

A PROPOSED SOLUTION TO THE ISSUE OF
PROPRIETARY DCCTV VIDEO FILE PLAYBACK

by

Brian Timothy Prendergast

B.S., University of Colorado Denver, 2009

A thesis submitted to the
University of Colorado Denver
in partial fulfillment
of the requirements for the degree of
Master of Science
Media Forensics

2012

This thesis for the Master's of Science

degree by

Brian Timothy Prendergast

has been approved

by

Catalin Girgoras

Jeff M. Smith

Sam McGuire

Date: 4/29/2012

Prendergast, Brian Timothy (M.S. Media Forensics)

A Proposed Solution to the Issue of Proprietary DCCTV Video File
Playback

Thesis directed by Associate Professor Catalin Grigoras

ABSTRACT

The use of Digital Closed-Circuit Television (“DCCTV”) is becoming so widespread in our society that most criminal investigations are likely to involve video surveillance as evidence. Maintaining the integrity and quality of video evidence is crucial to its purpose. The greatest challenge with DCCTV video files is that most manufacturers implement their own proprietary video file formats. Finding the resources and support for the playback of these proprietary video file formats can be a time-intensive task and sometimes ultimately futile. Transcoding the native file format to a different file format results in quality degradation, so working with the native file format is of paramount importance to the forensic video analyst to ensure the highest quality image for examination. This thesis explores different DCCTV proprietary video formats and describes the groundwork and initial implementation of a script designed to identify proprietary video file formats based on file header information in order to provide information for the playback of video.

This abstract accurately represents the content of the candidate’s thesis. I recommend its publication.

Approved: Catalin Grigoras

DEDICATION

I would like to dedicate this thesis to my wonderful wife, Julia, and to her Noni. She will be missed but never forgotten.

ACKNOWLEDGEMENT

I would like to express my gratitude to Mike Bush of the Denver Police Department for offering me his time and experience. His cooperation and assistance was invaluable to the creation of this thesis. I would also like to thank Catalin Grigoras for imparting his extensive knowledge to me and also providing me with the guidance to put it to use. I could not have done this without him. Also, thank you to Jeffrey Smith for providing me with the instruction necessary to pursue a career in this field and for his dedication to his students at the NCMF. And finally, I'd like to take this opportunity to thank Sam McGuire for giving me the privilege of learning from while I was an undergraduate student and for his participation on my thesis committee.

TABLE OF CONTENTS

List of Figures.....	viii
<u>Chapter</u>	
1. Introduction.....	1
1.2 An Overview of DCCTV.....	1
2. DCCTV Retrieval Best Practices Concerning Native Files.....	3
3. Video Compression and Codecs	8
4. Compression Standards.....	9
5. MPEG Compression.....	11
6. Image Compression.....	16
6.1 YCbCr and Subsampling.....	17
6.2 Discrete Cosine Transform.....	19
6.3 Quantization.....	21
6.4 Entropy Encoding.....	23
7. Container vs. Codec.....	24
7.1 FourCC.....	24
8. File Headers and File Signatures.....	26
9. Video Container.....	27
10. Issues with DCCTV Evidence.....	30

11. Forensic Video Analyst Resources.....	32
12. DCCTV Video Format Database Research.....	34
13. DCCTV Proprietary Video Playback Proposal.....	46
13.1. Script Implementation.....	50
14. Future Research.....	56
15. Conclusion.....	57
<u>References</u>	59

LIST OF FIGURES

Figure 1 Best Practices for the Retrieval of Video Evidence.....	4
Figure 2 Video Stream Data Hierarchy.....	12
Figure 3 Group of Pictures (“GOP”).....	14
Figure 4 Video Motion Estimation.....	15
Figure 5 Image Compression Process.....	15
Figure 6 Bayer Color Filter Array.....	17
Figure 7 Chroma Subsampling Ratios.....	18
Figure 8 The Discrete Cosine Transform.....	19
Figure 9 Inverse Discrete Cosine Transform.....	20
Figure 10 DCT Basis Patterns.....	20
Figure 11 Quantization Tables.....	22
Figure 12 Entropy Encoding.....	23
Figure 13 AVI File Hex Data.....	25
Figure 14 AVI Header Format.....	26
Figure 15 AVI File Format.....	29
Figure 16 Hex Data for a .cx3 File (1).....	36
Figure 17 Hex Data for a .cx3 File (2).....	37
Figure 18 EYEMAX DVR File Output.....	38

Figure 19 Hex Data for .mu File.....	39
Figure 20 Hex Data for .mp4 File (Not Actual MPEG-4 File).....	40
Figure 21 Hex Data for .264 File.....	41
Figure 22 Hex Data for .AJP File.....	42
Figure 23 Hex Data for .AJP File (Start Section)	43
Figure 24 Hex Data of .AJP File (End Section)	44
Figure 25 Scripting for Intraframe Compression Detection.....	45
Figure 26 Script Procedure Block Diagram.....	48
Figure 27 File Signature for .WMV File.....	50
Figure 28 Hex Data for .WMV File.....	51
Figure 29 Scripting for Identifying .WMV File.....	52
Figure 30 Onscreen Script Output for .WMV File.....	52
Figure 31 Text File Script Output for WMV File.....	53
Figure 32 Scripting for Identifying .264 File.....	54
Figure 33 Onscreen Script Output for .264 File.....	54
Figure 34 Scripting to Identify .da File.....	55
Figure 35 Onscreen Script Output for .da File.....	55

1. Introduction

The emergence and subsequent success of Digital Closed Circuit Television (“DCCTV”) over analog CCTV as a means for security surveillance has provided key evidence for numerous criminal investigations. The effectiveness of video evidence relies on its quality and accessibility. The challenge facing forensic video analysts is that most manufacturers of DCCTV systems implement proprietary video file formats in their digital video recorders (“DVR”). Proprietary video file formats refers to the methodologies used by a DCCTV manufacturer to maintain the exclusivity of the video file format they have developed. Support and resources for the playback of these video formats can be difficult to find or even nonexistent since it relies wholly on the manufacturer of the technology. The acquisition of digital video in its proprietary, or native, file format is important to the perseverance of its quality and integrity, and therefore less desirable methods of acquisition should not be employed when not completely necessary. Most DVRs offer the option of transcoding the video to a more “open” file format for output, but this results in the degradation of video quality. The ability to properly interpret events and identify people, places, and things within a video are critical during analysis, so the highest quality video available is required.

Currently, there is no standardization for proprietary video file formats used with DCCTV systems. This has resulted in a plethora of video file formats being used in security surveillance. This creates many issues with finding the proprietary software required for video playback software as well as finding proprietary codecs. This thesis examines some of the proprietary video formats associated with DCCTV DVR units in an effort to reach a practical solution of providing the information needed for the playback of proprietary video formats. A script capable of accurately identifying proprietary video files based on the file header data is proposed as a way to connect a proprietary video evidence file to the proper video player or codec required for playback. The framework and initial implementation of this script are described in this thesis.

1.2 An Overview of DCCTV

When DCCTV systems first appeared on the market in the mid-1990s, they were much cheaper than analog CCTV systems. They also

didn't require as much maintenance, such as having to change tapes or the upkeep of moving mechanical parts. Digital storage meant that greater amounts of recorded video could be stored for a much cheaper price. Digital video also meant consistent recording quality, the ability to use high-resolution cameras, and the elimination of the multiplexer. However, the use of DCCTV does present some downsides and a new set of problems unique to its format. Although DCCTV DVR units can hold much more video data, it is usually at the cost of image quality. More detail means more digital data. Compression is a necessity of DCCTV since video information consumes so much bandwidth and results in large data files. Video compression is the use of algorithms to reduce data redundancy when encoding the video data, but the tradeoff is between video quality and storage. Image detail and storage amount conversely affect each other. Almost all video compressions are lossy, meaning that some detail in the video will be lost. The option to adjust the level of compression usually results in compression levels being maxed out by the operator in an effort to retain even more recorded video on a hard drive. Slower frame rates are also utilized to achieve higher detail in video images, so often times crucial moments are not documented on video. [1]

The rise of DCCTV sales in recent decades has created a large diversity of manufacturers in the industry. In some ways this diversity has helped to bring about improvements and innovations in the industry, but with it has also resulted in a variety of different proprietary video containers and codec algorithms. The standardization of video compression algorithms was occurring at the same time DCCTV was becoming popular. As a result of this timing, there has been a persistence of the proprietary video formats that were developed by manufacturers in their early DCCTV units. Even since the creation of standards, DCCTV manufacturers have been using proprietary video formats with their software to implement "tamper proof" video files. This is to ensure that surveillance video evidence will maintain its integrity through the investigative process and will be admissible in court as evidence. It is also done to keep customers from integrating their systems with their competitors surveillance systems. These proprietary video containers and codecs require proprietary video players to properly interpret the recorded video file for playback. Finding the correct proprietary codecs and/or playback software can be the most challenging part in an investigation involving DCCTV and, often, the most time consuming. Video surveillance

can be vital evidence or, sometimes, the only evidence in an investigation, so finding resources to playback the video is essential.

Although more DCCTV system manufacturers are moving toward using more “open” video file formats, there are many older DCCTV units still in use. As these older DCCTV manufacturing companies go out of business, it only gets more difficult to find important information regarding their recording technology. Currently, there is no central database that contains all the information necessary to understand the video formats and software needed to playback proprietary DCCTV video files. [1]

2. DCCTV Retrieval Best Practices Concerning Native Files

To date, there are few resources that outline the best practices for the acquisition and retrieval of DCCTV evidence. One of the most recent is an instructional video entitled “FBI: Caught on Camera” that was released in March of 2010 [4]. The video is 20 minutes long and available free of charge in DVD format to members of the law enforcement community, business owners, CCTV vendors, suppliers, contractors, and educators. It can also easily be found online for free viewing. The video does little to elaborate on the issues surrounding proprietary video formats. It focuses primarily on the installation and implementation of an effective CCTV system by describing how cameras function with respect to lighting conditions, focal length, location, etc. During the portion of the video that discusses retrieval of video evidence, there is only a brief mention of native file format and video compression, and only insofar that it mentions that compression results in less detail and that the native file is ideal because it is least compressed. The video also demonstrates how open format videos recovered from a DVR, such as Audio Video Interleaved (“AVI”) files, are much more compressed than the native file format. Although the importance of retrieving the native file is greatly stressed in this informational video, no explanation of how to overcome the issues associated with codecs and proprietary players.

There is also a written guide entitled “Best Practices for the Retrieval of Video Evidence from Digital CCTV Systems.” [2] It was created by the Technical Support Working Group (TSWG), FBI Forensic Audio, Video, and Image Analysis Unit (FAVIAU), with the assistance of numerous International, Federal, State and local Law Enforcement agencies. The

only version of the guide was released in October, 2006. The purpose of the guide is to provide responding Law Enforcement personnel guidance in securing and collecting video data from DCCTV systems in a manner that will maintain the integrity of the evidence at the highest quality for forensic analysis. The publication is also meant to assist in the development of Standard Operating Procedures for an agency or company.

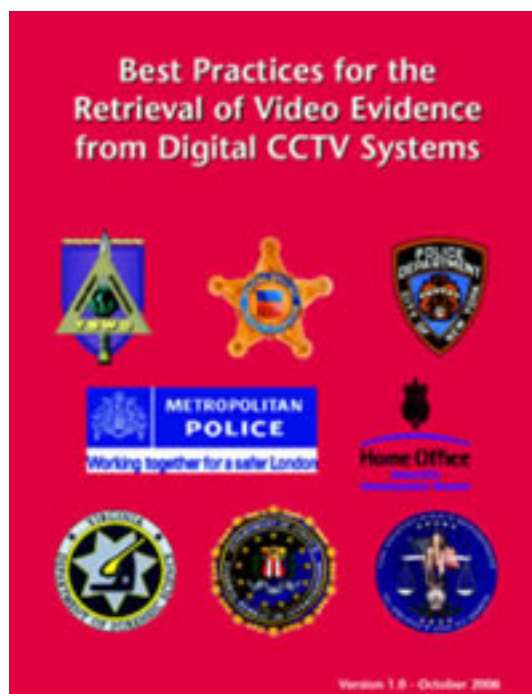


Figure 1 Best Practices for the Retrieval of Video Evidence
Written publication for best practices concerning DCCTV evidence.

The written guide begins by outlining different types of DVRs and a very brief explanation of the video data contained therein. This section notes that all DVRs use compression and that most DVR systems utilize a native or proprietary file format, which would require proprietary playback software or codecs from the manufacturer in order to play back the files or to view any metadata. This metadata could include time, date, camera number, etc., which may prove to be critical evidence. The written guide recommends that the native/proprietary video files be exported in addition to any available open file formats, which should only be used for quick reference. The guide also recommends the retrieval of any native or

proprietary files “whenever possible” to maintain the integrity and image quality of the evidence.

The written guide sets forth a number of different output types that can be encountered including the following:

- Compact Disc Rewritable/Digital Versatile Disc Rewritable
- Compact flash
- USB
- IEEE 1394 Firewire/iLink
- Network port
- RCA
- S-Video
- Composite
- VGA
- DVI
- SCSI port (60 pin and 50 pin)
- Removable hard drive
- Magnetic digital data storage tape (DAT, DLT, DDS, AIT)
- DV cassette drive (e.g., Sony HSR-1P)
- Jaz
- Zip
- Magneto Optical

The next section of the guide describes steps to take upon arrival at the scene. This includes a checklist for aspects such as documentation of the video recorder make, model and serial number, whether the system is networked, system time and date displayed, retrieving the system password, etc. All of this information is important to understanding what will need to be done with the evidence and for properly documenting the data for evidentiary purposes. It can also be the key to tracking down the proprietary video player or codec that is needed for video playback.

The next section describes the different types of output encountered in the field. The options for exporting video data are listed in their respective order of suggestion from advisable to least advisable:

1. CD/DVD Writer
2. Compact Flash Drives

3. USB/Firewire/SCSI Devices
4. Network Connection
5. Replacing Hard Drives
6. Drive Duplication
7. Legacy Output
8. Removal of DVR Unit

All of these methods of export would still result native file formats that would require finding a proper player, unless the DVR has one available for export.

