Video Forensic of Fragmented Video Based on H.264/AVC Video Compression Standard

Kang Sheng
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China
jokeforce@gmail.com

Jiaqing Qu
Shanghai Radio Equipment Research Institute
Shanghai, China
qujiaqing@hrbeu.edu.cn

Xinyi Liao
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China
suiyi528@163.com

Yu'an Tan
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China
victortan@yeah.net

Quanxin Zhang
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China
zhangqx@bit.edu.cn (corresponding author)

Abstract—Traditional video forensics were just for complete video files, which aim at reconstructing the processing history of the video data and validating their origins or authenticity. They have obtained great achievements. However, we cannot always get complete video file in practice, sometime we only get part of it. In this paper, we came up with a method that can restore images of IDR frames from fragmented video files. After analyzing the format of MP4 and H.264/AVC Compression Standard, we proposed an algorithm to search for valid slices from fragment video files. With these slices, images of IDR frames can be restored by reconstructing sequence parameter set and picture parameter set. Test results show that images can be successfully restored in most cases except that FMO or data segmentation is adopted in the process of encoding.

Keywords- video forensics; fragmented video; H.264/AVC; restore image; reconstruct SPS and PPS

I. INTRODUCTION

In the recent years, the availability of inexpensive, portable, and highly usable digital multimedia devices has increased the possibility of generating digital audiovisual data without any time, location, and network-related constraints. In addition, the versatility of the digital support allows copying, editing, and distributing the multimedia data with little effort. As a consequence, the authentication and validation of a given content have become more and more difficult, due to the possible diverse origins and the potential alterations that could have been operated. At the same time, a significant research effort has been recently devoted to the forensic analysis of multimedia data [1-3]. A large part of the research activities in this field are devoted to the analysis of still images. At present, researches on video forensics are mainly concentrated on the following three methods: (1) Forensic tools for video acquisition analysis [4-5]; (2) Forensic tools for video compression [6]; (3) Forensic tools for video doctoring detection [7-9]. These methods have all achieved good results, however, their objects are complete video files, and we often find that video files have been deleted by parties and would not be restored in actual video forensics, which brings great difficulties to the forensic work. This paper just focused on video forensic of incomplete files in the disk and satisfactory results were achieved by restoring images of IDR frame from fragmented video through the reconstruction of SPS (sequence parameter set) and PPS (picture parameter set). We select H.264/AVC Video Compression Standard as the standard video encoding method, because H.264 is currently the most commonly used formats for the recording, compression, and distribution of video content [11].

II. ACQUIRE SLICES FROM FRAGMENTED VIDEO

A. Cluster

The sector is the smallest unit of physical storage in the disk, but the operating system fails to address a large number of sectors, so it combines contiguous sectors together to form a cluster, and then it manages these clusters. Therefore, the cluster is the basic unit of disk file storage management in the operating system. The number of sectors in a cluster is decided by the file system format and the allocated unit size. Generally, a cluster may include 2, 4, 8, 16, 32 or 64 sectors.

The operating system stipulates that only a file can be stored in one cluster for a more efficient management of

disk space, therefore, the file in the disk is consisted of continuous or dispersed clusters, but sectors within any cluster must be continuous and only belong to this file.

So, as long as there is a complete video file's cluster, forensic analysis can be carried out even if only part of the file fragments would be acquired.

B. Analysis of Video Format

The video format is numerous, but no matter in any format, a video file is essentially different tracks wrapped inside a container. Therefore, different video formats are just put in different containers, the core data of video bitstream is the same. The first step is to open the container so as to get the data of video bitstream inside, so we need to analyze each video format concretely.

MP4 (MPEG-4 Part 14) is a common multimedia container format, which is defined in the "ISO/IEC 14496-14" standard file and belongs to a part of MPEG-4. In this paper, we take MP4 format as an example to explain how to extract the bitstream data from videos.

MP4 is composed of "boxes" of different sizes, in which media information is stored by placing small-size boxes in large-size boxes. The basic structure of box is shown in Fig .1.

