TAMPERE UNIVERSITY OF TECHNOLOGY

Sami Hautamäki

ForeVid: an Open Source Software for Forensic Video Analysis

Master of Science Thesis

Examiner: Tarmo Lipping
Examiner and topic approved in the council meeting of the Faculty of Computing and Electrical Engineering on April 7, 2010

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Tampere University of Technology
Sami Hautamäki: ForeVid: an Open Source Software for Forensic Video Analysis
Master of Science Thesis, 45 pages, 2 Appendix pages
January 2011
Major: Signal processing
Examiner: Tarmo Lipping
Keywords: ForeVid, Avisynth, Forensic, Video processing

The main aim of the thesis was to develop a free software application for processing surveillance videos. Methods discussed in this thesis are mainly incorporated in the developed application. The open source Avisynth environment is intensively used both in the application for filtering video clips and in the thesis for gaining more deeper understanding of how video processing works. In this thesis I focus mostly on processing digital video material. Digital imaging is usually much more accurate than analog, but because the lack of storage capacity makes video compression necessary, the quality of video is often significantly decreased.

Before starting actual programming process, it was necessary to find out what features are essential for software used in forensic video editing. In this task the guidance given by forensic experts in NBI play big role. Main tools used in this project were: Python 2.6 as basic programming tool, PyQt 4.0 was used creating GUI for the software, Avisynth provides the tools for video editing, FFMPEG is being used for video compression, PDF library is being used for creating PDF files.

Open source software created in this project, enables all basic functions needed in forensic video processing. This sofware was named ForeVid for it's Forensic video editing features. With this software it is possible to resize, sharpen and cut video. Avisynth that is being used for powering video editing features, offers wide variety of filters. In ForeVid the user can take still photos and create videos or PDF documents.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn koulutusohjelma
Sami Hautamäki: ForeVid: Avoimen lähdekoodin ohjelmisto forensiseen video analysointiin
Diplomityö, 45 sivua, 2 liitesivua
Tammikuu 2011
Pääaine: Signaalinkäsittely
Tarkastajat: Tarmo Lipping
Avainsanat: ForeVid, Avisynth, forensiikka, Videon käsittely

Tämän diplomityön tärkeimpänä osana oli tuottaa viranomaiskäyttöön soveltuva videonkäsittely-ohjelmisto videovalvontatallenteille. Sovelluksessa hyödynnetään diplomityössä käsiteltäviä menetelmiä. Kehitetty sovellus perustuu Avisynth ympäristöön. Avisynth ympäristöä käytetään sekä ohjelman luomiseen, että videoeditointiominaisuuksien tutkimiseen. Tarkastelussa keskitytään lähinnïä digitaalisen kuvamateriaalin käsittelyyn, silä suurin osa tämän päivän valvontavideo-materiaalista on digitaalista. Vaikka digitaalisen kameran kuvanlaatu olisikin huomattavasti parempi kuin analogisen edeltäjänsä, pakottaa tallenustilan puute turvautumaan datanpakkaukseen. Datanpakkaus saattaa olla useissa tapauksissa erittäin voimakasta, jolloin menetetään huomattava osa informaatiosta.

Ennen varsinaisen ohjelmointityön aloittamista, oli välttämätöntä selvittää millaisia vaatimuksia forensinen videoeditointi ohjelmistolle asettaa. Tässä työssä Keskusrikospoliisilla työskentelevien experttien asiantuntemus osottautui erittäin tärkeäksi. Projektissa käytetyt pääasialliset työkalut olivat: Python 2.6, joka toimi perus ohjelmointi työkaluna, PyQt 4.0:n avulla toteutettiin ohjelman graafinen käyttöliittymä, Avisynth tarjosi työkalun videon varsinaiseen editointiin, FFMPEG:n avulla toteutetaan editoitujen videoiden pakkaus, PDF kirjaston avulla luodaan PDF tiedostot.

Työssä toteutettu ohjelmisto mahdollistaa kaikki perustoiminnot forensiselle videotutkinnalle. Ohjelma sai nimen ForeVid sen forensisten videon editointi ominaisuuksen vuoksi. Ohjelman avulla voidaan suorittaa mm. suurennus, terävöinti ja leikkausoperaatioita. Pohjana käytettävä Avisynth editointiympäristö tarjoaa erittain monipuolisen suodinvalikoiman. Ohjelma mahdollistaa niin still kuvien, videoiden kuin PDF dokumenttienkin tallentamisen.

ABBREVIATIONS

ACM	Audio Compression Manager
ATM	Automatic Teller Machine
AVC	Advanced Video Coding
AVI	Audio Video Interleave
AVS	Avisynth Script File
CCTV	Close Circuit TV
DIB	Device-Independent Bitmap
DLL	Dynamic-Link Library
DTG	Desk Top Grabber
DVB	Digital Video Broadcasting
DVD	Digital Video Disc or Digital Versatile Disc
DVI	Digital Visual Interface
DVR	Digital Video Recorder
FLV	Flash Video
GPL	General Public License
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
IP	Internet Protocol
IR	Infrared
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector for ITU
JPE	Joint Video Team
MAE	Mean Absolute Error
MKV	Matroska Video
MP4	MPEG-4 Part 14
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MSN	Multi Sensor Network
NBI	National Bureau of Investigation
PDF	Portable Document Format
RGB	standard for red, green, and blue color model
SQL	Structured Query Language
TIFF	Tagged Image File Format
TUT	Technical University of Tampere
VCEG	Video Coding Experts Group

VDR	VirtualDub frameserver
VGA	Video Graphics Array
XML	eXtensible Markup Language
YUV	Color Space Model

ACKNOWLEDGEMENTS

I would like to deeply thank the various people who, during the several months in which this endeavor lasted, provided me with useful and helpful assistance. Without their help and expertise, this thesis would likely not have matured.

First, I would like to thank Antti Lehmussola for creating reason for this thesis and being essential help in the programming and design process. Equally as much thanks goes to my instructor professor Tarmo Lipping who helped me a lot in layout and language issues.

Secondly, I would like to thank all personnel in National Bureau of Investigation (NBI) and Technical University of Tampere (TUT) for making it pleasant to work during my studies.

Third, I want to thank the support of our institutions, the Forensic Laboratory of the NBI and the Department of Computer Science of the TUT.

Fourth, I want to thank all the people who's previous work helped me to do mine. That includes creators of the Avisynth and all other programs that are essential parts in the ForeVid or helped in it's creation.

Most important, to Katri Ekola, who put up with a string of trips, and odd working hours.

Kauhava, January 11, 2011

Sami Hautamäki

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This thesis is concerned with processing of surveillance videos according to the needs of forensic science. Development of surveillance technology has been exponential, especially after the attack in September 11, 2001. The amount of video material to be handled grows rapidly as surveillance technology becomes more an more common all around the World.