Following this section are last resort options for data retrieval that result in non-native and non-proprietary video files. Although it is rare, some DVR systems only allow for analog output. The guide recommends that analog output be recorded onto digital magnetic tape through the use of video tape recorder. It is noted that a video capture card can be used to digitize the video signal but attention must be paid to maintaining the frame size. When a DVR has both an s-video and composite output, the s-video output is the preferred method. This is because s-video carries the video signal on two channels: one for luminance (intensity, or “Y”) and one for chrominance (color, or “C”), while composite video encodes all this information into one channel. Neither of these types of output carries audio information. When a VGA or DVI output is available, the video signal must be processed using a scan converter, which degrades the resolution of the video. This method is the last resort for analog output. Analog video retrieval of evidence should not be used in lieu of exporting the native DVR file format simply because the proper software player could not be found.

The Federal Bureau of Investigation formed the Scientific Working Group on Imaging Technology (“SWGIT”) in 1997 to establish the best practices for video and image forensics. [3] Of their documents relating to photography, videography, and video and image analysis, the document titled “Section 7” has the most relevance to the issue of proprietary DCCTV video file playback. The document states the following:

“For Digital CCTV (DCCTV), if possible obtain the pertinent video information in the native file format with the appropriate player. The analyst should be aware that different methods of playback and

extraction (including universal players) may yield different results. When viewing digital video using the proprietary software, the player or on-screen display (OSD) may affect the representation of the video. An incorrect display aspect ration will not accurately depict the dimensions of the actual recorded video. For example, objects that should have been recorded as circles may be depicted as oval instead.

In some instances, the original recording hardware, or equipment of the same make and model, may be necessary for playback.” [3]

This demonstrates another hazard of not playing a proprietary video file with the intended playback software – it is difficult to predict how different video players will visually distort the video evidence , which can lead to the misrepresentation of the events caught on camera. One of the primary goals of the video forensic analyst is to avoid introducing any alterations to the evidence in the effort to preserve its integrity for analysis and its admissibility into a court of law.

Despite the helpful information contained within these resources, they fail to provide solutions to the problems associated with native or proprietary file formats and codecs, the same way that “FBI: Caught on Camera” omits that information. Given that video evidence is ideally retrieved in its native or proprietary format, every effort must be made to retrieve the data in that format rather than resorting to less desirable methods of acquisition. Retrieving video in its native format results in the best quality evidence to be used for forensic analysis, which is the highest priority of the video forensic analyst. The reason that native file format retrieval is paramount is because of video compression levels. The first stage compression performed by a DVR unit provides the highest quality image that can be obtained. The conversion of this first stage video compression to another video format only leads to more compression and other possible complications. An understanding of how video compression works helps to demonstrate how this process affects video quality and how this has led to the need to create a set of standards for encoding video.

3. Video Compression and Codecs

A large amount of digital data is required to represent analog video information. Digitized analog video can consume as much as 165 Mbps of bandwidth [1] and it would be expensive to store the large amounts of data. Therefore, the only practical way of storing and transmitting video information is to use compression. This reduces the consumption of storage and bandwidth, which greatly reduces costs. Compression is the reduction of data needed to reconstruct the original information that was captured. To achieve this, compression removes redundant or unnecessary information. Compression algorithms rely on the fact that the information exhibits order and patterning. Because of this, redundant information can be eliminated or reduced by removing extraneous information. For example, a video containing frequency information that a display is not capable of producing can easily be eliminated from the transmitted information.

Codec refers to a device or program capable of encoding and decoding digital data. Codecs encode one or more data streams for transmission, storage, or encryption. There are thousands of different codecs. The same codec used to encode a video must be used to decode the video for viewing. For the purpose of DCCTV, codecs are often proprietary. The manufacturers of the proprietary software are known to create their own algorithms for encoding video or slightly change widely used video compression algorithms. Therefore, in most cases it is necessary to have the specific codec that is only available from the DVR manufacturer to decode the video data, which can be very problematic.

The basis for video compression techniques are an extension of image compression techniques, but the introduction of the temporal domain in video offers more effective ways to achieve higher levels of redundant data reduction. Video compression has the ability to use image prediction between frames, meaning that instead of only looking at redundant data within the one image, it can look at redundant data across the many images that make up the video sequence. The codec defines the how the video will compressed and decompressed.

4. Compression Standards

The need to standardize digital video compression began with the advent of high-definition television (“HDTV”) and the growing use of teleconferencing. A number of standard video compressions (as well as audio compression formats) have been established by both the International Telecommunication Union (“ITU”) and the International Organization for Standardization (“ISO”). Since the ITU is not a formal standardization organization, their documents relating to digital video are released as recommendations rather than standards. ISO works with the International Electrotechnical Commission (“IEC”) to create standards within the information technology world. Their joint works are recognized as the “ISO/IEC.” The two basic compressions standardized by the ISO/IEC are JPEG (named for the Joint Photographic Experts Group) and MPEG (named for the Moving Picture Experts Group). JPEG and MPEG have established the following image and video compression formats that also come recommended by the ITU:

- JPEG
- Motion JPEG
- JPEG 2000
- Motion JPEG 2000
- H.261
- H.263
- MPEG-1
- MPEG-2
- MPEG-4
- H.264

A new group was formed at in the late 1990s known as the Joint Video Team (“JVT”), which is comprised of MPEG and the Video Coding Experts Group (“VCEG”). VCEG is responsible for the line of “H.26x” video coding standards that have embodied many of the innovations used in all modern compression standards. VCEG is currently responsible for many of the image and video compression standards. Their partnership with MPEG began in the mid-1990s with the development of H.262 video coding, which was developed to be the MPEG-2 standard. [1] In May of 2003, VCEG joined with MPEG to create the standardization known as MPEG-4 AVC/H.264. The ISO/IEC standard is technically known as

MPEG-4 Part 10 where “Part 10” refers to the family of Advanced Video Coding (“AVC”) standards. In the ISO/IEC standards, a “Part” refers to a certain aspect of the whole specification. Some Parts of video compression standards are added years after the initial creation of the standard, as is the case with Part 10 of the MPEG-4 standard. This means that it is possible to have two DCCTV units that both compress video that meet the MPEG-4 standard, but that are not exactly the same. An older unit may meet the MPEG-4 Part 2 standard, also known as MPEG-4 Visual and commonly referred to simply as MPEG-4. This video compression standard is distinctly different from the newer MPEG-4 Part 10, which is commonly referred to as AVC or simply H.264. The MPEG-4 AVC and H.264 standards are technically identical and are jointly recognized as MPEG-4 AVC/H.264. The different types of MPEG-4 are still compatible, but it is important to be aware of the differences between seemingly identical video files. MPEG-4 Part 2 is also compatible with H.263 in that it can be decoded by a MPEG-4 Video decoder. The MPEG-4 Part 2 video compression includes popular codecs such as DivX and Xvid. [1][6][7]

Motion JPEG compression, or M-JPEG, treats each frame in the video as an independently compressed image and it is therefore the least complex of the video compression types. It is an example of intraframe compression since it only deals with the spatial domain information within a single frame. With intraframe compression, each frame is treated as an individual digital image to which JPEG compression is applied. As a result, the amount of data reduction is not as efficient as with the video compressions that use interframe compression, where redundant information is reduced from one frame to another within the temporal domain. M-JPEG relies on frame rates no higher than 5fps in order to process the video fast enough, which may result in crucial moments not being captured on video. [5] Another major downside is that compression artifacts do not remain in one place from frame to frame and they appear to “float” around during playback. The only real advantage of M-JPEG is that if one frame of video is corrupt or dropped during transmission, the rest of the video remains unaffected. Motion JPEG 2000, or M-JPEG 2000, is the only other ISO/IEC standard besides M-JPEG that uses intraframe compression. The only major difference between the two is that M-JPEG 2000 takes advantage of improvements made to image compression from the original JPEG algorithm to the new JPEG 2000

algorithm. The video compression types that take advantage of interframe compression include MPEG1, MPEG2, and M-PEG4 AVC/ H.264. [1]

Although many of the old video compression standards established by the ISO/IEC are becoming obsolete, there are still many older DCCTV systems that use them. Fortunately, the superior MPEG-4 and H.264 video compressions are quickly becoming the most standard DCCTV video compressions on the market. For a company to manufacture a product that uses the compression standards established by ISO/IEC, they are required to pay licensing fees. A Denver-based firm that is not affiliated with MPEG called MPEG LA licenses patent pools covering essential patents required for the use of MPEG-2, MPEG-4 Part 2, IEEE 1394, VC-1, ATSC, and MPEG-4 AVC/H.264 standards. It is likely that many manufacturers of DCCTV systems choose to make their own proprietary containers/codecs based on the standardized formats rather than pay to be completely compliant with the standards. This results in video files that are similar to the standard formats, but not 100% compatible.

5. MPEG Compression

There are many different types of video compression, but MPEG compression is the most widely used. It is a type of interframe compression that takes advantage of both the spatial and temporal domain. The compression techniques described in this section serve as an overview for all interframe video compression types since most have been based on this system in one way or another. With interframe video compression, the hierarchy of the video data stream is as follows:

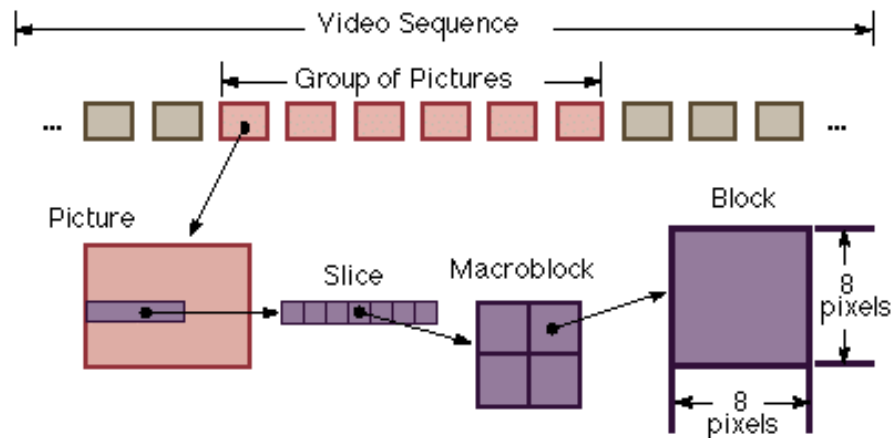


Figure 2 Video Data Stream Hierarchy

- 1) Video Sequence
- 2) Group of Pictures
- 3) Picture
- 4) Slice
- 5) Macroblock
- 6) Block

The video sequence is simply the video clip. These are the recorded events in a video of any length of time. The term refers to the video file clip as a whole. A Group of Pictures (“GOP”) is comprised of specific types of frames, or pictures. A frame can be predicted based on the surrounding frames that precede and/or follow it. These frames take advantage of both the spatial and temporal domain types of compression. Interframe compression is a method of predicting frames based on other key frames. This technique relies on a sequence of three types of frames:

- I-frames: Intraframe
- P-frames: Predicted frames
- B-frames: Bidirectional frames

This type of video compression relies on image-to-image prediction and each of these frames plays a role in the prediction. The I-frame is self-contained and is independent of the other frames and therefore its encoding for spatial compression is similar to that of a JPEG image. In this sense, it is also the only frame that can always be retrieved on its own as a complete image. The P-frame is created based on the preceding I-frame. Since frames are so close together in time and movement is so slight, this technique, known as “motion compensation,” is quite effective.

The difference between the P-frame and the I-frame is known as the prediction error. The P-frame records only the data that is different from the preceding I-frame, generally this is only the movement between the two frames. It may retain about 40% to 50% of the information from the I-frame it is referencing. In between the I- and P-frames are B-frames. B-frames reference both the I- and P-frames that both precede and follow it. This uses the same technique of motion compensation used by the P-frames but it uses the least amount of data since it looks at the differences between the current frame and the frame that precedes and follows it. Since the different frame types in a GOP are created out of sequence in the temporal domain, all of the frames are numbered and can therefore be reassembled in the correct order during the decoding. [5][6][8]

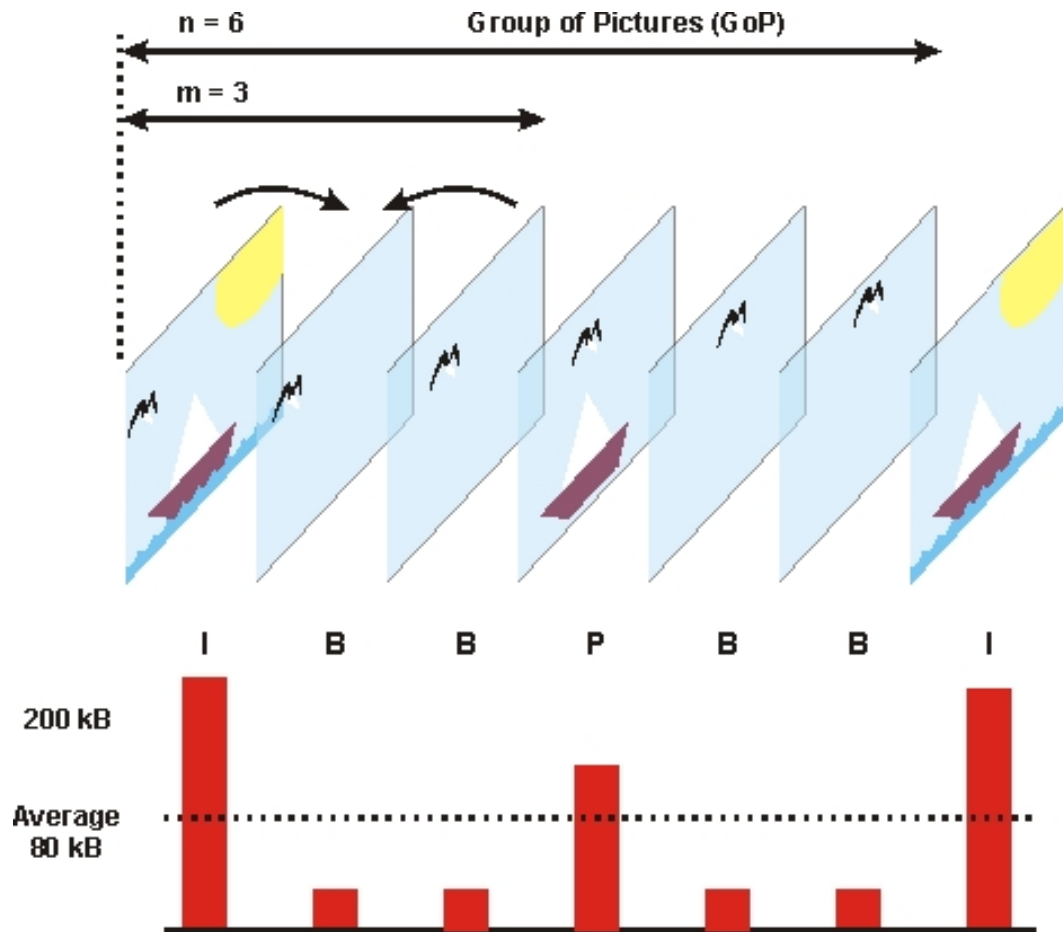


Figure 3 Group of Pictures (“GOP”) [5]

This is a typical pattern configuration seen in interframe compression. The typical amount of data usage is graphed below each frame type.