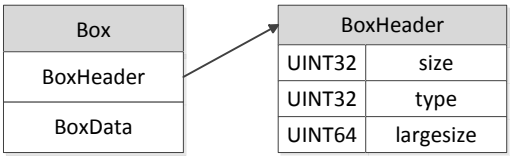

Figure 1. Structure of MP4 box

Among them, "size" specifies the space occupied by the whole box, including the header part. If the size of box is so large that exceeds the maximum number of uint32, "size" is set as 1, and the next 8 bit "largesize" is used to store the size.

As shown in Fig .2(a) that MP4 file consists of three large boxes, respectively are ftyp box, moov box and mdat box, which are used to indicate the file type, store the media information and store the media data separately.

The mdat box is the key box that we need to pay further attention, and bitstream data is stored in this box. Meanwhile, in most cases, video file fragmentation obtained is data from the mdat box. As shown in Fig .2(b), mdat box consists of "size" of 4 bytes, "type" of 4 bytes (namely "mdat") and "BoxData" of size-8 bytes.

The "mdat BoxData" consists of continuous slice data, among which the structure of slice is as shown in Fig .2(d), the actual data comes after the 4 byte's size.

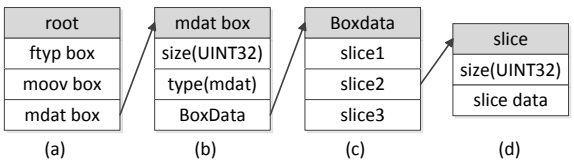

Figure 2. Structure detail of MP4 format

C. Extraction of Video Frame

In the H.264/AVC Video Compression Standard, an image is composed of N slice groups, and N=1 if FMO (Flexible Macroblock Ordering) mechanism isn't adopted. Each slice group is composed of one or several slices, and a slice is equal to a NALU if there isn't any data segmentation. Image is always decoded by independent slices, and then the decoded MBs (Macroblock) are reprogrammed into an image according to slice groups. So in that sense, the slice is the largest decode unit in practice [15].

Besides, FMO is only allowed within the Baseline and Extended profiles. The much more common Constrained Baseline, Main, and all High profiles do not support it, and software that can create or decode it is rare. Some video conferencing units use it; otherwise, the JM reference software is the primary support [10].

Therefore, under the condition of without considering the data segmentation, a full image may be restored if we can obtain several consecutive slices of a frame by searching in fragmented files with slice as the basic unit, otherwise part of the image may be restored.

The maximum size of a frame is 8000000 bytes according to the official documentation, so a single slice must be less than or equal to 8000000, and the "size" part of slice must be in the form of 0x00xxxxxx, namely the first byte is 0x00. The following algorithm is given to search for slices:

Setp1 Reads a byte, ch=read (1)
Setp2 Judge whether ch is equal to 0, if yes, turn step3, otherwise turn step1
Setp3 Read three bytes, size1=read (3)
Setp4 Read size1 bytes, nalu1=read (size1)
Setp5 Judge whether nalu1 is legal, if yes, turn step6, otherwise seek (-size1-3), and then turn step 1
Setp6 Read a byte, ch=read (1)
Setp7 Judge whether ch is equal to 0, if yes, turn step 8, otherwise seek (-size1-4), and then turn step 1
Setp8 Read three bytes, size2=read (3)
Setp9 Read size2 bytes, nalu2=read (size2)
Setp10 Judge whether nalu2 is legal, if yes, turn step11, otherwise seek (-size2-4-sie1-3), and then turn step1
Setp11 Succeed, both nalu1 and nalu2 are legal slices

To determine whether a nalu is legal is simply to check nal_header (the first byte of nalu). The structure of NAL_Header is as shown in Fig .3. The 'F' is the forbidden bit, it should be 0 and nal_unit_type must be in the range [0, 9], otherwise the nalu is illegal.