The main aim of the thesis was to develop a free software application for processing surveillance videos. Methods discussed in this thesis are mainly incorporated in the developed application. The open source Avisynth environment is intensively used both in the application for filtering video clips and in the thesis for gaining more deeper understanding of how video processing works. In this thesis I focus mostly on processing digital video material. Digital imaging is usually much more accurate than analog, but because the lack of storage capacity makes video compression necessary, the quality of video is often significantly decreased. Typically video data is processed in the same way as images, frame by frame. However, time dependence of the frames in a video offers additional possibilities for video data processing. One can, for example, improve video quality and decrease the noise level for more detailed images. If we can recognize object movement in the video we can try to remove shaking camera effect or even use more advanced operations like super-resolution.

In chapter 2 an introduction to forensic video analysis is given. The general purpose as well as the problems faced in video analysis for forensics are described. In chapter 3 the most common methods and algorithms for surveillance video analysis are presented. The main aim of the thesis work was to develop a new open source forensic video analysis application. The developed application was named ForeVid. The application, its functions and user interface are described in chapter 4 of this thesis. Finally, in chapter 5 I draw some conclusions and make suggestions for future work.

2. FORENSIC VIDEO ANALYSIS

In this chapter a background on video surveillance is given with the emphasis on forensic video analysis. Also, an overview on available forensic video editing software is presented and a need for yet another application is discussed. Firstly I look at video surveillance in general. I also take a fast look at surveillance cameras to give some idea on what kind of data we are getting from our sensors. Secondly I look at programs available for forensic video editing.

2.1 Introduction to Forensic Video Analysis

Forensic video analysis incorporates scientific examination, comparison and/or evaluation of video material for legal matters. Although a relatively new field in forensic science, it has proven its effectiveness in clarifying events. As video surveillance develops and image quality improves, more powerful analyzing methods are needed to handle the growing amount of data and to get greatest advantage of it. Ongoing development and evolution in video processing methods makes it possible to use these new methods in forensic analysis. As video technology develops, so do the processors used in the hardware platforms for video processing. Therefore, when more complex algorithms are used, better processors help making those methods operate in reasonable time. Nowadays video data is usually in digital form and old analog systems are being replaced by newer digital IP-based systems as shown in Figure 2.1. Cameras may be analog but the actual storing is made in digital form. If video is still saved in analog form, it must be digitalized before it can be processed and analyzed. Video processing is usually quite similar to photo editing. Video is typically processed frame by frame like images. Images have typically much better resolution and much lower compression ratio. More advanced methods like de-shaker, super resolution analysis or motion detection use multiple frames to enhance a frame.

Some problems in the surveillance video analysis come from the fact that, video processing methods are generally created according to the needs of the entertainment industry. For entertainment industry actual data doesn't mean as much as the visual experience. This concerns most video editing and compression methods. Video compression taking place in the camera is actually a much greater problem for forensic analysis than for video processing for entertainment applications because it deteriorates the video badly. In surveillance video applications, compression is

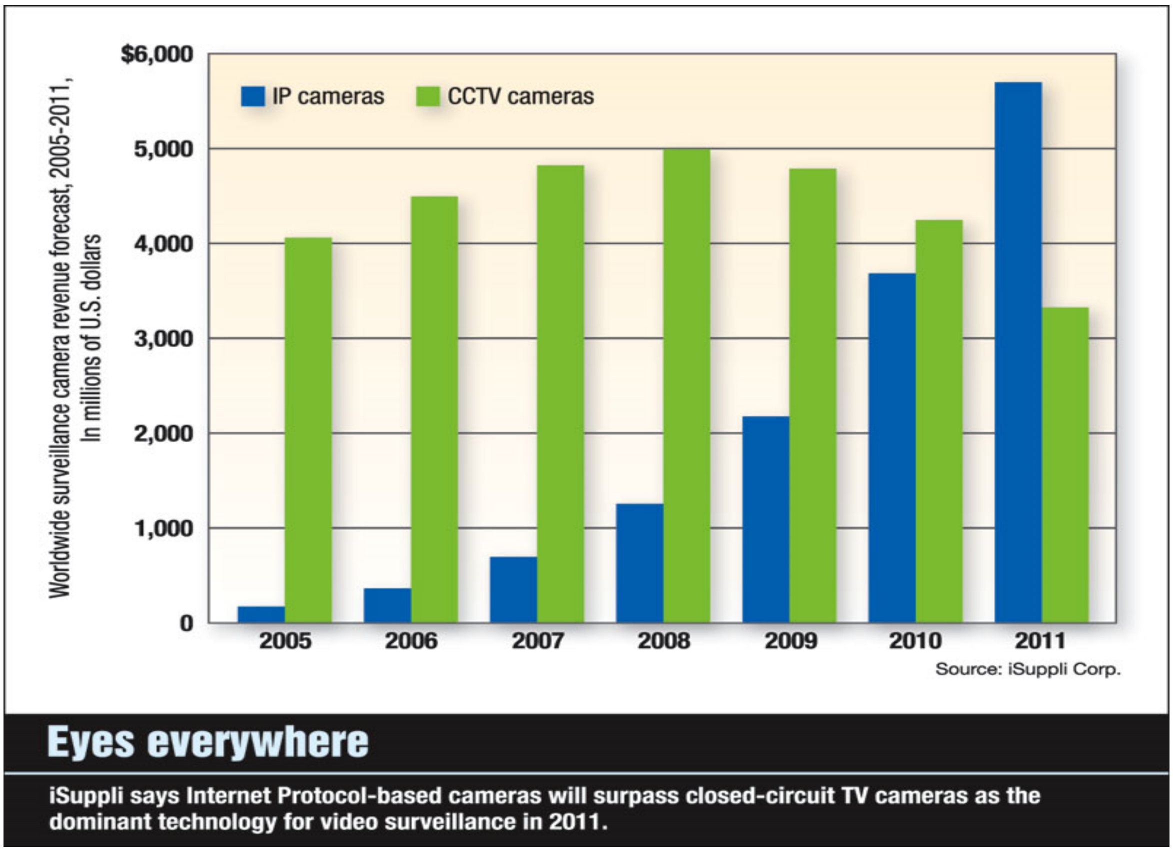


Figure 2.1: IP-technology based surveillance systems are taking over the industry.[6]

usually quite heavy. Still compression is considered necessary because available hard drive capacity is usually quite limited. Even as storage capacity grows, growing resolution and amount of cameras used for surveillance eats the development up. Video compression is a process that should be avoided to the last moment. Basically every process that loses information can be considered as harmful. Selecting right tools from the available video processing toolkit can be quite a challenge for a person who doesn't know the basics of video technology and processing. Many of the basic methods fit for the forensic applications but as surveillance business grows, more and more algorithms are developed purely for forensic use. Some of those methods can be quite advanced including face recognition and super resolution. With the advancement in technology, video surveillance has gone to the next level of facial feature recognition, in not only finding people from previously enrolled database but also trying to understand the moods and intentions of strangers through pre-programmed attributes and alert people upon detecting a facial attribute associated with threats. This technology is promising and can be considered as a basic and beginning stage of 'Artificial Intelligence' in facial recognition. Some principles of facial feature recognition are presented in Figure 2.2.