The figure above demonstrates how a GOP can be described by the depth of compressed predicted frames (“m”) as compared to the total number of frames (“n”). In a typical MPEG GOP, m equals 3 and n equals 12. Some network DCCTV surveillance systems allow the user to control the GOP pattern based on their needs. B-frames contain predictions of how objects have moved between I-frames and P-frames. B-frames look at macroblocks, usually 16 x 16 squares of pixels that are comprised of the 8 x 8 blocks of pixels used by the DCT during frame compression. Macroblocks are described in more detail in the following paragraph. Motion vectors are determined in this way to predict how objects in I-

frames moved to their new position in the P-frame. Determining which blocks of pixels have changed from frame to frame can be the most CPU intensive in the video compression process, but it is the part of the process that allows for the greatest data compression. By determining which macroblocks contain similar groupings of pixels, the computer identifies what is considered to be an object and it maps its motion through the picture. This is known as the motion vector. [5][6]

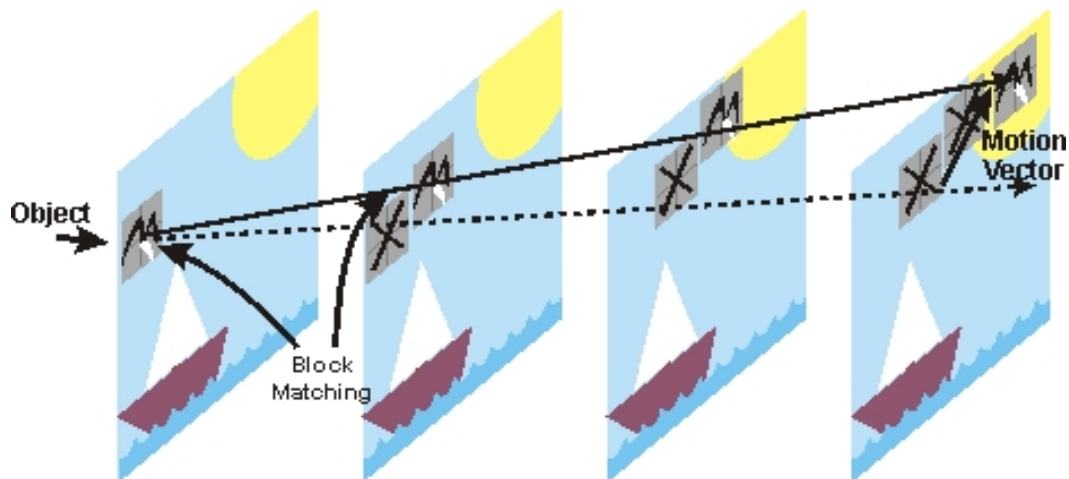


Figure 4 Video Motion Estimation [5]

This system looks for blocks that approximate other blocks in previous pictures to determine motion estimations.

The macroblock is the building block of an MPEG frame. The blocks are arranged into a slice, which is a contiguous sequence of macroblocks that are arranged in scan order from top left to bottom right. The biggest innovation in MPEG 4 AVC/H.264 compression is that the techniques that were carried out on entire frames are performed on the Raster scanning is used on the fields to rebuild the entire image, or frame. A slice is encoded without reference to any other slice, therefore if a slice is lost or corrupt, decoding can commence with the next slice. The header data of the slice describes the address or position of the macroblock within the picture. When the macroblocks are reassembled, they comprise one entire frame. The macroblocks and blocks are handled in the same way as a digital image using JPEG for compression. This process is described in the following section. [5][6]

6. Image Compression

Although many of the basic JPEG digital image compression concepts are relevant to video compression, the introduction of the temporal domain in video creates new challenges as well as new opportunities for data reduction. Most of the techniques used to compress video stem from JPEG image compression. Many of the techniques used in image compression take advantage of how the human eye perceives images. A basic understanding of JPEG compression concepts is essential to understanding how video compression works. The key elements to JPEG compression are the conversion of RGB to the YCbCr color space, Discrete Cosine Transform (“DCT”), quantization, and finally Huffman entropy encoding. [9]

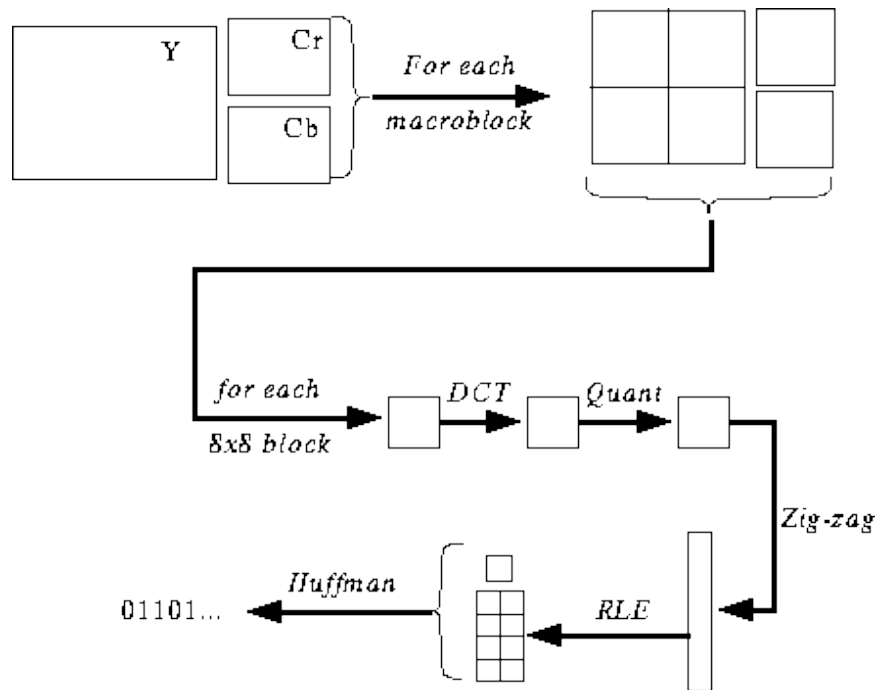


Figure 5 Image Compression Process

- 1) Sampling of each macroblock
- 2) DCT
- 3) Quantization
- 4) Entropy Encoding
- 5) Run-length encoding
- 6) Huffman encoding

6.1 YCbCr and Subsampling

The digital image processing and compression begins with the native spectral red, green, and blue (“RGB”) light that has been separated via the Color Filter Array (“CFA”) on the image sensor inside of the camera.

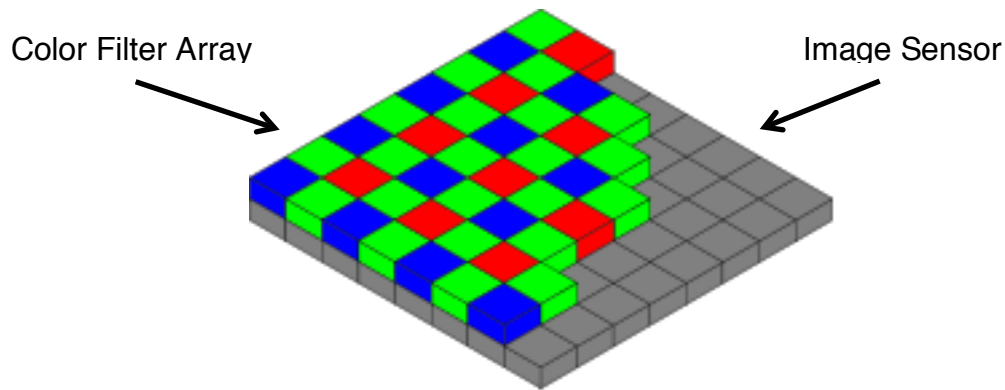


Figure 6 Bayer Color Filter Array

The image sensor can be either a Charge-Coupled Device (“CCD”) or Complementary Metal-Oxide Semiconductor (“CMOS”) that records the analog signals of R, G, and B as voltages. The RGB component signal is converted to luminance (“Y”) and chrominance values of chrominance-blue (“Cb”) and chrominance-red (“Cr”). In analog video and image encoding this concept was referred to as YUV, but in digital video and image encoding it is known as YCbCr, though the two terms are sometimes used interchangeably. Today, YUV is commonly used to describe file formats using the YCbCr encoding. [7]

YCbCr takes advantage of the fact that the human eye is more sensitive to black and white (the luminance component) than color (the chrominance component). There is a lot of redundancy in RGB signals, so eliminating the redundancy during conversion is the first step in the digital process where compression occurs. Since the Y component and the Cb and Cr components are separated, they can be processed differently. The Y component is sampled at the full resolution since the human eye is very sensitive to this component. The Cb and Cr values are subsampled and

compressed to eliminate the high rate of redundancy. This can be done with virtually no visual difference perceived by the viewer. Some digital systems use a luminance range of 0 to 255, where 0 is black and 255 is white. The scale of 0 to 255 is used because this is the range of values that can be stored in one 8-bit byte. If a one byte is used for each value of Y, Cb, and Cr per pixel, there are $256 \times 256 \times 256 = 16.7$ million different colors that can be produced. The human eye can distinguish only about two million different colors, so the 8-bit system proves to be more than sufficient. The widely used ITU-R 601 digital coding standard sets black at value 16 and white at 235, allowing room for error on either side without leading to undesirable clipping. This standard is applied to MPEG-2. Subsampling is then applied to the chrominance values to reduce data. The subsampling scheme is commonly expressed in a three part ratio as J:a:b where “J” is the number of Y samples are taken per line, “a” is the number of Cb and Cr samples that are taken on the first line, and “b” is the number of Cb and Cr samples that are taken on the second line. “J” is almost always 4. [5][7]

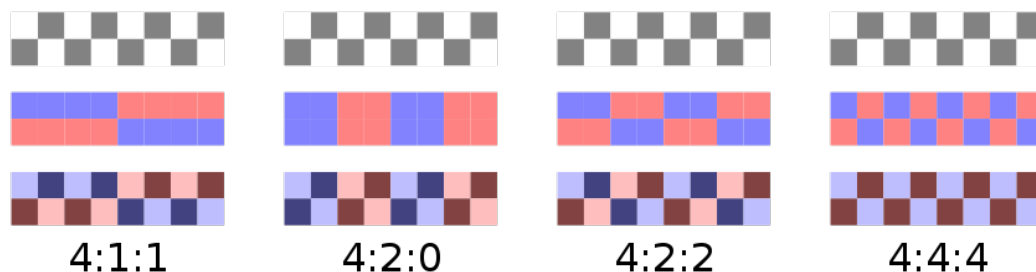


Figure 7 Chroma Subsampling Ratios

These are the most commonly used subsampling schemes. The diagrams are theoretical illustrations to demonstrate the chroma subsampling ratio concept.

In a 4:1:1 sampling, the Cb and Cr signals are sampled at one-quarter of the luminance sampling rate. In a 4:2:0 sampling, the 0 means that Cb and Cr are sampled for every other luminance pixel on one line, and then for none of the luminance pixels on the next line, and so on. In a 4:2:2 sampling, color is sampled at half the rate of the luminance. MPEG-4 Part 2 and H.264/MPEG-4 AVC are capable of a 4:4:4 sampling scheme which results in no subsampling. [5]

6.2 Discrete Cosine Transform

The Fourier Transform is used to transform signals from the space or time domain into the frequency domain. However, the Discrete Cosine Transform (“DCT”) is used in digital image processing since it can perform the same transformation using fewer lines than the Fourier Transform. Although there are several versions of the DCT, the two-dimensional DCT formula is most applicable to image compression.

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \quad k=0, \dots, N-1$$

Figure 8 The Discrete Cosine Transform

The two-dimensional DCT formula is commonly used in JPEG compression.

The DCT is the process of transforming the information, which is in the spatial domain, into the frequency domain. Spatial domain signals are multidimensional, meaning they relate to width and height of an image. Frequency domain values represent the rate of change of the pixels. This is achieved by breaking an image into a square in multiples of four, which is known as a block. In older video compression, this has typically been 8 x 8 pixel blocks. This makes for a total of 64 values per matrix per block. More modern video compressions such as H.264 are fixed at 16 x 16 blocks. The matrix consists of values that are derived from the sampling process. Both the luminance and the chrominance data from the original image samples are broken into their own blocks. The luminance block is always either equal or larger than the chrominance block. For black and white images, only one matrix is necessary to represent the gray scale. The top left corner coefficient in the matrix is the “DC” coefficient. This value represents the average value of the entire block. The rest of the values in the matrix are the “AC” coefficients. The DCT typically transforms the 8-bit block values into 11-bit coefficients. Since the DC coefficient is an unsigned 11-bit integer, this would require different handling than the AC coefficients that can all be represented by an 11-bit signed integer. To resolve this, all the pixel values within the matrix have 128 subtracted from them, centering the values around zero. The DCT is then applied to each matrix to compute the DCT coefficients. The

reconstruction of the original image is interpreted from these coefficient values by using the Inverse Discrete Cosine Transform (“IDCT”).

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k + \frac{1}{2}\right)\right] \quad k=0,\dots,N-1$$

Figure 9 Inverse Discrete Cosine Transform

The inverse of the two-dimensional DCT formula can be applied to DCT coefficients to recover the original values.