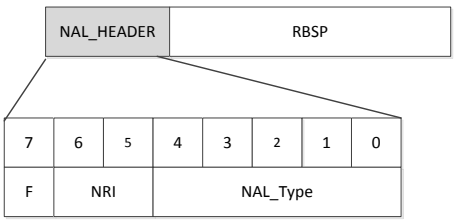

Figure 3. Structure Unit of NAL

III. RESTORE IMAGE FROM SLICES

We have already obtained slices that can be decoded independently in the sections above, however, only slice of I frame can restore images, because the encoding of I frame just depends on itself and is independent of other frames [12]. In this paper, we only deal with IDR frame for convenience's sake.

Firstly, we choose certain nalus of which nal_unit_type is equal to five in order to select IDR frames. And nal_unit_type is the last 5 bits of the first byte of slice as shown in Fig .3.

Next, we can start restore images with the bitstream data from slices of IDR frame that get from the original disk.

In fact, we can easily restore images by directly decoding IDR frames if we can obtain related SPS and PPS in numerous nalus, because the slice is decoding-independent. But in reality, we just fail to obtain the corresponding SPS and PPS frequently, therefore, we need to reconstruct SPS and PPS with a combination of various conditions in order to restore images.

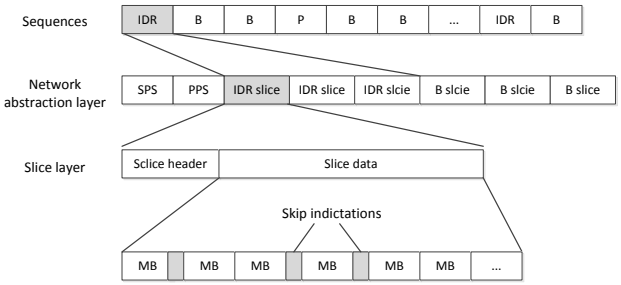

Figure 4. Hierarchy of video stream

A. *Reconstruct SPS*

On the basis of SPS syntax in Section 7.3 of Standards Documentation, a number of explorations and tests are made with a combination of JM source code and a set of effective reconstruct scheme of SPS is finally concluded in this paper as follows:

There are probably 40 parameters in SPS, so it is almost impossible for us to reconstruct all these parameters. However, our purpose to reconstruct SPS is simply to restore images of IDR frame, so a big part of parameters herein are not helpful for the decoding of IDR [13-14]. In this paper, parameters that need to be focused on are given.

1) profile_idc & level_idc: profile defines a set of coding tools or algorithms that can be used in generating a conforming bitstream whereas a level places constraints on certain key parameters of the bitstream. All decoders conforming to a specific profile must support all features in that profile. As the term is used in the standard, a "level" is a specified set of constraints that indicates a degree of required decoder performance for a profile. A decoder that conforms to a given level must be able to decode all bitstreams encoded for that level and all lower levels. So as to decode IDR frames, we can always set profile_idc to Main Profile and level_idc to max (51) which can support the maximum picture resolution, frame rate, and bit rate that a decoder may use.

2) seq_parameter_set_id: it specifies the ID number of SPS, set it as 0.

3) log2_max_frame_num_minus4: it is a parameter to read another syntax element of frame_num, which also indicates the maximum value of frame_num:

$$MaxFrameNum = 2^{log2\_max\_frame\_num\_minus4+4}$$

So this parameter can be calculated by the combination with the digit of framenum in slice header, e.g. a framenum is a four-bit binary figure like 0010 or 0001, then log2_max_frame_num_minus4 is 4-4=0.

4) pic_order_cnt_type: it specifies the coding method of POC (Picture Order Count) and the play order of POC logo image. But this value is irrelevant since we only decode IDR frames, than we set it as 0 for simplicity.

5) log2_max_pic_order_cnt_lsb_minus4: it indicates the value of the variable MaxPicOrderCntLsb:

$$MaxPicOrderCntLsb = 2^{log2\_max\_pic\_order\_cnt\_lsb\_minus4+4}$$

In most cases, it is set to log2_max_frame_num_minus4 + 2(or 1). If it is not correct, then other values should be tried.

