(curr\_loc < img\_length)$ do
3:     Set NAL_count = 0
4:     Set start_loc=-1
5:     Set end_loc=-1
6:     Search NAL_Unit_header
7:     if $(NAL\_Unit\_header = found)$ then
8:         Shift back 4 bytes
9:         Get NAL_length
10:        Go to curr_loc + length + 8
11:        while $(NAL\_Unit\_header = found$ and $curr\_loc < img\_length)$ do
12:            Shift back 4 bytes
13:            Get NAL_length
14:            if $(NAL\_count == 0)$ then
15:                NAL_count=2;
16:                start_loc = curr_loc - NAL_length - 8
17:            else
18:                NAL_count++
19:                end_loc = curr_loc + length + 4
20:            end if
21:            Go to curr_loc + length + 8
22:        end while
23:        if NAL_count>=2 then
24:            Carve fragment from start_loc to end_loc
25:        end if
26:    end if
27: end while
```

V. EXPERIMENTS AND RESULTS

This section presents experimental results with discussion of the results.

A. Datasets

Existing datasets in the area of digital forensic investigation are usually in the form of a forensic image of a target device. For instance, a bit by bit copy of hard disk or SD memory. The process of evaluating the identification and recovery of video fragments depends on the detailed knowledge of memory locations that are occupied by each file. We need to run the experiment on an image where we know exactly where all the fragments of all the video files are in advance , but the tool is executed without this information. This is essential in order to calculate how well the tool performed recovery of video fragments.

To evaluate the performance of the identification and recovery of video fragments, there are a total of five datasets that were used in the testing process of software applications with file carving capabilities by NIST [13]. NIST is in the process of developing Computer Forensic Reference Data Sets (CFReDS) for testing and examining forensics tools such as video file carving tools. The objective of constructing these datasets and making them available to the public is to increase confidence in the capability of available tools. However, none of these 5 datasets have a video file based on the H.264 bitstream with the codec specification part overwritten. These datasets have only one MP4 file based on the H.264 bitstream, fragmented into two or three fragments with the codec specification intact. This is the reason why these datasets are not suitable for our experiments.

Due to the lack of availability of public datasets with heavily fragmented and partially overwritten video files, datasets were specially constructed. We created 5 disk images with 256 MB each. These images had no file system meta data. The video files were captured using different digital devices and saved in different file types. The files were fragmented into between 2 and 274 pieces. Each disk image contained 10 video files and 10 other non-video files. The other files included jpg, pdf, doc and ppt. The meta-data box of each video file in the datasets was overwritten. The details of each of these datasets is illustrated in Table II.

B. *Video file carving tools*

We implemented a prototype of our proposed method, called VidCarve, written in the C++ programming language. VidCarve and other video carving tools were tested using the datasets described in Section V-A.

There are several video file carving tools that can be used to identify and recover video fragments. Three video carving tools were selected to be included in this evaluation process. The selection criteria of these tools were tool availability, support for the recovery of video files based on the H.264 Codec, ability to carve fragmented video files, ability to carve partially overwritten video files based on the H.264 Codec. Table III list the tools that were included in the evaluation process.

C. *Result*

In this section, we will present the results of our experiment. The datasets had a known layout, seeing as we generated them. The video carving tools were tested using these datasets. The resulting outputs of the testing were classified according to the numbers of true positives, false positives and false negatives. From these, the recall, precision and $F_{measure}$ rates were calculated.