The value obtained from the DCT is a measure of how much of a given frequency is present in the entire block. The coefficient values represent the horizontal and vertical basis patterns present in each pixel of the block (Figure 10). The DCT basis patterns correspond with the frequency of pixel change present in each block.

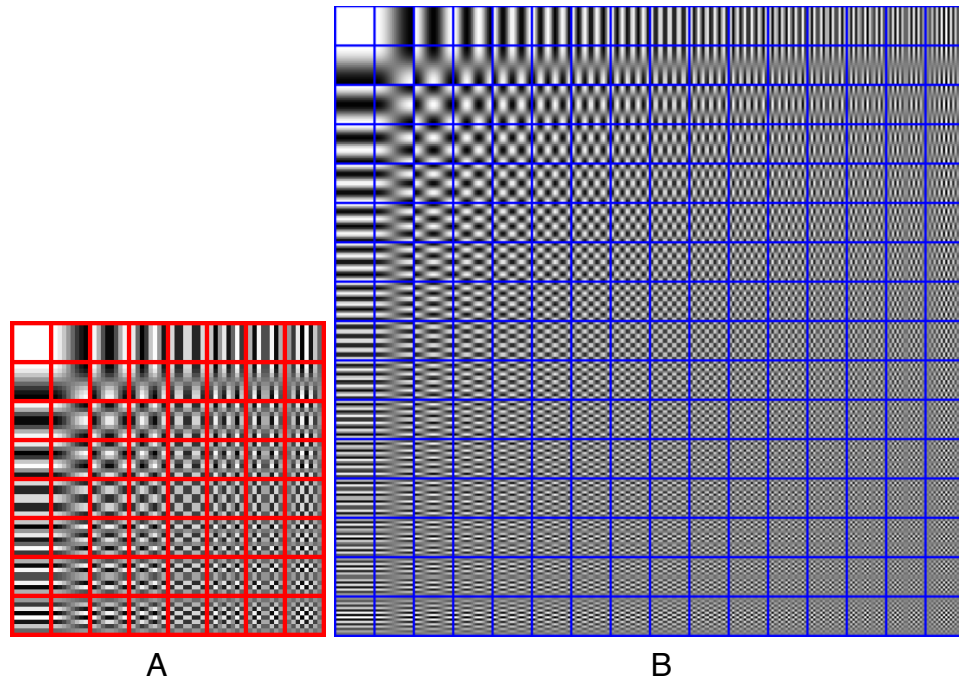


Figure 10 DCT Basis Patterns

The DCT coefficients are measured by the amount of each frequency in an 8 x 8 block (A) and a 16 x 16 blocks (B).

A value is given for how much of each basis pattern is present in each block. The combination of all these weighted patterns will combine to form a similar representation of the original block.

6.3 Quantization

The human eye is more sensitive to low frequency in the visual spectrum. Therefore, reducing high-frequency information is an easy way to reduce data in a video file without detection from the human eye. However, if too much high frequency is eliminated, the image becomes softer in appearance. The human eye is also more sensitive to luminance than it is to chrominance, therefore more focus is placed on retaining luminance information and reducing or consolidating chrominance information. Quantization is a process for reducing the precision of the DCT coefficients, which in turn reduces the bit rate in the compressed data stream. This is the step that achieves the highest degree of compression. Quantization is the process of transforming coefficient values into values between 0-255 so that the data can fit into a byte. A quantization table is used to divide the coefficients for both the luminance and chrominance tables. By looking at the quantization table in Figure? Table 3, it is evident that the lower numbers concentrated in the upper left corner will preserve the lower frequency values after quantization, as seen in Table 4 of Figure 11 on the following page. [5][7]

151	147	152	140	138	125	136	160
157	148	152	137	124	105	108	144
152	151	146	128	99	73	75	116
154	148	145	111	91	68	62	98
156	144	147	93	97	105	61	82
155	139	149	76	101	140	59	74
148	135	147	71	114	158	79	66
135	120	133	92	133	176	103	60

1) Sampled values from block

23	19	24	12	10	-3	8	32
29	20	24	9	-4	-23	-20	16
24	23	18	0	-29	-55	-53	-12
26	20	17	-17	-37	-60	-66	-30
28	16	19	-35	-31	-23	-67	-46
27	11	21	-52	-27	12	-69	-54
20	7	19	-57	-14	30	-49	-62
7	-8	5	-36	5	48	-25	-68

2) Subtract 128 from each value

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

3) Quantization table

-4	14	2	1	-1	-1	1	0
5	-1	3	-5	2	0	-1	0
5	-5	-2	2	0	0	0	0
1	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

4) Quantized coefficients using the DCT

19	31	13	19	18	-9	4	32
25	35	17	8	-3	-29	-21	9
23	32	18	-10	-28	-49	-47	-14
17	23	15	-26	-43	-49	-59	-28
19	17	16	-34	-41	-32	-63	-43
27	13	17	-36	-27	-1	-61	-61
24	0	9	-38	-7	35	-48	-67
3	-15	-1	-41	7	62	-33	-63

5) Dequantized coefficients using the Inverse DCT

147	159	141	147	146	119	132	160
153	163	145	136	125	99	107	137
151	160	146	118	100	79	81	114
145	151	143	102	85	79	69	100
147	145	144	94	87	96	65	85
155	141	145	92	101	127	67	67
152	128	137	90	121	163	80	61
141	113	127	87	135	190	95	65

6) Add 128 to each value

Figure 11 Quantization Tables

These figures demonstrate the process of quantization.

Standardized compression types use specific quantization tables. Encoders such as DivX, Xvid, 3ivx, MPEG-2, and MPEG-4 AVC/H.264 allow the user the option to customize the quantization table. The ability to set the level of compression is known as the scaling factor. The scaling factor value is represented by Q. The value of Q can range from 0 to 100 where 100 is the least amount of quantization and resulting in the best quality image, and of course, the largest file size. [5]

6.4 Entropy Encoding

After quantization has been applied, many of the high frequency coefficients have a value of zero. A technique called “run-length” coding is applied to take advantage of this fact. Consecutive zero value coefficients can be grouped together to be represented by one value, and then the coefficients that do not have a zero values are encoded. Using one value to represent a set of repeated values greatly reduces redundant data. This accomplished using a zigzag patterning across the matrix. This is the most efficient method since it starts with the upper left corner, where the lower frequency values reside. Most of the higher frequency values have a value of zero after quantization and will not have to be transmitted.

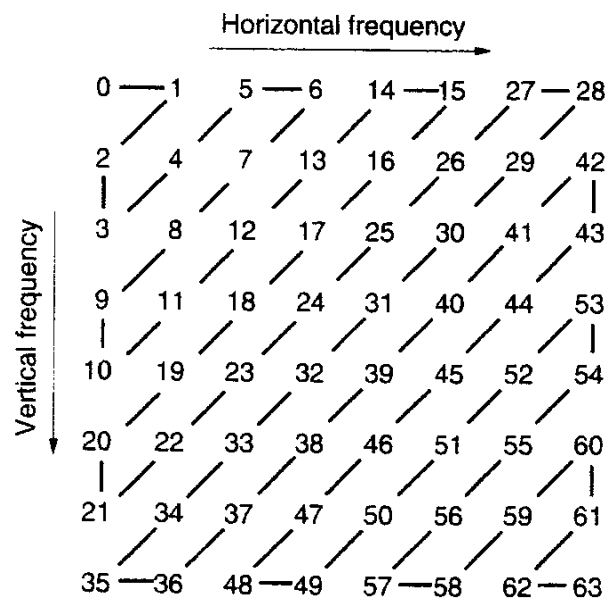


Figure 12 Entropy Encoding

The DC coefficients are predictively coded, using the previous block's quantized DC coefficient as the predictor. Next, Huffman coding is utilized to assign a code word for each combination of run length. A Huffman table is included in the data stream so that the code words can be properly decoded for video playback. [7]

7. Container vs. Codec

The two most rudimentary elements of a digital video file are the container and the codec. Most people get the two confused and do not fully understand the distinction between them. Basically, the video data is encoded by the codec and that data is then stored in a video container file. The encoding stage is where the video gets compressed. A video container could theoretically be used to contain any kind of encoded video, however, certain codecs are typically associated with certain containers. This is traditionally due to licensing and incompatibilities that prevented the use of certain codecs in other containers. Although the container is separate from the codec, it supplies the format that is recognizable by a computer allowing it to identify the codec. The computer needs to identify the codec that was used to encode the video data stream contained in the file since the same codec is needed to decode the data.

7.1 FourCC

Apple developed OSType to identify files by a four-byte sequence. The development of the Interchangeable File Format ("IFF") in the mid-1980s was based on OSType and later adapted into the Resource Interchangeable File Format ("RIFF") in the early-1990s, which is commonly used today for a number of different file formats. The four-byte identifier is now referred to as the Four-Character Code, also known as the FourCC. The four-byte identifier typically has values that translate into four-letter text words for easy identification within the American Standard Code for Information Interchange ("ASCII") character encoding scheme. The FourCC can be used as an identifier for many things. For example, the different chunks in the AVI video container are identified by FourCCs. The first chunk that contains metadata about the video file is identified by the "hdrl" FourCC. A program that is used to play an AVI file will look for this FourCC to locate this information. The AVI file chunk that contains the data for the video itself is identified by the "movi" FourCC. Although a

FourCC can be used to identify a number of different things, the most well-known use is to identify a video codec.

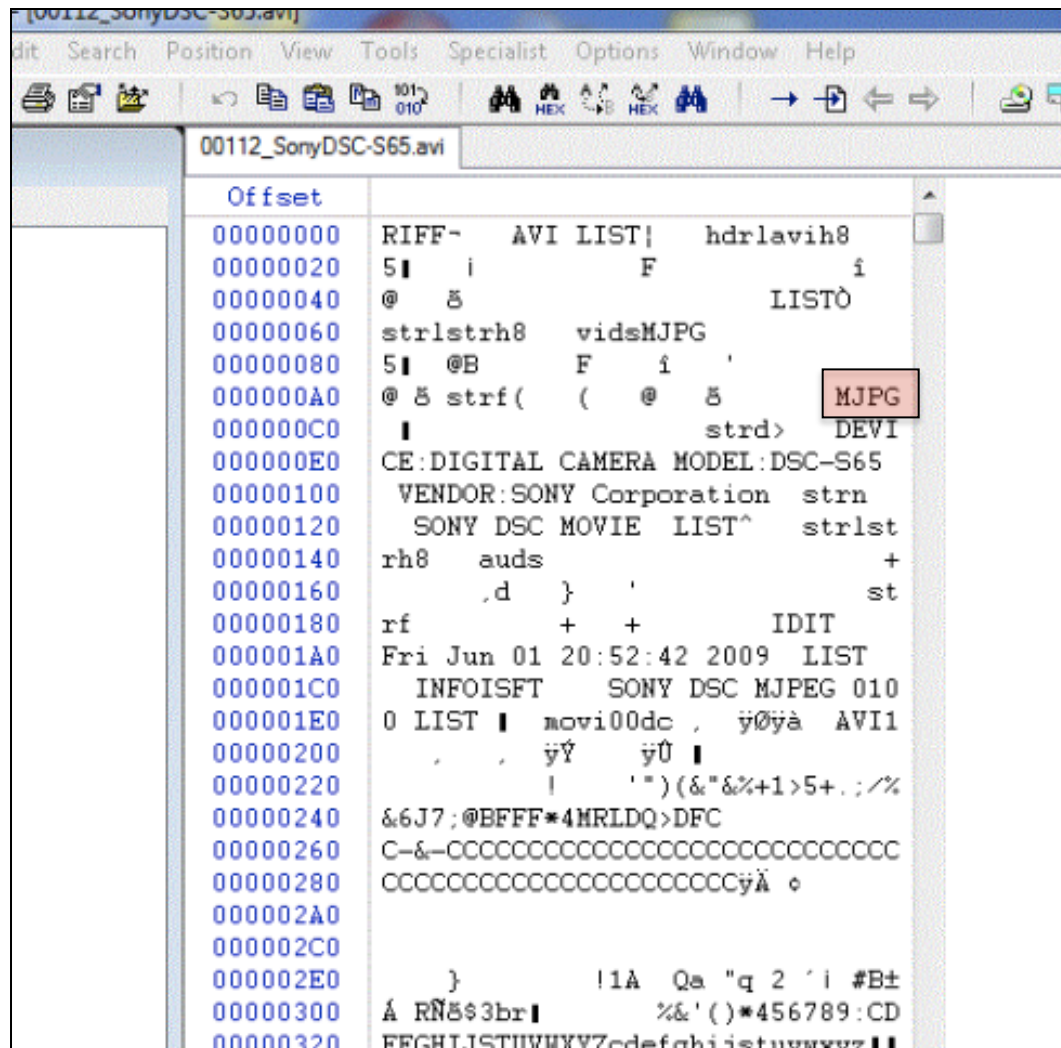


Figure 13 AVI File Hex Data

The FourCC within this RIFF container indicates that this AVI file was encoded using the MJPG codec.

The container structure also allows the computer to find the parts necessary to reassemble the contents of the video data and how to do it.

8. File Headers and File Signatures

In information technology, a file header is a small amount of data used to indicate the type and format of the data that is contained within the file. It tells the computer or program where to start reading the file and how to interpret it. It can usually be found somewhere before the main block of data in the file, typically within the first line of the hex data. It can be thought of as the table of contents for the file since the header information is a way of describing the file contents, as well as the location of the contents. The file header may also contain the date the file was created, the date it was last updated, and the file's size. Subsequently, a file also contains a file footer, indicating to the computer where the file ends. The figure below shows the standard file header established for use with AVI files:

Offset	Size	Description
0	4	time delay between frames in microseconds
4	4	data rate of AVI data
8	4	padding multiple size, typically 2048
12	4	parameter flags
16	4	number of video frames
20	4	number of preview frames
24	4	number of data streams (1 or 2)
28	4	suggested playback buffer size in bytes
32	4	width of video image in pixels
36	4	height of video image in pixels
40	4	time scale, typically 30
44	4	data rate (frame rate = data rate / time scale)
48	4	starting time, typically 0
52	4	size of AVI data chunk in time scale units

Figure 14 AVI Header Format

AVI files contain a 56-byte header, starting at offset 32 within the file.