6) num_ref_frames: it indicates the maximum length that may be achieved by the reference frame queue, which does little to help decoding IDR frames. Set it as 2 by default.

7) pic_width_in_mbs_minus1&pic_height_in_map_units_minus1: these are two important parameters, indicate the image width and the image height respectively, which are both calculated in Macroblocks. These two parameters are essential for decoding because they directly indicate the video resolution. Two methods can be used to obtain the two values, among which one method is to enumerate all mainstream resolutions violently, whose number is not very large, and there would certainly exists a decoding process that wouldn't cause any conflict after trying to use every possible resolutions. In that case, we find the values of these two parameters; another method doesn't require enumeration, but there are some limitation for it. We have mentioned above that each frame video is divided into N slices that can encode and decode independently, if N>1, then we can get the width and height according to first_mb_in_slice parameter in slice header. Here we show how we get it with the following example:

first_mb_in_slice is the first parameter of slice_header, we get a value of 00000000100111111 from a certain slice header. It is encoded by unsigned Golomb entropy encoding, which can be decoded into a decimal number of 318.

It means that the first MB of the slice is the 318th MB in the whole IDR frame, obviously, due to 318=53x6, the size of a single slice is 53x6. As for the number of slices, we can calculate it out by using the maximum value of first_mb_in_slice. Still in this example, the maximum value is 00000000010011111001, namely 1272, then we have 1272/328 + 1 = 5 slices, so the size of the complete IDR frame is 53x30, which can be transformed into the standard resolution of 848x480. In this way, the values of these two parameters we finally obtained are 52 and 29.

B. *Reconstruct PPS*

The parameters of PPS are less and simpler than these of SPS. Some important parameters are listed as follows:

1) pic_parameter_set_id & seq_parameter_set_id: the id of PPS and SPS, set both of them to 0.

2) entropy_coding_mode_flag： 0 stands for CAVLC entropy coding and 1 stands CABAC. Try both of them respectively.

3) num_slice_groups_minus1: it specifies the number of slice groups in an image, which is set as 0 if there is no

slice groups. The methods proposed in this paper would be useless if a slice group mode is adopted for video encoding.

4) num_ref_idx_l0_active_minus1&num_ref_idx_l1_active_minus1： these two permeants specify the length of the current reference queue, which is not so important. Set them to 1.

5) weighted_pred_flag: it is helpless for decoding IDR frames. Set it to 1.

6) pic_init_qp_minus26： it is a very important parameter which indicates the initial value of quantization parameter in luminance component. Its value range is [-26, +25]. Generally, we take 0, 1, 2 to try respectively, which can solve problems in most cases. If they don't work, the whole value range should be traversed, but it would be better if it is traversed from middle to endpoints.

There are some flag parameters of no great importance. Set them to 0.

IV. Experiment and Results

In the test, we copied multiple files to disk simultaneously, including some MP4 files. In this way, video clusters may distributed on the disk discontinuously. This was verified by WinHex, a tool that can view cluster distribution. Then we damaged some clusters.

We searched for slices in the disk using the algorithm mentioned above. Fortunately, all of the slices of IDR frames had been found, even including SPS and PPS which we must pretend not to find. By ignoring some parameters, guessing some parameters and calculating some parameters, SPS and PPS were reconstructed. Then we finally succeeded in restoring most images. In addition, the quality of images are not affected. However, many limitations exist in the methods proposed in this paper, mainly including the following four points:

1. The methods are invalid if FMO is adopted for encoding.
2. The methods are invalid for video with data segmentation.
3. It is difficult to use these methods to restore images if a resolution should be enumerated and the resolution of a video is just relatively rare.
4. At present, only IDR image can be restored, while nothing can be do with video sequence.

V. Conclusion

Using the method mentioned in this paper, we can search disks arbitrarily to find useful slices and restore them as images. It would be quite significant for video forensic.

At the same time, further improvement should be carried out based on the above four points, so as to allow our methods to deal with more videos. We made a little attempt for points four: if the disk media we have is TF card of a camera, because most encoding parameters of videos shot by a same camera are consistent, we can always easily find PPS or SPS from other videos in disk and use them directly by a slight modification or, if condition permits, we can use the original camera to shoot some videos and reconstruct a complete SPS and PPS by comprehensively applying these parameters, thus restoring the video sequence.