The basic way to check if the identified fragment is positive result is to check if there is exact match between its hash value

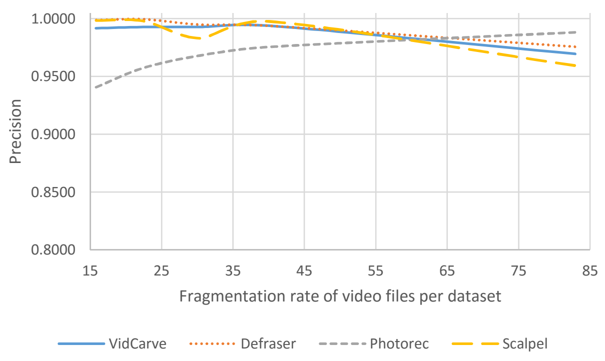

Figure. 4. Fragmentation Rate VS Precision of video carving tools

against one of the hash values of the fragments that exist in this dataset. As the video carving tools recover the video fragments with missing bytes at the beginning or end of video fragments, there will be no match between the identified fragment and its corresponding fragment in the dataset. Although the content of the identified fragment and its corresponding fragment in the dataset are more than 90% similar, using hash function will indicate it as a false result. Thus, the identified fragment will be split into smaller pieces to $F_{measure}$ performance. We choose to count true positive, false positive and false negative based on small unit of video fragments which is the NAL unit. Thus, we are going to check the hash value of each NAL unit in the recovered fragment against the hash values of NAL units that exist in the dataset.

Fig. 4 illustrates the relation between the fragmentation rate of video files in the five datasets verses the precision of the carving tools that were included in this experiment. All the tools had a high precision ratio, which was always more than 0.94. The values of the precision of all the tools were relatively stable, ranging from 0.999 to 0.940.

Fig. 5 shows the relation between different fragmentation rate of video files in the five datasets verses the recall of the carving tools. As can be seen from the graph, the recall of Defraser was relatively stable as the fragmentation rate varied. VidCarve shows mostly the same performance as Defraser, except when testing done using dataset 2. The recall rate of VidCarve was always above 0.85 except for dataset 2, where it was 0.48. Photorec and Scalpel have a lower recall for all the different fragmentation rates, scoring less than 0.5 in all cases.

Fig. 6 summarizes the overall experimental results of evaluating the identification and recovery of video fragments using different video carving tools. All the tools that were included in this experiment show approximately similar precision, with Defraser getting the highest. There are noticeable differences between the recall of each of the carving tools. Defraser was the highest, followed by our Vidcarve prototype. The recall of both Photorec and Scalpel were less than 0.50. According to $F_{measure}$ value we can rank the tools that were included in this experiment from highest to lowest as follows: Defraser,

TABLE II
DATASETS DETAILS OF FRAGMENTED AND PARTIALLY OVERWRITTEN VIDEO FILES

ID	Total No. of files	No. of video files	Avg file size	Avg video file size	Total no. of fragments	Total no. of video fragments	Avg fragment per file	Avg fragments per video file
1	20	10	5,853	11,350	199	158	9.95	15.8
2	20	10	9,130.56	16,805.70	420	206	21	22.4
3	20	10	12,111	23,657	406	318	20.3	30.4
4	20	10	9,676	18,921	519	409	25.95	40
5	20	10	12,158	23,886	728	610	36.4	82.9

TABLE III
VIDEO FILE RECOVERY TOOLS DETAILS

Name	Version	Licensee	Operating system
Defraser	1.4.2	Proprietary	Windows
Scalpel	1.60	Open source	Linux, Windows, Mac OS X
Photorec	7.0	Open source	Linux, Windows, Mac OS X

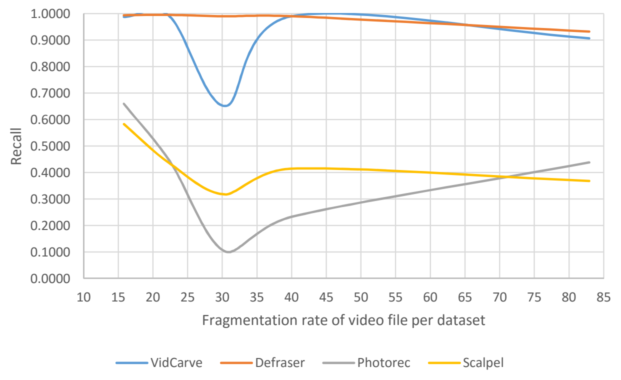

Figure. 5. Fragmentation Rate VS Recall of video carving tools

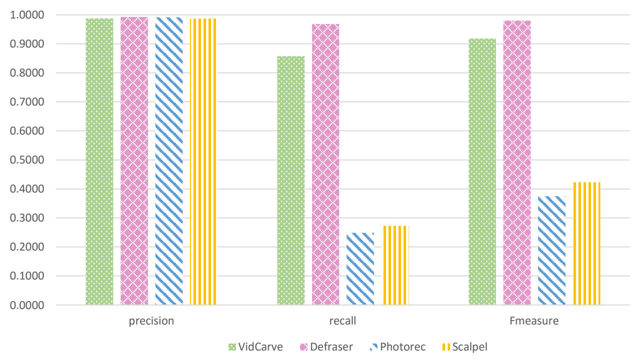

Figure. 6. Experimental results of evaluation recovery process of video carving tools

VidCarve, Scalpel and Photorec. The average $F_{measure}$ values of Defraser, VidCarve, Scalpel and Photorec were 0.9804, 0.9186, 0.3752 and 0.4237 respectively.