There are many registered file types that have strictly defined structures, which are then associated with specific file extensions, file headers, and file signatures. The file signature is a component of the file

header that identifies a file type with just a few characters, sometimes formatted as a FourCC. This identifier is also sometimes referred to as the “Magic Number,” referring to the hex data values that make up the ASCII characters of the FourCC. For example, the Magic Numbers used to identify an AVI file are “52 49 46 46” and “41 56 49 20 4C 49 53 54,” which when translated to ASCII characters can be read as the file signatures “RIFF” and “AVILIST,” respectively. There are many file formats, file-format classes, bitstream structures and encodings, and different mechanisms used to compress files or bitstreams. The creators of proprietary video file formats have no intention of registering their file formats for wide use, so the structure of their files is not necessarily available to the public. The identification of file structure and file signatures while analyzing the various proprietary video file formats is the main focus for the research in this thesis.

9. Video Container

A video container is a file that contains some form of video data. It is a metafile format and a member of the multimedia container formats, which is a file format that can store multiple types of data. In the case of a video file, multiple data streams are interleaved and contained in one file. In addition to the video, this can include data for audio and subtitles, as well as the information needed to properly synchronize the data for playback. The container defines the structure of how the data is stored and how the computer identifies the file. For example, AVI containers are structured into “chunks.” One of the chunks contains metadata about the video for details such as the aspect ratio and frame-rate while another chunk contains the encoded video data. Similarly, Quicktime/MP4 files are ordered into “atoms,” while MPEG transport stream utilizes “packets,” and JPEG uses “segments.” Each of these containers has a defined file structure that a program uses to parse out data for playback. The file format ensures that the software will know where to find the data it needs.

Many different video container formats have emerged over time from a number of different organizations and companies as more sophisticated ways of structuring video containers have been developed and as formats have become standardized. Most developers borrow heavily from one another in an effort to create their own format that will

compete with their rivals. Some of the most popular video containers are listed below with their associated file extensions:

- QuickTime (.MOV, .QT)
- Video for Windows (.AVI)
- DivX (.DIVX)
- RealVideo (.RM)
- Windows Media (.WMV, .WMA, .ASF, .ASX)
- MPEG-1 (.MPG, .MPEG, .MPE)
- MPEG-2 (.VOB)
- MPEG-4 (.MP4)
- Ogg (.ogg)
- Matroska (.mkv)
- Flash Video (.flv)

Containers are typically identifiable by their file extensions. People commonly mistake the container for the codec, usually because certain containers are synonymous with a particular codec. A container can technically contain any type of data independently of the codec. Today, perhaps the most common association between codec and container is the H.264 codec and the .MP4 container. The .MP4 file is a standard multimedia container format established by ISO/IEC as Part 14 of the MPEG-4 standard, however an .MP4 file can technically contain video encoded by other codecs. The MPEG-4 container was made to have similar features to that of the Apple Quicktime container. The .MP4 container was built upon MPEG standards combined with Quicktime features. Apple has even dropped the .MOV Quicktime file format in favor of the .MP4 format. The once popular AVI container was designed to compete with Apple's Quicktime file format and is therefore very similar in design. Listed below is a diagram of how the AVI container is structured:

• RIFF 'AVI '	Audio/Video Interleaved file
○ LIST 'hdrl'	Header LIST
▪ 'avih'	Main AVI header
▪ LIST 'strl'	Video stream LIST
• 'strh'	Video stream header
• 'strf'	Video format
▪ LIST 'strl'	Audio stream LIST
• 'strh'	Audio stream header
• 'strf'	Audio format
○ LIST 'movi'	Main data LIST
▪ '01wb'	Audio data
▪ '00dc'	Video frame
▪ ...	
○ 'idx1'	Index

Figure 15 AVI File Format

The FourCC identifiers are shown on the left with a description of the corresponding encoded data content on the right.

The AVI container was later dropped in favor of the Advanced Systems Format (“ASF”) container, which was Microsoft's newer proprietary digital audio/video container format. Windows Media Video (“WMV”) is technically one of several proprietary compression formats created by Microsoft to be used with the ASF container, but the extension of an ASF file using the WMV codec is almost always WMV. In fact, the .ASF extension is completely interchangeable with the .WMV extension. Similarly, compression formats developed by DivX have become synonymous with AVI containers over the years. Historically, this is because DivX was a hacked version of the Microsoft MPEG-4 Version 3 video codec that became commonly used in pirated AVI versions of movies. Later on, the DivX compression format was legitimized as two different compression formats: one that is MPEG-4 Part 2 compliant and a newer one that is M-PEG4 AVC/ H.264 compliant. DivX has since created the DivX Media Format (“DMF”) media container, though the DivX compression format is still commonly found in AVI containers.

Included in the video container is data that indicates the codec that was used to encode the video. Even if a software media player can parse out the video data from the structure of the container, it may not have the

proper codec to decode the video data contained therein. This is why the codec is the essential part for the playback software to interpret a video file.

10. Issues with DCCTV Evidence

A primary reason that DCCTV manufacturers develop proprietary file formats is to maintain commercial control over the DVR format to prevent third parties from copying their files. The motive for creating these unique formats is usually done in the name of developing a ‘tamper-proof’ video for their customers. The result of having so many proprietary video files is that the forensic video analyst frequently has to struggle to find a means of video playback for the growing variety of video formats on the market.

There are many different types of DCCTV video formats currently in use. Videos can come in the form of executable files that are completely contained in one file. These do not require the investigator to search for a means to play the video, but this can be extremely limiting since the analyst has no choice but to use the playback software that is built into the video. This format also requires the analyst to convert the video into some kind of open file format if they intend to use forensic video and image software for analysis during their investigation. Some DCCTV videos are in an open format, but require proprietary codecs that are only available from the manufacturer. Other video formats are made up of multiple files that must all be present in order for the video player to playback the video. These are perhaps the most problematic video types to deal with because it is difficult to understand how the video files work together. Some file structures rely on two or more separate files for the file containing the encoded video data stream to be interpreted properly by the player. These proprietary video types are so unique that it could be a great challenge to backwards engineer the file format. There is little information available for these formats and the video software designed to play the video file is almost completely necessary for their playback.

Many video formats are known to have “.idx” files, or indexing files, which contain text to be superimposed on the video and the cue information needed to synch the information to be displayed. This can contain information such as date and time stamps, camera number

identification, etc. Some proprietary players will not play a video at all if this file is not present, even though this information is not vital to the playback of the video. Playback software like this is hardwired to only play a video file when its associated .idx file is present, despite the fact that the video file is otherwise readily playable. The growing list of file extensions used for proprietary video files is often times of little help in identifying the correct proprietary software required for video playback. One the most popular generic file extension used in DVR units is “.dvr”. This does not mean that there is a particular file format associated with .dvr files. In fact, numerous different manufacturers use this generic extension for completely different video types. There are many other file extensions that are used and reused with video file formats that are completely different from one another.

The majority of DCCTV manufacturers are located in Asia. The manufactures are known to take an operating system (“OS”) that is already used in one DVR model, and repackage it in a different model. It is commonplace to find the exact same OS used by completely different manufacturers. This is because some of the companies that manufacture the DVR units are using the same software vendors for their products. Because of this, it is not uncommon to see the exact same graphical user interface (“GUI”) in multiple DCCTV units from different manufacturers. Often times, the only difference is the logo present on the GUI. Even so, it is common practice for the output video file from this seemingly similar software to be changed slightly as to differentiate it from other DCCTV units using the same software. This can be frustrating since the expected video format output from a familiar GUI may prove to be incompatible with other playback software known to work with videos retrieved from the similar GUI. This is even true of newer versions of the same DCCTV model. Everything may appear to be identical on the surface, but the exported video may have been changed just enough to be incompatible with previous versions of the playback software.

Currently, the only way to find information associated with a particular DCCTV video file is through time-consuming research. This information can come from a number of different resources including printed DVR manuals, manufacturer websites, and online forums. Forensic analysts have even had to resort to calling the manufacturers in foreign countries in an effort to find support for discontinued models. As

manufacturers merge or go out of business, support for some certain devices becomes nonexistent, including any availability of unique proprietary video player software.

11. Forensic Video Analyst Resources

There are a few resources available online to aid the video forensic analyst in finding useful information about proprietary DCCTV video files and the proprietary video playback software and codecs needed to watch them. The content of these websites is generally made up of information contributed by the community. The most popular resources for information about DCCTV proprietary video files are described below.

Media-Geek: The Forensic Multimedia Community
(<http://media-geek.com/>)

This website consists of a section that is available to the general public, and a section that requires membership to view. To qualify for membership, one must be a government employee, whose current duties include forensic analysis of multi-media related evidence, including but not solely limited to sworn Law Enforcement officers. Employees of private sector companies who regularly provide related forensic analysis services may also apply to become members. Membership is also available to vendors of DCCTV hardware and software for their expertise so long as the website is not used to market their products or solicit customers.

Among the advantages to becoming a member is the creation of a profile, private messaging documents and downloads library, private forums, forensic community events calendar, industry “news flashes,” and Wikis, including a DVR codecs Wiki, DVR documentation Wiki, and DVR extensions Wiki. The Wiki pages are perhaps the most useful resources for the identification of proprietary video files and supporting information about them. For some of these files, there are even video player downloads associated with them and documentation. The forums can also be a good source for the exchange of information within the community concerning information about DCCTV systems. The forums are a good place for the video forensic community to exchange knowledge about video file types and manufacturers. This also provides a place where a forensic analyst can address specific questions about video file playback to the community.

FourCC

(<http://www.fourcc.org/>)

The FourCC website was first created as a resource to offer information about FourCCs for raw pixel formats, but 99% of the site's traffic was directed at video codecs. At the time of writing this paper, there are 320 codecs listed on the site. The content of the site is derived from information sent to the site administrators from the general public concerning video codecs. The video codecs are listed by their FourCC and most are accompanied by a short description. Some of the codecs are even available for download from the site.

CCTV Forum: CCTV Surveillance Discussion

(<http://www.cctvforum.com/>)

The CCTV Forum is not specifically designed to deal with the problems associated with proprietary video playback. The site is intended for the use of security professionals, including links to the International Security Conference and Exposition ("ISC") and the Army Surveillance Information System ("ASIS"). Even so, many of the forums do discuss video file formats and their playback. Many of the discussions are about the implementation and installation of security systems. The site also allows for vendors to advertise their CCTV products. The involvement of vendors on the website can also provide a conduit to request information regarding their legacy products.

Sustainability of Digital Formats: Planning for Library of Congress Collections

(<http://www.digitalpreservation.gov/formats/index.shtml>)

An additional resource used for this thesis was the Library of Congress Digital Preservation website. The goal of this website is to ensure the long-term preservation of digital content by maintaining an inventory of file formats, primarily media file formats. The site contains detailed documentations about standardized file formats, including descriptions, a their history, and file signatures.

Usually the primary resource for the forensic video analyst is their own record of video files and players they have collected through experience. A log documenting the different video file types and their proprietary video players can be the only place to find some of the information certain file types. It is also not uncommon for a forensic analyst to keep any video player software that he or she encounters. The majority of the research conducted for this thesis was done at the Denver Police Department Crime Lab in the forensic imaging department. Unfortunately, most forensic video analysts do not have the unique privilege of having such a great resource allowing for the examination of a wealth of different proprietary video file types and players. Therefore, every attempt was made to take advantage of this opportunity for collecting as much information about as many different file formats as possible.

12. DCCTV Video Format Database Research

A sampling of over 50 real case DCCTV surveillance evidence video files was analyzed to guide the scripting of the program. The Denver Police Department provided access to their archived video evidence files to aid in this research. Their 2008 and 2009 archive cases were used for the research since they were among the oldest video surveillance archives available. They were chosen in hopes of providing video files from older DCCTV units and to avoid handling evidence that may be part of an active investigation. The video files were loaded into X-Ways Win-Hex v.16.3 software to examine the file content at a hex data level. Information regarding the origin of the evidence was not always available with the video file, so these file types were researched later for any available information.

To start the process of data collection, files were selected in date order starting with the 2008 archives. As repeat video file types were encountered, it was decided to target file types based on either their obscure nature or their popularity. The reasoning for this was that the obscure video file types could be examined for new kinds of file structures and header identifiers. This provided insight into the extent of variety found among proprietary video files. By examining as many of the popular video files as possible, it was possible to observe the consistencies and inconsistencies found among a particular file type. Examination of these

files demonstrated how proprietary video files from a particular DVR unit change over time as newer models are introduced and newer software are implemented.

File header data was typically found within the first string of the hex data. Some of the header strings were very apparent, and some not at all. Since there is no standardization for header data among proprietary video files, general criteria for what would be considered a video file unique identifier string had to be developed. It was determined that for a string to be considered as the possible file signature, it had to satisfy a combination of the following:

- readable ASCII characters
- location within 1,000 offsets of offset "0"
- consistent among similar file types
- string is isolated (often surrounded by white space)

There were some video files that satisfied all of the criteria, which were considered to be very strong indications of being a reliable identifying header string for a specific file type. The video files that did not appear to satisfy any of these criteria were not added in the script. Some of the video files that did not appear to contain an obvious identifying string had to be compared to other video files from the same DVR unit. It was only by comparison of files with similar file extensions that some type of identifier could be verified. Some of the evidence archives only had one video file available for a particular file type so the suspected identifying strings could not be verified, due to the limitation of availability. The suspected unique identifiers were documented anyway, along with their offset location. This was done so that future research could confirm if these strings could be used to accurately identify the file type.

The script also includes all known codec FourCCs. This includes all registered video codec FourCC codes, and all FourCC codes that have been identified by the video forensics community. This part of the list is easily expanded as new video codec FourCCs are discovered.