Acknowledgment

This work is supported by National Natural Science Foundation of China (No. 61300047), 863 Program (No. 2013AA01A212), Students' Innovative Plan of BIT (BJ1317), Shanghai Aerospace Science and Technology Fund (SAST201341).

References

[1] Venkatraman, D.; Makur, A.: A compressive sensing approach to object-based surveillance video coding, in Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2009), Taipei, Taiwan, April 19–24, 2009, 3513–3516.

[2] Wang, W.; Farid, H.: Detecting re-projected video, in Information Hiding, Lecture Notes in Computer Science, K. Solanki, K. Sullivan, and U. Madhow, eds., vol. 5284, Springer, Berlin 2008, 72–86.

[3] Poisel, R.; Tjoa, S.: Forensics investigations of multimedia data: a review of the state-of-the-art, in 2011 Sixth Int. Conf. on IT Security Incident Management and IT Forensics (IMF), Stuttgart, Germany, May 10–2, 2011, 48–61.

[4] Lin, Y.-C.; Varodayan, D. P.; Girod, B. Image authentication using distributed source coding. IEEE Trans. Image Process., 21(1) (2012), 273–283.

[5] Bianchi, T.; Piva, A.: Detection of nonaligned double jpeg compression based on integer periodicity maps, IEEE Trans. Info. Forensics Secur., 7(2) (2012), 842–848.

[6] Conotter, V.; Boato, G.; Farid, H.: Detecting photo manipulation on signs and billboards, in ICIP, IEEE, 2010, 1741–1744.

[7] Lee, M.-J.; Kim, K.-S.; Lee, H.-K.: Digital cinema watermarking for estimating the position of the pirate. IEEE Trans. Multimed., 12(7) (2010), 605–621.

[8] Tagliasacchi, M.; Tubaro, S.: Blind estimation of the QP parameter in H.264/AVC decoded video, in 2010 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), April 2010, 1–4.

[9] Bestagini, P.; Allam, A.; Milani, S.; Tagliasacchi, M.; Tubaro, S.: Video codec identification, in Proc. 37th Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2012), March 25 30, 2012, 2257–2260.

[10] Kang, L.-W.; Hsu, C.-Y.; Chen, H.-W.; Lu, C.-S.; Lin, C.-Y.; Pei, S.-C.: Feature-based sparse representation for image similarity assessment, IEEE Trans. Multimed., 13(5) (2011), 1019–1030.

[11] Stamm, M. C.; Liu, K. J. R.: Anti-forensics for frame deletion/addition in mpeg video, in ICASSP, IEEE, 2011, 1876–1879. Stamm, M.; Liu, K.: Anti-forensics for frame deletion/addition in mpeg video, in 2011 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), May 2011, 1876–1879.

[12] Dias, Z.; Rocha, A.; Goldenstein, S.: First steps toward image phylogeny, in 2010 IEEE Int. Workshop on Information Forensics and Security (WIFS), December 2010, 1–6.

[13] Kang, L.-W.; Hsu, C.-Y.; Chen, H.-W.; Lu, C.-S.; Lin, C.-Y.; Pei, S.-C.: Feature-based sparse representation for image similarity assessment, IEEE Trans. Multimed., 13(5) (2011), 1019–1030.

[14] Valenzise,G.;Nobile,V.; Tagliasacchi,M.; Tubaro,S.: Countering jpeg anti-forensics, in ICIP, B. Macq and P. Schelkens, eds., IEEE, 2011, 1949–1952.

[15] vanHouten,W.;Geradts, Z. J.M.H.; Franke, K.;Veenman, C. J.:Verification of video source camera competition (camcom2010), in ICPR Contests, Lecture Notes in Computer Science, D. Ünay, Z. Çataltepe, and S. Aksoy, eds., vol. 6388. Springer, 2010, 22–28