Because we are processing data for legal matters it is important to know what

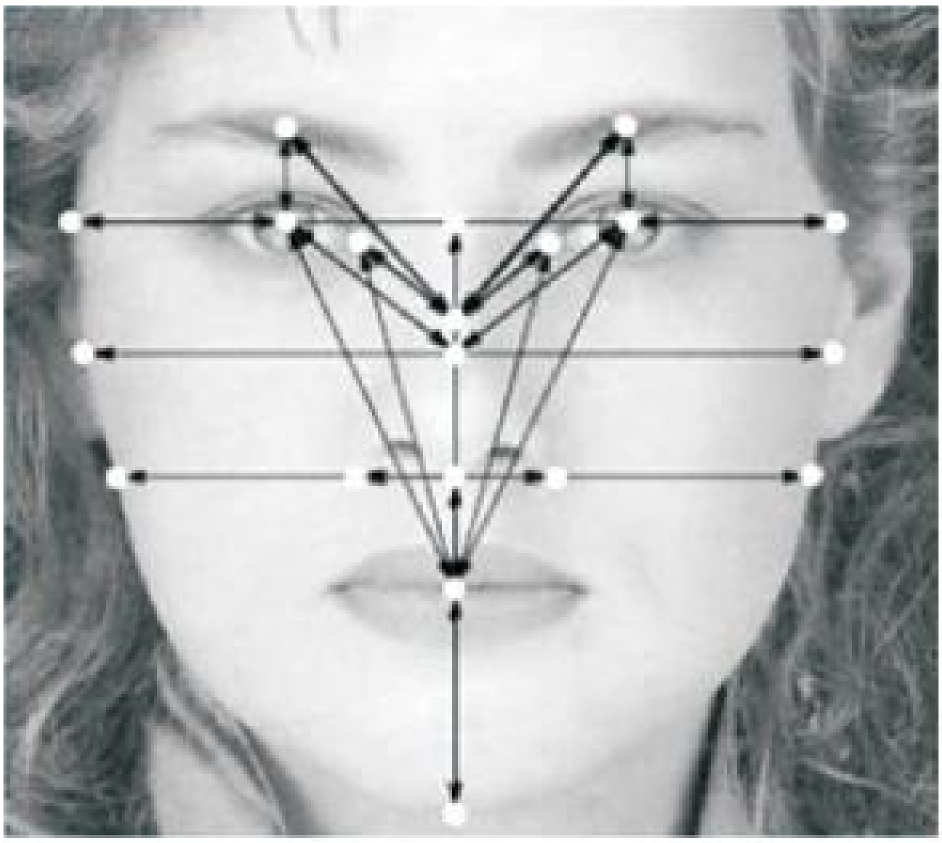


Figure 2.2: With the advancement in technology, video surveillance has gone to the next level of facial feature recognition. In this image the basic measurement principles for facial feature recognition are presented.[19]

processing steps the video has gone through. Different situations allow different kinds of methods to be used. It is obvious that the value of video material as an evidence decreases if any questionable methods have been used. Forensic scientist can never forget legal rights of the suspect. When using more powerful methods that, for example, are based on probability theory or some data model, it is important to keep in mind how the results will be used. They can direct the investigation to the right track or push it badly offtrack. So it is important that the value of processed data is estimated.

The future of security and surveillance brings up advancement in data processing towards artificial intelligence and mobility of surveillance. In terms of mobility, cameras mounted on wheels are already available and the future could be that more and more cameras are mounted both in land, sea and air vehicles. These vehicles could be even unmanned doing only surveillance missions by preprogrammed instructions, taking surveillance to hitherto unknown nooks and corners of capabilities. Similarly, artificial intelligence technology has a lot of promise in processing data and reacting appropriately to counter perceived threats even without the intervention of human beings for prolonged periods[1]. Future surveillance will strongly depend on the networks using IP-technology, creating MSN (see Figure 2.3). This MSN uses advanced artificial intelligence technology to process huge amount of data created by security cameras and other security sensors.

2.1.1 Surveillance Cameras

When looking for a security surveillance system for home or business use, one of the decisions that has to be made is whether to go with a digital or an analog camera. Even when digital cameras present the future of the surveillance technology, it is not necessarily simple decision to make as both types of cameras have their advantages and disadvantages. In the following I will explore some of these differences in more detail. The key difference between these two camera types is the way they deliver the video signal. In analog cameras the video signal is delivered in a format that can be received by a television or other analog receiver such as a VCR or a monitor. A digital-based camera digitizes the video signal using a specialized encoder. Digital camera can act as a network device, thus allowing the captured video images to be viewed not only through an existing network but also through a web browser that can be accessed through the Internet. Analog and digital-based video cameras can transmit signals either wirelessly or through a wired connections such as Cat-5 cables. IP-based cameras have the added benefit of being able to use switches, hubs, and routers that allow the Cat-5 network to be expanded to much broader ranges. Let us take a closer look at some of the pros and cons of each camera type. It