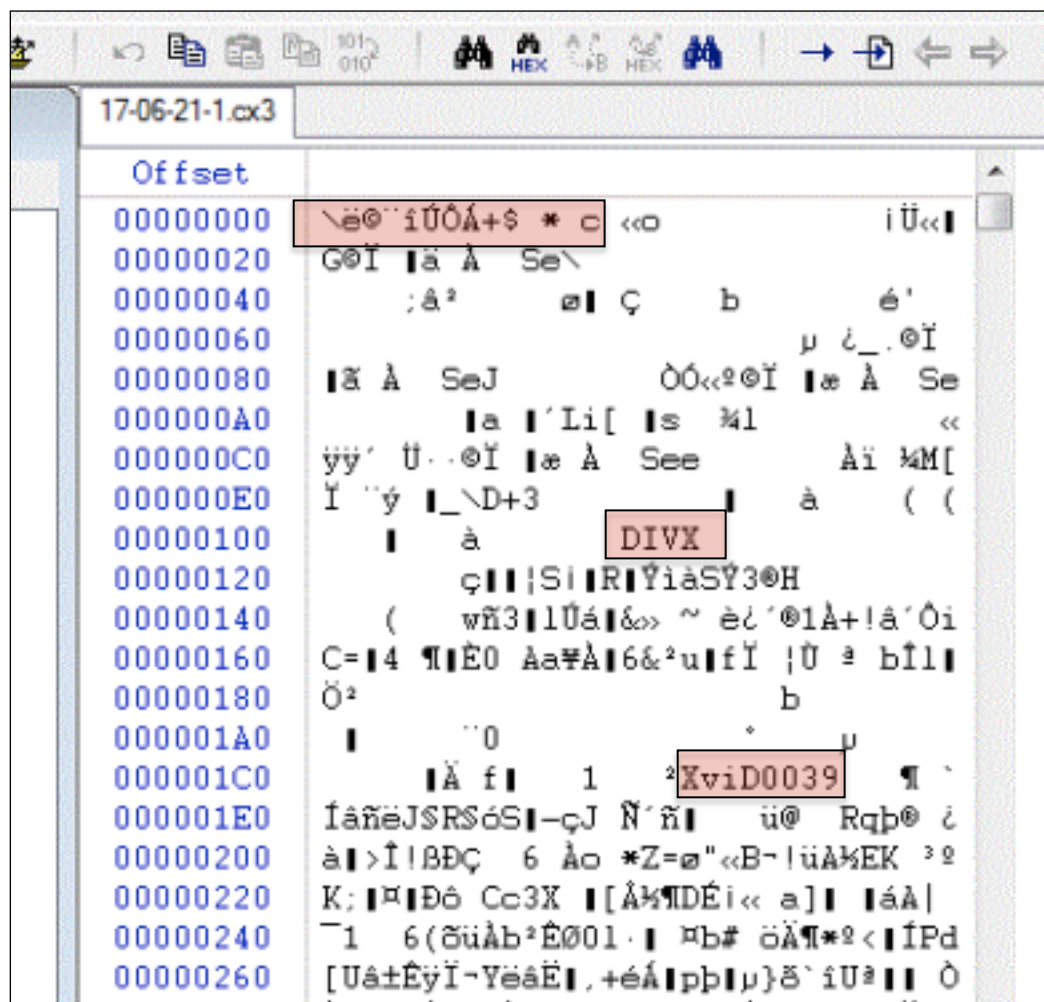


Figure 16 Hex Data for a .cx3 File (1)

The first string in this header could possibly be the identifier for a .cx3 file.

The “DIVX” FourCC indicates the codec and the “XviD0039” below that indicates the DIVX version.

The example in Figure 16 is the hex data for a .cx3 file. No other information was associated with this evidence video file, which is not an uncommon scenario for the forensic video analyst in the real world. By using the unique identifier criteria, it is possible to try to identify the string that makes this video file recognizable as a .cx3 video container. There are no readable ASCII characters besides the familiar “DIVX” FourCC, which is followed by another string, “XviD0039,” that indicates the version

of the DIVX codec that was used. It is possible that the first string can be used as a unique identifier, but it is impossible to tell without verifying this by comparison with another .cx3 file. The hex data from a second case involving a .cx3 file is show below.

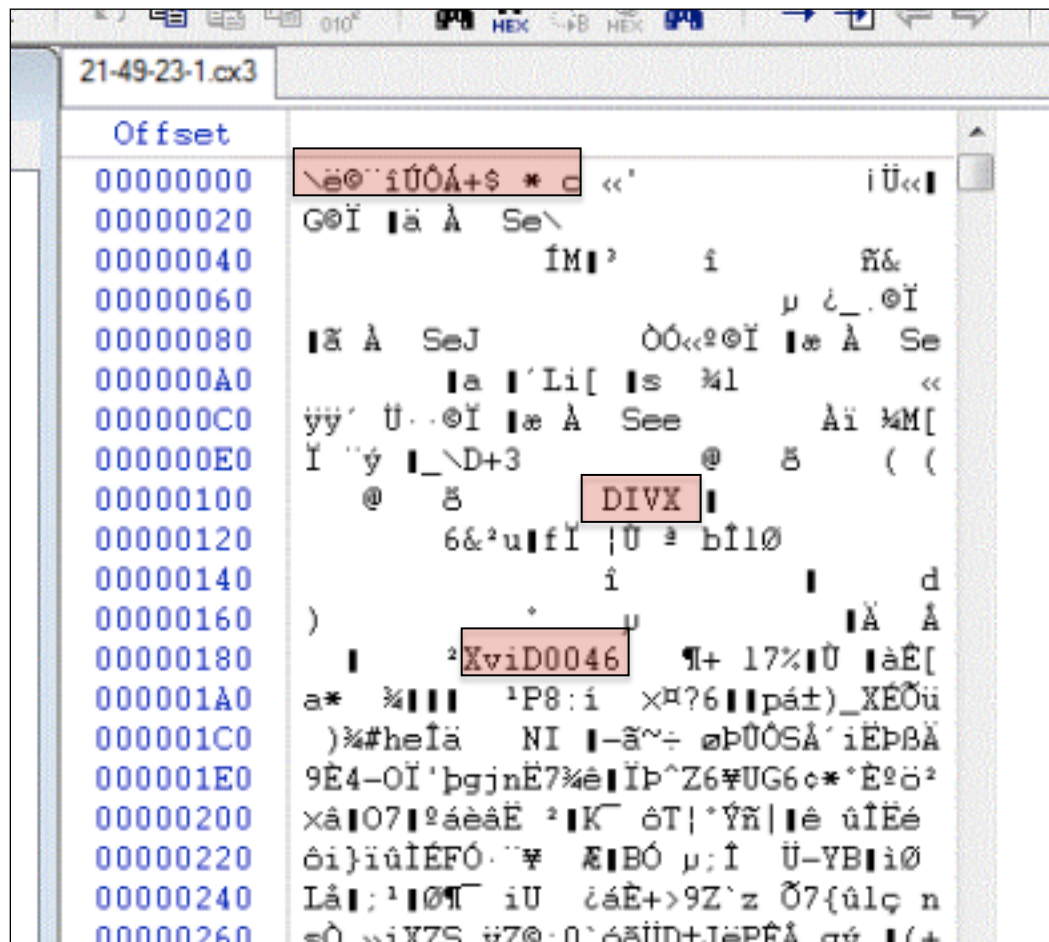


Figure 17 Hex Data for a .cx3 File (2)

This appears to verify the first string as an identifier for a .cx3 file. The “DIVX” FourCC indicates the codec and the “XviD0046” below that indicates the DIVX version.

The hex information from this second .cx3 file appears to verify the string from the first .cx3 file as a reliable identifier for this file type. In fact, the string directly under the first string appears to be consistent between both files as well. Because the first string is so unique, it is not completely

necessary to use another string for identification. The hex information shows that the only major difference between these two versions of .cx3 is their codec. All the information pertaining to this file type and codec would be entered as part of the script with supporting information for both.

Some DCCTV systems output a batch of files that are all needed for native video playback. A popular company called EYEMAX has DVRs that output multiple files for their video player.

File Name	Date/Time	File Type	Size
00000269.00.m4v.gix	11/14/2008 12:03 ...	GIX File	3 KB
00000269.00.ulaw.gix	11/14/2008 12:03 ...	GIX File	4 KB
00000269.01.m4v.gix	11/14/2008 12:03 ...	GIX File	3 KB
00000269.01.ulaw.gix	11/14/2008 12:03 ...	GIX File	4 KB
00000269.02.m4v.gix	11/14/2008 12:03 ...	GIX File	3 KB
00000269.02.ulaw.gix	11/14/2008 12:03 ...	GIX File	4 KB
00000269.03.m4v.gix	11/14/2008 12:03 ...	GIX File	3 KB
00000269.03.ulaw.gix	11/14/2008 12:03 ...	GIX File	4 KB
00000269.04.m4v.gix	11/14/2008 12:03 ...	GIX File	1 KB
00000269.05.m4v.gix	11/14/2008 12:03 ...	GIX File	1 KB
00000269.06.m4v.gix	11/14/2008 12:03 ...	GIX File	2 KB
00000269.07.m4v.gix	11/14/2008 12:03 ...	GIX File	1 KB
00000269.db4	11/14/2008 12:02 ...	DB4 File	16 KB
00000269.mu	11/14/2008 12:03 ...	MuPAD Code	13,146 KB
00000270.00.m4v.gix	11/14/2008 12:03 ...	GIX File	23 KB
00000270.00.ulaw.gix	11/14/2008 12:03 ...	GIX File	33 KB
00000270.01.m4v.gix	11/14/2008 12:04 ...	GIX File	23 KB
00000270.01.ulaw.gix	11/14/2008 12:04 ...	GIX File	33 KB

Figure 18 EYEMAX DVR File Output

The native file export from an EYEMAX DVR is a series of files with the extensions .m4v.gix, .ulaw.gix, db4, and .mu.

The file containing the video data is easily identifiable by its large size. In this case, it is the .mu file.

00000269.mu		00000269.00.m4v.gix	00000269.00.ulaw.gix	00000269.db4
Offset				
00000000		FFmpeg0.4.9-pre1b4718		
00000020				
00000040				
00000060				
00000080				
000000A0				
000000C0				
000000E0				
00000100				
00000120				
00000140				
00000160				

Figure 19 Hex Data for .mu File

The FFmpeg codec is used in this video format.

Although this video contains a familiar video codec, it is still unclear how the other three file types associated with the video file work together. The presence of multiple files associated with a video file is not uncommon. These files associated with this video type could easily be documented in the script to be included with the output for such a video file type.

ch01_20080809004340.mp4		
Offset	Hex	ASCII
00000000	4HKHb³DÖ	> `ö
00000020		-* 7 `ö
00000040		æ "
00000060		xæ ý çl?Á q \$ q A 0 !â
00000080		GÂC1Æ Æ FÍ PâT T-øv SGÂC""
000000A0		v` T T743(4 â 8 Oÿÿ_SS E
000000C0		PÍ8TÄL !1 8Ðø QhP :aH Ô 3I4
000000E0		hb2 oÿÿ Ä ³4Ôó Dx' Nm)²G #4h'"
00000100		a -F2¥ m(ñ 42 Ð¥ : é xÿü E
00000120		úS I 8 @a 4ó!1èÑ G 3 jç0Ôz æ
00000140		ÜhZ ÿÿU"ö 8f <10E a hð 4 4a1
00000160		PÀ <9 r ü; 8f bg X¥ ÄLL3 9æ\$ B%
00000180		ÿÿbç TPT²j Ä tS q Îx'ñs
000001A0		\Ä ÿÿÿSSúÓ "> 8 æ L a K
000001C0		y@pð ÿÿÖ ÿ;Nfá `Xs Lx4 0, C&
000001E0		d :f k«7 ÿbY f#ó1 q± ÇÆ z {
00000200		h'u ÿÿÿ= "æ">1 cFÔ3 æ «7Ä

Figure 20 Hex Data for .mp4 File (Not Actual MPEG-4 File)

This file appears to be in the MPEG-4 container, however the hex data reveals that is not truly an MPEG-4 file.

It is also common to see files that appear to be in more open file formats, but are actually proprietary. This is also used to market DVRs that claim to use a file container or codec such as MPEG-4 and H.264, but that are actually still in proprietary formats. The example above shows a file with the file extension .mp4 used with a MPEG-4 container. It is obvious from looking at the first string in the hex data that this file is not in the true MPEG-4 file container as outlined by the ISO/IEC standard. The manufacturer has simply used the popular extension on for their own proprietary container since it is recognizable. Furthermore, the absence of any FourCC designating the codec makes it impossible to determine how the video is encoded.

Similarly, this file displayed below uses the extension .264 and the company advertises the unit as having H.264 capabilities. This could very well be true, but the software still utilizes a proprietary file format that would require the use of a proprietary video player only available from the manufacturer.

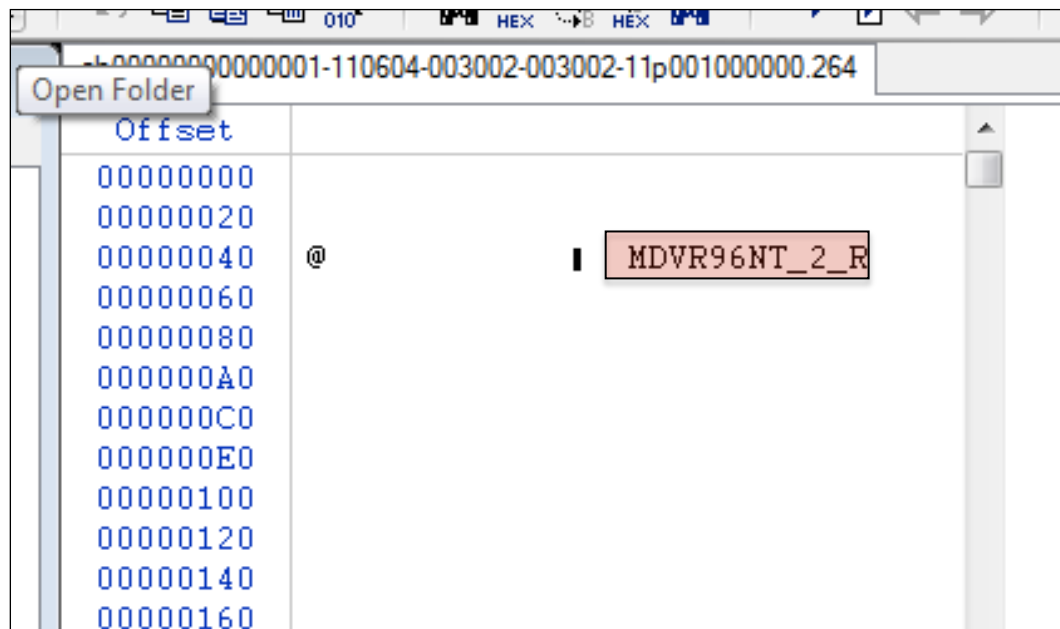


Figure 21 Hex Data for .264 File

This manufacturer claims to use H.264 technology. The video format is still in a proprietary format.