From the above results, all the tools obtained a high precision rate. This means that they are capable of avoiding false positive results. However, the recalls were less, which means that they missed identifying some of the available NAL units in the forensics image. Although Defraser obtained the highest $F_{measure}$, precision and recall with a difference around 7% in the $F_{measure}$, VidCarve is an open source algorithm. It is available for researchers for enhancement and improvement, whereas Defraser is closed source tool.

VI. CONCLUSION

File carving is an essential part of forensic data recovery in any cases involving deleted data, corrupted file systems, reformatted file systems or unknown file systems. The problem with existing carving techniques is that they do not work well with fragmented files. From the literature, the recovery of video fragments depends on the availability of video Codec specifications. If this part of video file were overwritten, existing techniques cannot recover video fragments. The amount of digital video content is increasing exponentially. There is a need for methods that can carve fragmented video content.

We conducted several experiments to evaluate our proposed algorithm against other video carving tools using 5 datasets. Our experiments have shown that the precision of our prototype and video carving tools were high. However, the recall values were not. Thus, more research is needed to reduce number of false negative results.

REFERENCES

[1] N. I. of Justice, U. D. Justice, and O. o. J. Programs, "Electronic Crime Scene Investigation: An On-the-Scene Reference for First Responders," 2009.

[2] W. Jansen and R. Ayers, "Guidelines on cell phone forensics," *NIST Special Publication*, vol. 800, p. 101, 2007.

[3] S. L. Garfinkel, "Carving contiguous and fragmented files with fast object validation," *digital investigation*, vol. 4, pp. 2–12, 2007.

[4] G. G. Richard III and V. Roussev, "Scalpel: A frugal, high performance file carver.," in *DFRWS*, 2005.

[5] A. Pal and N. Memon, "The evolution of file carving," *Signal Processing Magazine, IEEE*, vol. 26, no. 2, pp. 59–71, 2009.

[6] J. Lee, G. Na, K. Shim, K. Moon, S. Kong, and E. Kim, "Frame-based recovery of corrupted video files using video codec specifications," 2014.

[7] J. Park and S. Lee, "Data fragment forensics for embedded dvr systems," *Digital Investigation*, vol. 11, no. 3, pp. 187–200, 2014.

[8] A. B. Lewis, *Reconstructing compressed photo and video data*. PhD thesis, University of Cambridge, 2012.

[9] E. Casey and R. Zoun, "Design tradeoffs for developing fragmented video carving tools," *Digital Investigation*, vol. 11, pp. S30–S39, 2014.

[10] "ISO/IEC 14496-12 Information technology – Coding of audio-visual objects – Part 12: ISO base media file format," 2012.

[11] "ISO/IEC 14496-15 Information technology – Coding of audio-visual objects – Part 15: Advanced Video Coding (AVC) file format," 2010.

[12] " ISO/IEC 14496-10 Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," 2012.

[13] J. R. Lyle, D. R. White, and R. P. Ayers, "Digital forensics at the national institute of standards and technology," *National Institute of Standards and Technology, Interagency Report (NISTIR)*, vol. 7490, 2008.