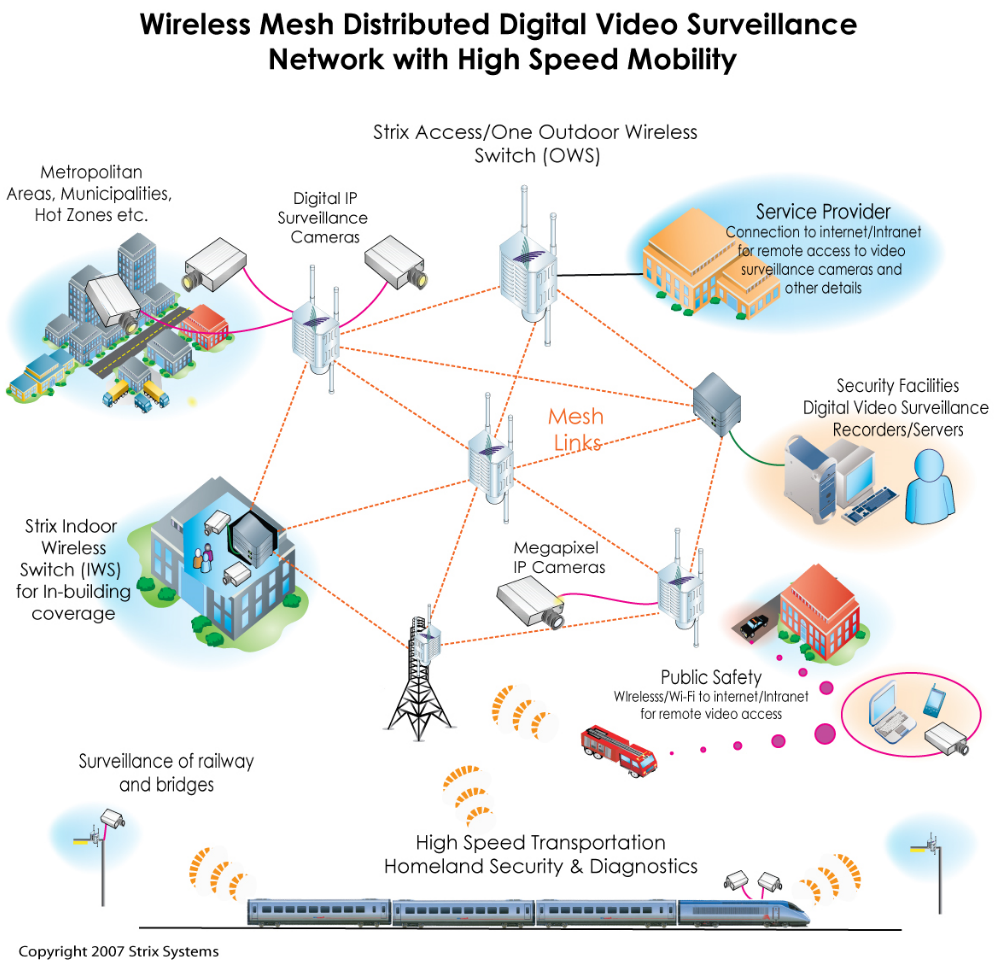

Wireless Mesh Distributed Digital Video Surveillance
Network with High Speed Mobility
Strix Access/One Outdoor Wireless Switch (OWS)
Metropolitan Areas, Municipalities, Hot Zones etc.
Digital IP Surveillance Cameras
Service Provider
Connection to internet/Intranet for remote access to video surveillance cameras and other details
Mesh Links
Security Facilities
Digital Video Surveillance Recorders/Servers
Strix Indoor Wireless Switch (IWS) for In-building coverage
Megapixel IP Cameras
Public Safety
Wirelesss/Wi-Fi to internet/Intranet for remote video access
Surveillance of railway and bridges
High Speed Transportation
Homeland Security & Diagnostics
Copyright 2007 Strix Systems

Figure 2.3: Future surveillance will strongly depend on the networks with IP-technology, creating MSN multi sensor network.[11]

should be kept in mind that IP cameras are evolving rapidly so the features and opportunities provided by IP cameras are developing rapidly.

Analog Cameras

Analog camera encodes image and sound information and transmits it as an analog signal: one in which the message conveyed by the broadcast signal is a function of deliberate variations in the amplitude and/or frequency of the signal. Analog cameras use old methods, so resolution is lower than in up to date digital cameras. In Finland the most common analog system is PAL. Saving can be done digitally even if an analog camera is used, meaning that the quantization operation is performed before saving and the data becomes digital. Typically analog components are much more expensive than digital components.[7]

Benefits of analog cameras include:

- Analog cameras generally have lower prices than IP cameras.
- Analog cameras often have a larger variety of designs such as mini covert cameras to large PTZ models. Given unique surveillance needs one may find a suitable analog camera more easily than a similar digital one.
- Compatibility with existing system: if there is a need to find a camera that suits an existing surveillance system, a compatible analog camera might be easier to find.

Disadvantages of analog cameras include:

- Many of the basic analog cameras lack some of the more advance features, which are cheaper to engineer using digital components and signal processing.
- If there is a need to install a wireless surveillance system, analog systems can have interference problems. More importantly, the resulting signals cannot be encrypted. This can potentially mean that someone else can interrupt the transmission.
- When surveillance needs to encompass a wide area, analog cameras may not be the best choice. Analog cameras generally do not accommodate big distances, and getting them to work over broad ranges can be difficult.

Digital Cameras

Digital camera records video by means of electronic image sensor. Electronic image sensor records video frame by frame, giving bit value for every pixel. Digital cameras are gradually replacing analog cameras in most applications. The resolution of digital cameras can be much higher and functions much more flexible than in analog cameras. Digital components are nowadays much cheaper and this lowers the manufacturing cost of digital cameras. Because of digitalization and IP-network capabilities, cameras can be connected directly to the network by cable or wirelessly enabling data to be transferred wherever it is needed.[7]

Benefits of Digital cameras include:

- Better wireless connection. Digital cameras have encryption built right into them providing a more secure network. Interference is also not a problem with IP-based models.
- IP cameras can utilize existing wiring. Because digital cameras can usually act as independent network devices, they can often take advantage of existing network wiring within a house. This can make the installation task much easier.

Disadvantages of Digital cameras include:

- If a system has too many unnecessary features built into each camera, the cost can be higher than if using analog versions.
- Higher bandwidth required. Digital cameras require more bandwidth than analog cameras.

IR Cameras

Last but not least, IR-cameras are discussed. IR-camera is a good choice if surveillance is done at night in low light areas. The answer to this problem is an infrared camera. Infrared camera is an ideal product for anyone who needs to capture images in the dark. One thing that is not often mentioned when using IR-cameras is that colors can be really confusing. Dark colors for the eye do not necessarily seem dark in IR-cameras, for example black clothes can seem lighter than white clothes because the light intensity in the visible wavelength band does not necessarily correspond to the intensity in the IR band.[7]

An infrared camera uses infrared light instead of the visible light spectrum in order to produce better images in complete darkness or low light conditions. Night vision cameras only record in black and white, but some will record color during the day. A regular camera can become an infrared camera with the use of infrared illuminators. The illuminator lights the area under surveillance with infrared light so that a regular camera can record black and white images with the use of infrared radiation, which the naked eye cannot see.[7]

Infrared cameras are not to be confused with a day/night cameras. Day/night cameras can record in low light, but not in zero light and do not use infrared lighting. Infrared cameras are also available for a CCTV camera system. The complexity of infrared camera system is entirely up to the user. The cameras are often lightweight and easy in both use and setup.[7]

In order to fully understand the clear images that can be produced in complete darkness, one has to see it. Most manufactures have examples of the images or videos their cameras are capable of producing available on their websites. [7]

3. PROCESSING METHODS FOR SURVEILLANCE VIDEOS

In this section main focus is on processes that are helpful in forensic video analysis. Processes affecting the creation of the video file are determined and their influence on later edition of the video is studied. Main focus is on Avisynth-filters that are used in ForeVid. First we study effects of compression on the image and video. Secondly we study some Avisynth-filters that may be useful in forensic use.