The .264 file extension is in use by more than one manufacturer, but the file headers have been observed to be the same within this research. It was also found that the playback software that comes with the DVR units from each of the manufacturers is also the same. Any video file with this unique identifying string in the header is likely to be playable with the same video playback software. The concept for the script is based on that fact.

Another DCCTV video format that was once very popular is the .AJP file. This file type is a well-known Motion JPEG 2000 video file type, therefore the files did exhibit the same file structure.

DVR-20080622125531.AJP	
Offset	
00000000	i'+ slø > Å U _Ê±P1Adw)aw wdm
00000020	ËÖNÄÄAU 5QcHÜOðÁtBèæ@_ÊX· 2ú÷Ã*
00000040	.'5 òÐTk-³ æ cif4> cèh è Î Ê *
00000060	- Ü ±Huûi -GÄÄèEÛr & ò Î çäs
00000080	/ @who è=& 'S h Ç r, 'ÜŸÄ·Ï:
000000A0	I % Î 'ùgØHòI iäç Òðof;ÊøpéÊ
000000C0	ðHðµÖðæÛ /=~r:S »>".i "[² /¿0
000000E0	ÿ) ä ót? ÑO;\$xDz çx . `iMi
00000100	Ô)û{e <»Np aÊ ÜFùzÇ7Ÿæ' çó(-
00000120	ý¼UZÎ -Ï-ÄÄ x Z ÄZ:. ÊÖ¼m>²* èÛ
00000140	& 38 9PÏ > ù 'ä W~Ÿ&iuB Á) åU
00000160	*u0/ i\ ä> B4èæ²imç' à¼xŸ@òò! C
00000180	p̄ [5mBFÓ % ÷~»v·²u»Ö è Z cièW
000001A0	æ@ Ä U ä (¼irÄ Êpä ÎI4@ ' òÄÏ ð
000001C0	@± 7P ïyá³ eä5}Î· Ÿ xI ösh ±Ä

Figure 22 Hex Data for .AJP File

The header of the .AJP file does not appear to contain any obvious identifying string.

The recognition of a structure for files such as this can be used to identify different software for playback. To deal with the category of videos utilizing intraframe compression such as MJPEG and MJPEG 2000, functions were built into the script to identify the components the unique identifiers to designate the JPEG start, JPEG end, Quantization table, Huffman table, Application Segment, Baseline DCT, and the Start of Scan.

```

if length(string2)==2
    if string2==char('~ÿ');
        disp(['Offset: ',dec2hex(k-1),' -> FFD8
= JPEG Start']);
    elseif string2==char('~ÿ');
        disp(['Offset: ',dec2hex(k-1),' -> FFD9
= JPEG End']);
    else if string2==char('~€')
        disp(['Offset: ',dec2hex(k-1),'
-> FFDB = Quantization Table']);
    end
    if string2==char('~f')
        disp(['Offset: ',dec2hex(k-1),'
-> FFC4 = Huffman Table']);
    end
    if string2==char('~.')
        disp(['Offset: ',dec2hex(k-1),'
-> FFE1 = APP1']);
    end
    if string2==char('~¿')
        disp(['Offset: ',dec2hex(k-1),'
-> FFC0 = Baseline DCT']);
    end
    if string2==char('~<')
        disp(['Offset: ',dec2hex(k-1),'
-> FFDA = Start of Scan (SOS)']);
    end
end
end

```

Figure 25 Scripting for Intraframe Compression Detection

The elements of a video file using intraframe compression can be parsed out by the script.

This can be very useful during analysis since it allows the forensic analyst to extract individual frames, or pictures, from the video. This can be done if the proper playback software cannot be found or if the video file

is found to be corrupt. The frames could be reorganized into a useable format for playback or the individual frames could be examined separately as pictures without relying on the playback of the entire video.

13. DCCTV Proprietary Video Playback Proposal

The proposed solution to the issues associated with identifying the means for the playback of proprietary DCCTV video is the creation of a program that can provide supplementary information about the video file by indicating the proprietary codec used or the proprietary playback software required to view the video based on data stored within the file itself. The requirements for such a program would be the ability to properly recognize a video file by some sort of unique identifier. The identification of a file signature is proposed in this solution. The program would also require a large database of information about all DCCTV units that are available on the market. The database would require constant upkeep in order to stay relevant within the rapidly growing and unregulated DCCTV industry as newer proprietary file formats are introduced.

For this thesis, a script was built in MATLAB software to serve as the preliminary groundwork for such a program. The central function of the script is to examine the video file at the hex level to search for file header strings that serve as unique identifiers for the file type. This method was chosen over using DCCTV video file extensions to identify the file type since the extensions have proven to be an unreliable source for identification. This is mostly due to the fact that the manufacturer of the proprietary file type can use any file extension it chooses, but does not have exclusive use of the extension. That is why it is not uncommon to find two completely different video file types with the same extension. For example, the generic “.dvr” and “.264” extensions are used for many different file formats that all require different video playback software. Because of this, it is only possible to suggest a number of different proprietary video players that are known to work with that particular extension, but not possible to say specifically which player will playback the video.

It became evident throughout the research process that some file formats do not appear to contain obvious unique identifier strings embedded within the file, therefore, a component was added to the script

to identify the file type by extension as a last resort. The output for the positive identification of a file type by its extension is the suggestion of a “possible player,” since the extension is not a completely accurate way to identify a file type, and therefore the proper video player could not be identified with absolute certainty. The list for the “possible player” could also include many video players that do not work, which would require some trial and error on the operator’s part. This is also not a robust means of finding the correct player for playback since a file format and its playback software may have numerous versions that are not backwards compatible. A video file may appear to be a certain file format based on its extension, but not compatible with certain versions of the playback software developed by the manufacturer. This is why using the file signature that the program uses to identify the file is the primary goal of the script. Computer programs typically look for certain file header information to quickly identify the file type. This information is typically stored in the very beginning of the file data, however some file formats are known to store this information at the end of the file or even embedded in some other specific location within the file.

In light of the research, it was decided that the script would look at three different components of the video file in an attempt to identify the correct playback software needed to view the video. The primary search is to identify the file type by its file signature. The secondary searches are to identify the codec by FourCC, if present, and also to examine the extension of the video file. The process of the script is listed on the following page:

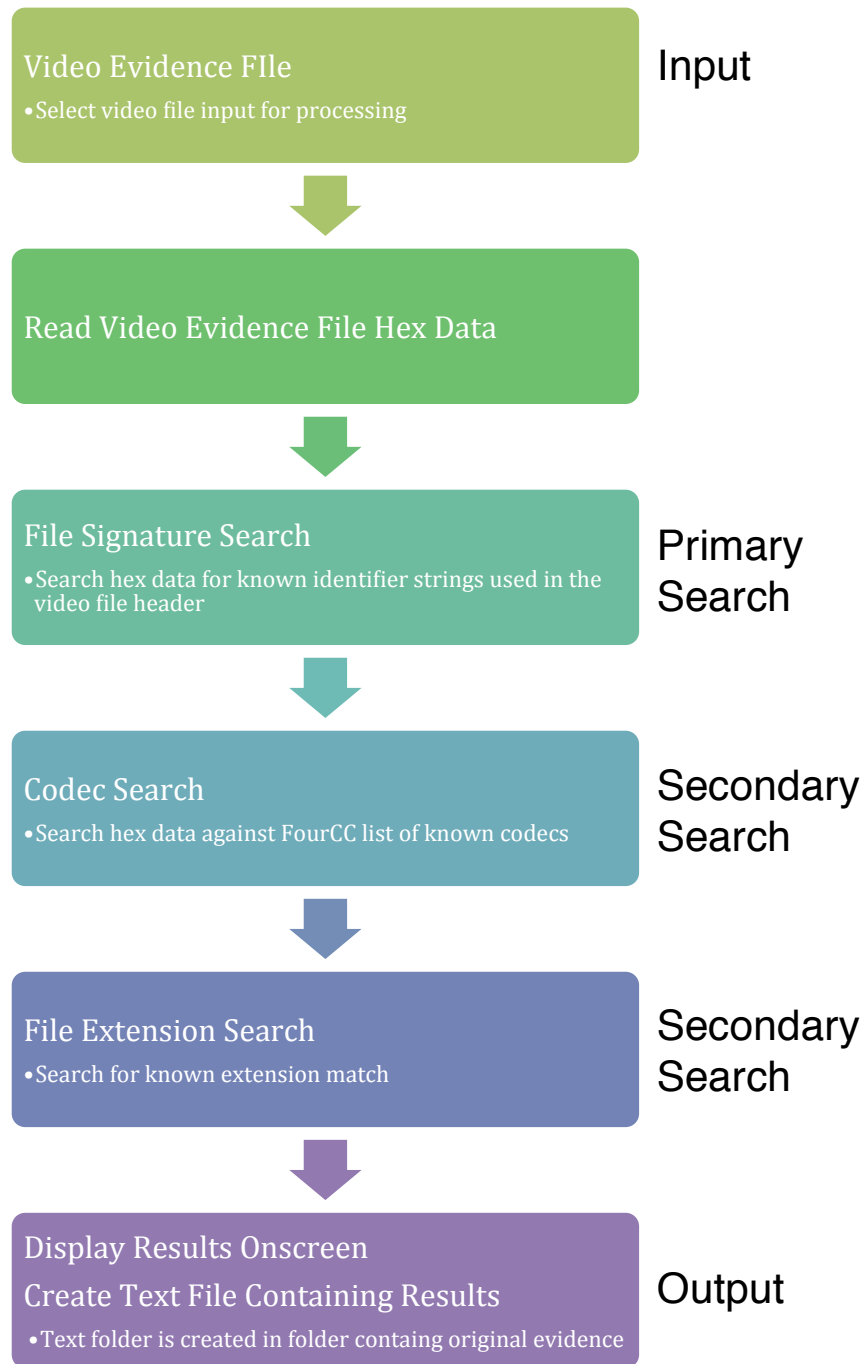


Figure 26 Script Procedure Block Diagram

This block diagram demonstrates the process of the script. The primary and secondary means for video file identification are labeled on the right.

When the script is run, the user is requested to select a video file by its path. Once loaded into the software, the hex data is scanned for specific strings that have been programmed into the script that are unique identifiers for certain file types. The strings are also searched at their specific offset locations. If any of the unique identifiers are discovered in the hex data, the output is information relating to that video format that is displayed onscreen. The next process is to search the hex data for known codec FourCCs that are listed in a text file. The FourCC codes were listed in a text file so that new ones could easily be added. The script displays any FourCCs that are identified and the offset location. Finally, the script looks at the extension of the video file to determine possible video players known to work with files using that extension, which are then displayed onscreen. All the supporting information that is displayed on screen is also written to a text file that is created in the folder containing the video file. The information primarily includes manufacturer information, the manufacturer website link, and suggested video playback software, as well as any other relevant information. By including an arsenal of video playback programs and video codecs with the script, the operator needs only to navigate to the proper files required to play the video evidence.

The original concept was to have a running list of unique identifying strings located in a text file. The intent was for the list to provide a user-friendly way to copy and paste identifying strings into an expandable list. As a file signature string was determined by the user to be consistent and reliable as a unique identifier for a file type, it could be added to the text file. The text file would then serve as the search list for the script against the video file hex data. However, many of the strings found within the headers contained characters below 32 in the ASCII character set, which are nonprinting control characters such as tab, backspace, and end-of-line. The characters above 128 make up the extended ASCII characters. Due to difficulties with processing these types of characters, a different approach would be needed to handle the header strings that contain these characters. Also, some header identifier strings were found to be as short as two characters, resulting in many false positive hits. However, it was observed that the file signatures were located at the same offset in the header of the file. Another issue became apparent when the files that require different playback software that appeared similar were observed to have slightly different identifiers, often times in slightly different offset

locations. For these reasons, it was decided that specific code must be built into the script for each unique identifier, taking into account the offset location. The result is a much more robust method for identifying exact strings in their specific hex offset locations. The outcome has been a higher accuracy of positive identifications.

The scripting used to identify video codecs utilized the expandable list concept that was initially intended to be used for identification of file signature strings. This logic was much more suited to the identification of codecs because they are always formatted as four-character codes, by definition, and using human-readable ASCII characters. This avoids the issues associated with non-printable ASCII characters and extended ASCII characters within the script.

13.1. Script Implementation

The framework for the script was built upon observations made concerning different file formats. Many of the identifying header strings were found in specific locations using non-alphabetical and non-numeric characters. This led to the development of a very robust technique for string identification by focusing on the specific location of the string. By doing this, any combination of characters for the string could be used as the identifiers. This reduces the chances of the script making a false match.

For example, the Windows Media Video (“WMV”) file signature is “0&²u.ř.Û.ª.bÎ”. The file signature is standard and always found in true .WMV files. The file signature always begins at hex offset “0”. The table below shows the characters of the file signature with their corresponding hex data and offset locations:

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ASCII character	0	&	²	u	.	f	ř	.	Û	.	ª	.	b	Î	.	
Hex Value	30	26	B2	75	8E	66	CF	11	A6	D9	00	AA	00	62	CE	6C

Figure 27 File Signature for .WMV File

The offset, ASCII character, and hex value assigned to the .WMV file signature string.

00000000	30 26 B2 75 8E 66 CF 11-A6 D9 00 AA 00 62 CE 6C	0s²u·fī· Ü·²·bîl
00000010	7C 01 01 00 00 00 00 00-07 00 00 00 01 02 A1 DC	;Ü
00000020	AB 8C 47 A9 CF 11 8E E4-00 C0 0C 20 53 65 68 00	«·Geİ··ä·À· Seh·
00000030	00 00 00 00 00 00 00 18 88-6B FE 32 D7 9B 4E A2 3B	kp2×·Ne;
00000040	8B EA BF 9C 37 F2 EE C4-DB 01 00 00 00 00 10 87	·ê·7ôîÄÜ.....
00000050	CA 84 00 FF CC 01 31 0F-00 00 00 00 00 00 20 71	Ê·ÿİ·1..... q
00000060	53 C5 00 00 00 00 00 00 76-E9 C2 00 00 00 00 88 13	SÄ.....véÄ.....
00000070	00 00 00 00 00 00 00 02 00-00 00 40 1F 00 00 40 1F	@.....@·
00000080	00 00 52 8E 57 00 B5 03-BF 5F 2E A9 CF 11 8E E3	·R·W·p·¿·.eİ··ä
00000090	00 C0 0C 20 53 65 14 12-00 00 00 00 00 00 11 D2	·Ä· Se.....Ò
000000a0	D3 AB BA A9 CF 11 8E E6-00 C0 0C 20 53 65 06 00	Ó«°eİ··æ·Ä· Se·
000000b0	E6 11 00 00 5D 8B F1 26-84 45 EC 47 9F 5F 0E 65	æ···]·ñ·EiG·_·e
000000c0	1F 04 52 C9 1A 00 00 00-00 00 00 00 02 01 EA CB	·RÉ.....êÈ
000000d0	F8 C5 AF 5B 77 48 84 67-AA 8C 44 FA 4C CA E0 00	øÄ[wH·g²·DúLÈä·
000000e0	00 00 00 00 00 00 00 04 00-00 00 01 00 0C 00 02 00	
000000f0	02 00 00 00 49 00 73 00-56 00 42 00 52 00 00 00	···I·s·V·B·R··
00001000	00 00 00 00 01 00 34 00-00 00 06 00 00 00 44 00	·····4·····D·
00001100	65 00 76 00 69 00 63 00-65 00 43 00 6F 00 6E 00	e·v·i·c·e·C·o·n·
00001200	66 00 6F 00 72 00 6D 00-61 00 6E 00 63 00 65 00	f·o·r·m·a·n·c·e·
00001300	54 00 65 00 6D 00 70 00-6C 00 61 00 74 00 65 00	T·e·m·p·l·a·t·e·
00001400	00 00 4C 00 33 00 00 00-00 00 02 00 0C 00 02 00	·L·3·····
00001500	02 00 00 00 49 00 73 00-56 00 42 00 52 00 00 00	···I·s·V·B·R··
00001600	00 00 00 00 02 00 34 00-00 00 0C 00 00 00 44 00	·····4·····D·
00001700	65 00 76 00 69 00 63 00-65 00 43 00 6F 00 6E 00	e·v·i·c·e·C·o·n·

Figure 28 Hex Data for .WMV File

The file signature used for all .WMV files is displayed above.

To create the most effective means of identifying this string, particular characters and their specific offset locations were added to the scripting as the identifiers for that file type. In the case of this file format, the “0”, “u”, and “f” were used to represent the entire string at their specific offset locations.

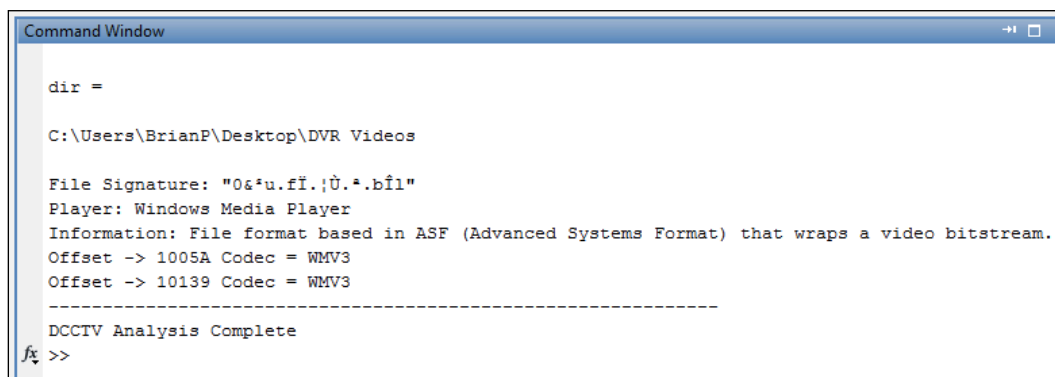
```

if h(1)==char('0') && h(4)==char('u') && h(6)==char('f')
    disp(['File Signature: "0&u.fœ.Ÿ.™.bEl"']);
    disp(['Player: Windows Media Player']);
    disp(['Information: File format based in ASF (Advanced
Systems Format) that wraps a video bitstream.']);
    fid=fopen(name12,'at');
    fprintf(fid,(['File Signature: "0&u.fœ.Ÿ.™.bEl"']));
    fprintf(fid,'\n');
    fprintf(fid,(['Player: Windows Media Player']));
    fprintf(fid,'\n');
    fprintf(fid,(['Information: File format based in ASF
(Advanced Systems Format) that wraps a video bitstream.']));
    fprintf(fid,'\n');
end

```

Figure 29 Scripting for Identifying .WMV File

This portion of the script is an example of how unique characters are handled in the identifying header string.



```

Command Window

dir =

C:\Users\BrianP\Desktop\DVR Videos

File Signature: "0&*u.fİ.Û.*.bİl"
Player: Windows Media Player
Information: File format based in ASF (Advanced Systems Format) that wraps a video bitstream.
Offset -> 1005A Codec = WMV3
Offset -> 10139 Codec = WMV3
-----
DCCTV Analysis Complete
fx >>

```

Figure 30 Onscreen Script Output for .WMV File

This is the information that is displayed on screen for the selected .WMV file.

The examples above show how the script can be used to identify a WMV file based on its file signature. The supporting information can easily be expanded to anything else that could aid the forensic analyst to better understand the file format. The proprietary player is listed as the Windows Media Player and the codec as been identified by the FourCC "WMV3." This information is also listed in a text file that is created in the location of the video file.

DCCTV Report

-----Evidence-----

Date & Time: 22-Apr-2012, 19:32:39

Evidence file: My Movie.wmv

File Signature: "0&²u.ř.!.Û.ª.bÎ"

Player: Windows Media Player

Information: File format based in ASF (Advanced Systems Format) that wraps a video bitstream.

Offset -> 1005A Codec = WMV3

Offset -> 10139 Codec = WMV3

Figure 31 Text File Script Output for WMV File

This text file is created in the video file location and it contains the information about the WMV that is displayed onscreen.

The following example shows a file type with the extension .264 that was retrieved from a popular commercially available video surveillance DVR.

```

if h(81:84)==char('MDVR')
    disp(['File Signature: "MDVR96NT_2_R"']);
    disp(['Video Player: Playback v.2.3.0.5']);
    disp(['Used With: Night Owl Security Products: ']);
    disp(['http://www.nightowlsp.com/']);
    disp(['Used With: SVAT Electronics: ']);
    disp(['http://www.svat.com/']);
    fid=fopen(name12,'at');
    fprintf(fid,('File Signature: "MDVR96NT_2_R"'));
    fprintf(fid,'\n');
    fprintf(fid,('Possible Player: Playback v.2.3.0.5'));
    fprintf(fid,'\n');
    fprintf(fid,('Used With: Night Owl Security Products - '));
    fprintf(fid,('http://www.nightowlsp.com/'));
    fprintf(fid,'\n');
    fprintf(fid,('Used With: SVAT Electronics - '));
    fprintf(fid,('http://www.svat.com/'));
end

```

Figure 32 Scripting for Identifying .264 File

This scripting identifies this particular video file with the .264 extension.

```

dir =

C:\Users\BrianP\Desktop\DVR Videos\NightOwl\NightOwlCASE2

File Signature: "MDVR96NT_2_R"
Video Player: Playback v.2.3.0.5
Used With: Night Owl Security Products: http://www.nightowlsp.com/
Used With: SVAT Electronics: http://www.svat.com/
Offset -> 102FC Codec = H264
-----
DCCTV Analysis Complete
>>

```

Figure 33 Onscreen Script Output for .264 File

The script shows that this particular video file type is known to be used by two different manufacturers.

The script is used to identify the proprietary video player that can playback this kind of proprietary video file that uses the .264 file extension. The “H264” FourCC has also been identified as the codec that was used

to encode the video. As shown in Figure ?, this type of video file is known to be used in both Night Owl, LLC and SVAT Electronics video surveillance DVRs. The video player that can be used to play this video is the software called “Playback v.2.3.0.5.” A text file was also created in the video file folder that contains this same information.

The next example of scripting shows how the script can use the extension to help identify the proper video player.

```
if length(ext)==3
    if ext==char('.da')
        disp(['Extension: ',ext]);
        disp(['Possible Player: F4Viewer.exe']);
        fid=fopen(name12,'at');
        fprintf(fid,'\n');
        fprintf(fid,['Extension: ',ext]);
        fprintf(fid,'\n');
        fprintf(fid,['Possible Player: F4Viewer.exe']);
        fprintf(fid,'\n');
    end
end
```

Figure 34 Scripting to Identify .da File

This scripting demonstrates how the file extension can be used to suggest a file player as a last resort.

```
dir =

C:\Users\BrianP\Desktop\DVR Videos\Astak

Extension: .da
Possible Player: F4Viewer.exe
-----
DCCTV Analysis Complete
fx >>
```

Figure 35 Onscreen Script Output for .da File

The script identifies a “possible” video player based on the file extension.

The script is designed to recommend a “possible” video player or multiple video players since the extension has proven itself to be unreliable as a means for determining exactly what video player can be used with the video file. This is considered the last resort for directing a forensic analyst towards finding the correct playback software.

14. Future Research

The highest priority for improving the script is to devise a better way of better identifying possible file signatures so that the database can be expanded. Upon realizing the difficulties in the task of properly identifying file signatures, it became apparent that the methods initially used to research the hex were not sufficient for all file types. Although many file formats had readily identifiable file signatures within their headers, others did not appear to have any kind of traditional header information at all. Furthermore, it is possible that these types of files do not contain any kind of identifier within the beginning for the file, but rather in some other fixed location in the body of the hex data. For this reason, it would be beneficial to create a script that automatically compares alike video file hex data for similar strings. This would eliminate human error and allow for faster identification of similar file strings. It is also more practical than manually scanning large amounts of hex data for similarities. The main function of this script would be to find identical strings in identical offset locations.

The concept to use the ASCII character file signatures to identify the file types was initially considered as the basis for determining unique identifiers because it was believed that many of the files would have human-readable strings. It was soon discovered that this is not always the case. The possibility that some identifier strings could be human-readable in a different character-encoding scheme was also considered. Considering that some of the non-printing ASCII characters also presented issues in processing, the script may be changed to rely on the raw hex data strings as a means of identification, rather than their translated ASCII characters. Using the “magic numbers” for identification would eliminate any issues related to decoding the hex information to characters.

Once a better way for comparing video files for their file signatures is created, many more files have to be examined. The reliability of the program depends on the verification of the file signatures built into the

script that are determined by research and file comparison. The only way to verify the suspected file signatures is by examining as many different file types or a particular extension as possible. Once a file signature is identified by automated file comparison, the file signature must be proven to be consistent among a certain file type. Along with this, the performance of the script has to be tested for accuracy. After the scripting for more specific file types is incorporated into the script, the script must be tested on as many files as possible to demonstrate its usefulness.

15. Conclusion

The research conducted for this thesis has demonstrated how the issues relating to the playback of proprietary DCCTV video files can be remedied in part by the implementation of a computer program that provides a centralized database for information concerning video format types as well as supporting information for their playback. It is hoped that the script can be developed to the point where many of the older formats in use are incorporated into the script and that newer proprietary formats can be examined for file signatures and easily added into the script. The initial research done for this thesis has exposed many of the underlying challenges associated with proprietary video file identification. As these challenges were encountered, new solutions were also developed to overcome them.

Most DCCTV manufacturers were found to be somewhat consistent in using their own proprietary formats in most of their DVRs, though some did exhibit slight changes between older versions of their software and more current versions. Since so many diverse file structures were discovered, the script was approached by creating specific scripting language for individual file types in order to ensure accurate identification of the file formats. This method has proven itself very effective in the preliminary tests. However, it is evident that an automated means of identifying these file signatures would be much more beneficial and practical.

The new approaches that have been derived as a result of this research could vastly improve the original concept for the proposed script that was outlined in this thesis. Based on the accomplishments achieved through the work for this thesis, the pursuit of developing a program that

can accurately identify the majority of proprietary DCCTV video file types to produce relevant supporting information, including the identification of the correct video playback software and codecs, will be continued.

References

- [1] Nilsson, F., & Axis Communications. (2009). *Intelligent network video: Understanding modern video surveillance systems*. Boca Raton: CRC Press.
- [2] Best Practices for the Retrieval of Video Evidence from Digital CCTV Systems. October 2006.
- [3] Scientific Working Group on Imaging Technology (2009). Section 7: Best Practices for Forensic Video Analysis v.1.0. Retrieved January 5, 2012 from the International Association for Identification website: <http://www.theiai.org/guidelines/swgit/index.php>
- [4] United States. (2010). Caught on camera. Quantico, VA: Federal Bureau of Investigation.
- [5] Symes, P. D. (2004). *Digital video compression: [featuring: JVT / H.264 / MPEG-4 part 10, the new compression standard]*. New York: McGraw-Hill.
- [6] Harte, L. (2006). *Introduction to MPEG: MPEG-1, MPEG-2 and MPEG-4*. Fuquay-Varina, NC: Althos Pub.
- [7] Mitchell, J. L. (1996). *MPEG video compression standard*. New York: Chapman & Hall.
- [8] Sullivan, G. J., & Wiegand, T. (2005). *Video Compression - From Concepts to the H.264/AVC Standard. Proceedings of the IEEE*, 93(1), 18-31. IEEE.
- [9] How Video Compression Works (2007). Electrical Engineering Times. Retrieved February 3, 2012 from the ElectricalEngineering Times website: <http://www.eetimes.com/design/signal-processing-dsp/4017518/How-video-compression-works>
- [10] FourCC (2011). FourCC.org. Retrieved January 25, 2012 from the FourCC website: <http://www.fourcc.org/>