3.1 Image compression

In this chapter a closer look at three different image compression methods is taken. Selected methods are common and widely used. Methods are being evaluated for their functionality in forensic use.

JPEG is probably the most common image format used by digital cameras and other photographic image capture devices. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG is not lossless so it has limited use in forensics. Still the incoming material can be in JPEG format so it's main features should be known. There are variations of the standard baseline JPEG that are lossless; however, these are not widely supported[2]. JPEG2000 is a different standard from JPG, but is not studied in this thesis because it is not currently used in ForeVid[3].

The BMP file format, sometimes called bitmap or DIB file format (for device-independent bitmap), is an image file format used to store bitmap digital images, especially in Microsoft Windows and OS/2 operating systems. Many graphical user interfaces use bitmaps in their built-in graphics subsystems; for example, the Microsoft Windows and OS/2 platforms' GDI subsystem, where the specific format used is the Windows and OS/2 bitmap file format, usually named with the file extension of .BMP or .DIB. Advantages of this type of images is that they are lossless, but the downside is high requirement for hard drive space.

TIFF is a file format for storing images, popular among Apple Macintosh owners, graphic artists, the publishing industry, and both amateur and professional photographers in general. As of 2009, it is under the control of Adobe Systems. Originally created by the company Aldus for use with what was then called "desktop publishing", the TIFF format is widely supported by image-manipulation applications, by publishing and page layout applications, by scanning, faxing, word processing, optical character recognition and other applications. Adobe Systems, which acquired Aldus, now holds the copyright for the TIFF specification. TIFF has not had a major update since 1992, though several Aldus/Adobe technical notes have been published with minor extensions to the format, and several specifications, including TIFF/EP (ISO 12234-2) and TIFF/IT (ISO 12639) have been based on the TIFF 6.0 specification. The ability to store image data in a lossless format makes a TIFF file a useful image archive, because, unlike standard JPEG files, a TIFF file using lossless compression (or none) may be edited and re-saved without losing image quality. This is not the case when using the TIFF as a container holding compressed JPEG.

3.2 Video compression

Compression is one of the most crucial procedures in forensic video analysis. In this section an overview on several video compression methods is given to get better insight into their performance. By comparing the error between the original video and the compressed-decompressed version of the video using similar compression ratios one can evaluate how a certain compression algorithm affects a video. Another important quality criterion is the visual appearance of the compressed-decompressed material compared to that of the original.

H.264 is only compression method that is used in ForeVid. H.264 also known as AVC(Advanced Video Coding) and MPEG-4 Part 10 is a video compression standard. The first version of the standard was completed in May 2003. H.264/AVC is block-oriented motion-compensation-based codec standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG), and it was the product of a partnership effort known as the Joint Video Team (JVT). The ITU-T H.264 standard and the ISO/IEC MPEG-4 AVC standard (formally, ISO/IEC 14496-10 - MPEG-4 Part 10, Advanced Video Coding) are jointly maintained so that they have identical technical content. Some of the applications that use H.264 are Blu-ray Disc, DVB broadcast, direct-broadcast satellite television service, cable television services, and real-time video conferences. Main advantage of H.264 over its predecessors is its capability to compress video with only minor loss of quality. The CCTV or Video Surveillance market has included the technology in many products. With the application of the H.264 compression technology to the video surveillance industry, the quality of the video recordings

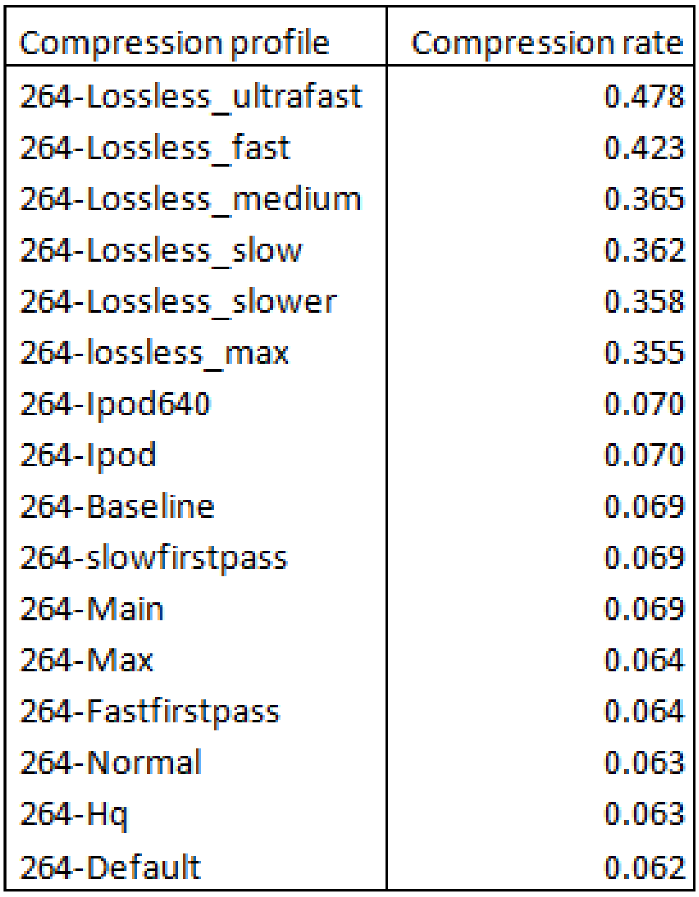
Table 3.1: MSE is used for determining errors of different compression profiles. Results for compression comparison are obtained using the MSU video quality measurement tool. Original video was coded with H.263 compression method which causes slight quantization error in lossless profiles.

Compression profile	AVG Y-MSE
264-Lossless_fast	0,14
264-lossless_max	0,14
264-Lossless_medium	0,14
264-Lossless_slow	0,14
264-Lossless_slower	0,14
264-Lossless_ultrafast	0,14
264-Max	3,33
264-Hq	3,66
264-Normal	4,18
264-Default	5,19
264-slowfirstpass	5,62
264-Fastfirstpass	5,79
264-Ipod	6,55
264-Ipod640	6,55
264-Baseline	6,59
264-Main	6,59

became substantially improved. Starting from 2008, the H.264 technology is sometimes considered as synonymous with "high quality" video.[4]

There are five wrapper possibilities for the user to select in ForeVid. They are AVI, FLV, MOV, MKV and MP4. ForeVid has several profiles for saving video file and here I study some of them. When comparing video files of different kinds of compression profiles it was discovered that even lossless doesn't necessarily means lossless. Compression error for each profile is shown in table 3.1. If video uses some other compression method like H.263 the translation to H.264 makes slight difference to the video due to YUV and RGB transformations and quantization selections. Compression rate for each profile is shown in table 3.2.

Table 3.2: Comparing compression rates for different compression profiles. As can be noticed lossless compression methods use much more hard drive space than lossy methods. Lossless methods still manage to halve the disk space used. Lossless methods that are fast to calculate are taking most disk space.



Compression profile	Compression rate
264-Lossless_ultrafast	0.478
264-Lossless_fast	0.423
264-Lossless_medium	0.365
264-Lossless_slow	0.362
264-Lossless_slower	0.358
264-lossless_max	0.355
264-Ipod640	0.070
264-Ipod	0.070
264-Baseline	0.069
264-slowfirstpass	0.069
264-Main	0.069
264-Max	0.064
264-Fastfirstpass	0.064
264-Normal	0.063
264-Hq	0.063
264-Default	0.062

3.3 Video editing with Avisynth

Avisynth is a powerful tool for video post-production. It provides many different ways for editing and processing videos. Avisynth works as a frame server. It makes possible instant editing without the need of temporary files. Avisynth script files work in principle with all media applications and players capable of opening AVI files. For instance, Ligos MPEG encoder will not let the user to choose anything but straight AVI files in the file selector. But if AVI is loaded and the name is manually changed in the filename box to .AVS - it works perfectly. Because Avisynth works like a frame server, filters are being applied 'on-the-fly'.[8]

Avisynth does not provide a graphical user interface, but instead uses a script system that allows advanced non-linear editing. It is remarkably powerful and is a very good way to manage projects in a precise, consistent, and reproducible manner. Because text-based scripts are readable, projects are inherently self-documenting. The scripting language is simple yet powerful and complex filters can be created from basic operations to develop a sophisticated palette of useful and unique effects.[8]

Not all tasks are equally easy to do in Avisynth. In general Avisynth makes it very easy to adjust the appearance of a video, and doing fast and easy editing. In general it's not easy to use Avisynth for cutting videos or doing advanced post-processing. For that it is much better to use programs with GUI. That is one of the main reasons for creating ForeVid. In general Avisynth is a very good tool when having to compress video. Many filters are written specifically for tasks regarding video pre-processing when compressing or decompressing video. There are many well written plugins to help you make your movie more compressible, reduce noise, adjust color and resize your source material. And because Avisynth works as a frame server, you often do not need a temporary file before you compress it.[8]

3.3.1 Media file filters

Media file filters are used to read or write media files. Usually they produce source clips for processing. There are many different kinds of filters in this category. Video codec that has been used affects the filter to be selected. ForeVid uses three different kinds of Media file filters: AviSource, DirectShowSource and FFMPEG.[5]

AviSource

AviSource takes one or more file names as quotes and opens the files using either the Video-for-Windows "AVIFile" interface, or Avisynth's built-in OpenDML code (taken from VirtualDub). AviSource filter can read any file with an AVIFile handler.

This includes not only AVI files but also WAV files, AVS files, and VDR files. When given multiple filenames as arguments, the clips will be spliced together with UnalignedSplice. The AviSource filter examines the video to determine its type. Type is used to determine which handler is appropriate to use, the AVIFile handler or the OpenDML handler. Only the OpenDML handler can read larger AVI files than 2GB, but only the AVIFile handler can read other file types like WAV, VDR and AVS. There is also built-in support for ACM audio.[5]

DirectShowSource

DirectShowSource opens a video using DirectShow. DirectShow is codec that is based in windows direct show. DirectShow has some minor problems with frame handling and it should be used only as secondary choice. DirectShowSource can read much greater number of codecs than AviSource. For better performance a newer DirectShowSource2 is available for download.[5]

FFmpegSource

If media file contains MPEG-4 ASP video, FFmpegSource can be used. Many non-AVI files contain video with variable framerate (AVI files always have a constant frame rate though), and in that case it is important to make sure the following two things:

- Frame rate and the number of frames must not be changed in Avisynth. If frame rate is changed and timecodes-file is not changed manually, video and audio in final encoding will be out of sync.
- When muxing the encoded video and audio, the timecodes file must be used again. If this in not done video and audio in final encoding will be out of sync.

The main reason for this is that FFmpegSource opens the video as it is. It doesn't add or remove frames to convert it to constant framerate video to ensure sync.[5]

3.3.2 Color conversion and adjustment filters

These filters can be used to change the color format or adjust the colors of a clip. Colors can be adjusted in many formats, typically RGB and YUV. When adjusting brightness in a video, the user should consider which kind of filter to use. ForeVid version 1.0 uses Levels-function for this task, however, it is more recommended to use Tweak- or ColorYUV-filter, because Levels also changes the chroma of the clip.[5]

GreyScale

GreysScale filter converts the input clip to greyscale without changing the color format. If clip is based on YCbCr formats, the chroma channels are set to 128. If clip is based on RGB formats, the conversion produces the luma using the coefficients given in the matrix parameter. By setting matrix="rec709", the clip is converted to greyscale using Rec.709 coefficients[5]. By setting matrix="Average" the luma is calculated as:

$$(R+G+B)/3. \tag{3.1}$$

Levels

Levels is a filter that adjust brightness, contrast and gamma. Input pixel values can be determined to be treated as pure black or pure white by inlow and inhigh parameters. Output value corresponding to black and white can be determined by outlow and outhigh parameters. Gamma parameter controls the degree of nonlinearity in the conversion. Level filter uses the conversion function:

$$out = \frac{in - inlow}{inhigh - inlow}^{\frac{1}{\text{gamma}}} * (outhigh - outlow) + outlow \tag{3.2}$$

If the video file is processed in the YUV mode, Levels only gamma-corrects the luma information, not the chroma. The reason for this is that gamma correction is really an RGB concept, and it is much more complex to do in YUV. However, if gamma = 1.0, the filter should have the same effect in RGB and YUV modes. For adjusting brightness or contrast it is more recommended to use Tweak- or ColorYUV-filter, because Levels also changes the chroma of the clip.[5]

Tweak

Tweaks are preferred as small modifications intended to improve a system. In Avisynth Tweak filter can be used to adjust the hue, saturation, brightness, and contrast of a video clip. For preventing banding, the interp-value interpolates the adjusted saturation.[5]

ColorYUV

ColorYUV allows many different methods of changing the color and luminance of the videos. All settings for this filter are optional. All values are defaulting to "0" or false. Gain, offset, gamma and contrast can be set independently for each channel.

A saturation of 0.8 gives for example:

$$contV = -0.2 * 256 = -51.2. \tag{3.3}$$

It must be noted that in Tweak YUV values will always be clipped to valid TV-ranges, but here opt="coring" has to be specified.[5]

3.3.3 Geometric deformation filters

These filters can be used to change image size, process borders or make other deformations of a clip. From this filter category the ForeVid uses flip, rotation and resize functions.[5]

FlipHorizontal / FlipVertical

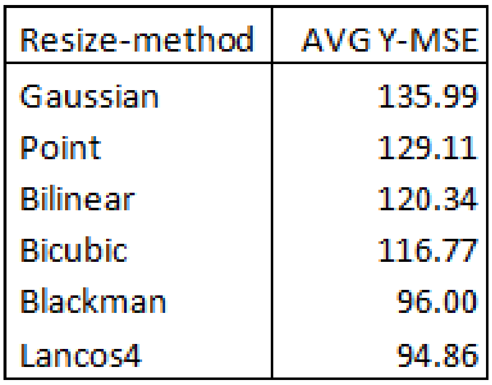
This filter flips the video from left to right or upside down. It is useful for dealing with some video codecs which output everything upside down. This function is known as mirror function in many other programs.[5]

Resize filters

The resize filters re-scale the input video frames to an arbitrary new resolution, using different re-sampling algorithms[5]. MSE comparison of the filters is presented in table 3.3. The filters in AviSynth internal filter list include:

- GaussResize: GaussResize uses a gaussian resize with sharpness parameter p (default 30) which can be adjusted. Range of p is from 1 to 100, with 1 being very blurry and 100 being very sharp.[5]
- PointResize: PointResize is the simplest possible resize. It uses ether Point Sampler or Nearest Neighbour algorithm and the result is usually a very blocky image. This filter should only be used if the intention is to have inferior quality or in need of clear pixel drawings. It is useful for magnifying small areas for examination.[5]
- BilinearResize: BilinearResize uses standard bilinear filtering for re-scaling the frame. It is good basic resize filter especially when shrinking.[5]
- BicubicResize: BicubicResize is quite similar to BilinearResize, except that it uses the Mitchell-Netravali two-part cubic instead of a linear filtering function. The properties of the cubic can be adjusted, values are sometimes referred to as "blurring" and "ringing" respectively. When magnifying video, BicubicResize gives much better-looking results than BilinearResize.[5]

Table 3.3: Comparison of resize-methods by Y-MSE value. By looking the numbers it is clear that Blackman and Lancos4 perform best in this evaluation. Therefore these two filters are the most recommended ones.



Resize-method	AVG Y-MSE
Gaussian	135.99
Point	129.11
Bilinear	120.34
Bicubic	116.77
Blackman	96.00
Lancos4	94.86

- BlackmanResize: BlackmanResize is created by modifying LanczosResize and it has better control of ringing artifacts for high numbers of taps.[5]

- Lanczos4: Lanczos4 is a convolution filter. Convolution functions are usually defined so that they are zero when the distance is larger than some value so that you don't have to consider every input value for every output value. For lanczos interpolation the convolution function is based on the $sinc(x) = sin(x\pi)/x$ function, but only the first four lobes are taken in lanczos4.

If input resolution is the same as output resolution along one axis, that resize will be skipped. Which one is called first, is determined by which one has the smallest down scale ratio. This is done to preserve maximum quality, so the second resize has the best possible image to work with.[5]

TurnLeft, TurnRight and Turn180

TurnLeft rotates the clip 90 degrees counter-clockwise and TurnRight 90 degrees clockwise. Turn180 rotates the image full 180 degrees.The operation is very simple and fast. It is notable that the frame size changes in 90 degree rotation from NxM to MxN.[5]

3.3.4 Pixel restoration filters

These filters can be used for image detail restoration of a clip. In forensic use this usually means sharpening image and possible noise removing. Especially sharpening is one of the most used filters.[5]

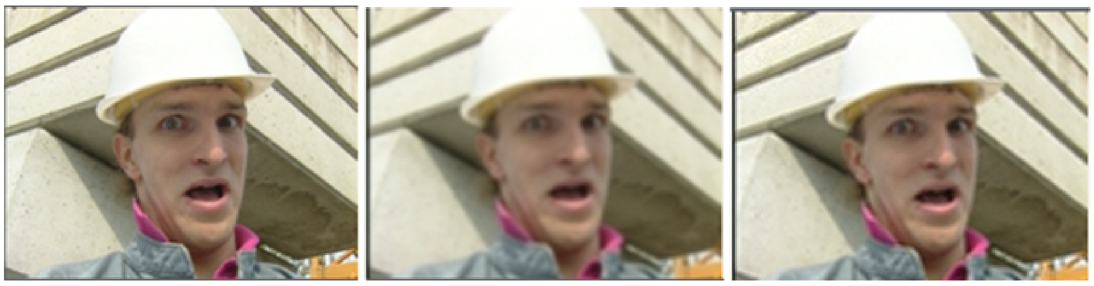


Figure 3.1: Starting from left: orginal clip, blurred clip, clip after sharpening. In visual evaluation sharpening seems to be working well. Frame is being sharpened near to original one.

Sharpen

Sharpening is one of the basic processes for forensic video enhancing. Sharping is basically a counter operation for blurring, which is commonly used for image softening. Softened image is usually more pleasant to watch, but it is not usually used in forensic because in forensic video analysis it is usually important to highlight the changes. It clarifies changes in the image providing more detailed version of the image. ForeVid has currently only one sharpening function which is called sharpen. In test arrangement, a video file is compressed using lossless max profiles and then blurred using blur function. Finally MSE is calculated for the video. Finally sharpen is used to restore the video and error is calculated again. That way it is possible to study how much the error decreased. MSE comparison of the filter is presented in figure 3.2. Error was created using the blur function with the parameter value of 1.58. That is maximum value for the particular function. Y-MSE value obtained is 74.17. The parameter value used in sharpening was 0.9. After sharpen Y-MSE was 51.96. In this test the MSE dropped by 1/3 which can be considered a lot. Visual comparison of the filter is presented in figure 3.1 The frame that is being processed has huge effects in sharpening result. If pixel values change a lot within the frame, result is not so good and more noise is being added to the frame. This results from camera motion that adds blur to the frame.[5]

GeneralConvolution

This filter performs a matrix convolution on a RGB32 clip. It uses given 3x3 or 5x5 convolution matrix. The user can also determine divider and bias for the filter. The divisor is usually the sum of the elements of the matrix. However, when the sum is zero, the divisor can be leaved to one and the bias setting to correct the pixel values. The bias could be useful if the pixel values are negative due to the convolution. After adding a bias, the pixels are just clipped to zero (and 255 if they are larger than 255).[5]

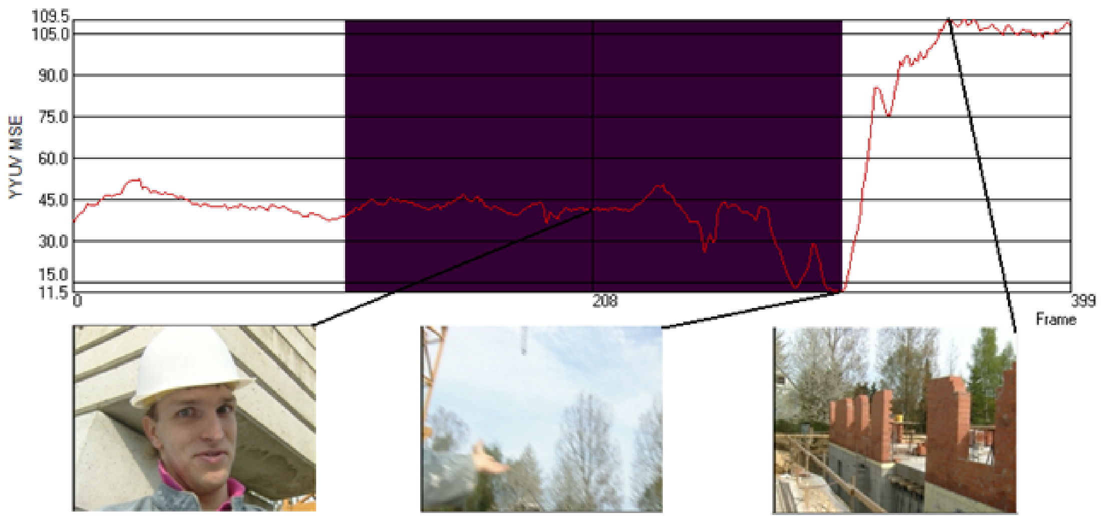


Figure 3.2: The figure shows how well sharpen works in general. The areas of similar pixel values (i.e., relatively flat areas) do not change much in blurring.

3.3.5 Timeline editing filters

These filters can be used to arrange frames in time (clip cutting, splicing and other editing). Trim function is the only one available in ForeVid from this section. Timeline editing filters include various kinds of filters from deleting frames to changing FPS.[5]

Trim

The Trim filter trims a video clip so that it includes only the frames from the first selected frame to the last selected frame. For example, Trim(40,260) removes frames which are not in the range 40-260. There are two arguments in the Trim filter separated by a comma: the first and the last frame to keep from the clip. If you put 0 for the last frame, it means "end of clip,". For more complex trimming one or more Trim functions can be summed together forming new trimmed clip. Audio is trimmed at the same time, so the user doesn't have to trim audio seperatly.[5]

3.3.6 Interlace filters

These filters can be used for creating and processing field-based material, which is frame-based material separated into fields. Field-based material usually comes from analog cameras which divide frame in two separate fields. In old displays these fields formed paired and un-paired columns, these columns can be separated or merged removing saw edge effect. Avisynth is capable of dealing with both progressive and interlaced material. This is the reason that the field-based flag exists and can be used when dealing with interlaced material[5]. Separate fields problem is shown in

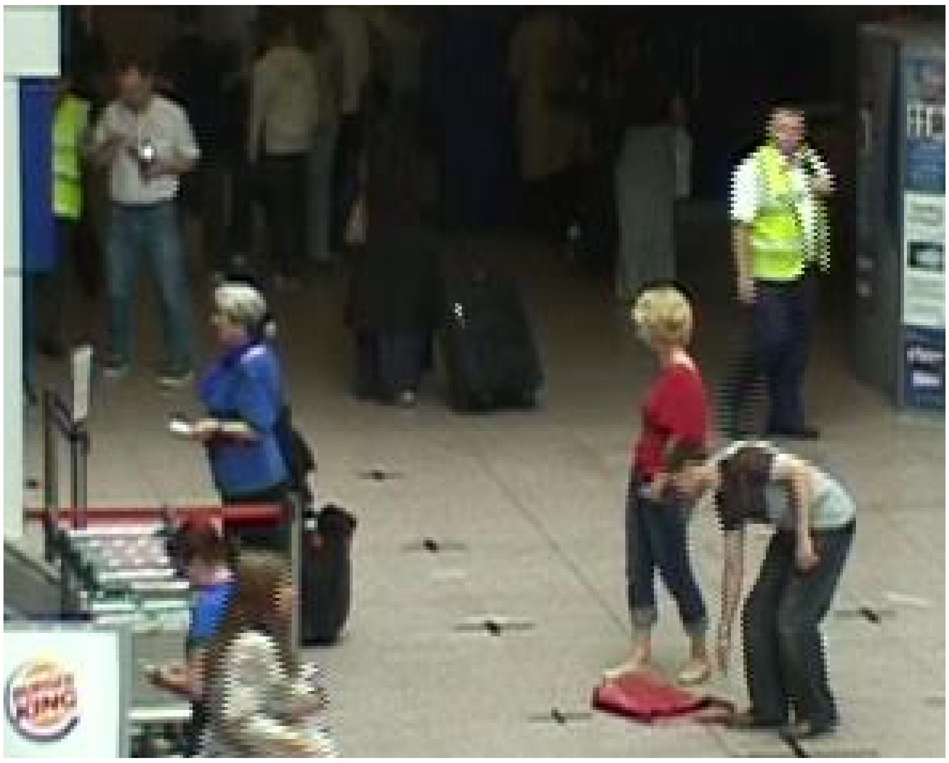


Figure 3.3: Separate fields causes edges of moving object to blur.

figures 3.3 and 3.4.

Bob

Bob takes a clip and bob-deinterlaces it. This means that it enlarges each field into its own frame by interpolating between the lines. The top fields are nudged up a little bit compared with the bottom fields, so the image will not actually appear to bob up and down. However, it will appear to "shimmer" in stationary scenes because the interpolation doesn't really reconstruct the other field very accurately. This filter uses BicubicResize to do its work. A bob filter does not really move the physical position of a field. It just puts it back where it started. If SeparateFields() is used alone then only 2 half height frames are gained: line 0 becomes line 0 of frame 0 and line 1 becomes line 0 of frame 1. Thus line 0 and 1 are now in the same place. Bob now basically resizes each frame by a factor of two but in the first frame uses the original lines for the even lines and in the second frame uses the original lines for the odd lines. If after doing a SeparateFields() each frame is resized vertically by a factor of 2, it wouldn't work right because the physical position of a field moves.[5]

Weave

Weave is the opposite of SeparateFields. It takes pairs of fields from the input video clip and combines them together to produce interlaced frames. The new clip has

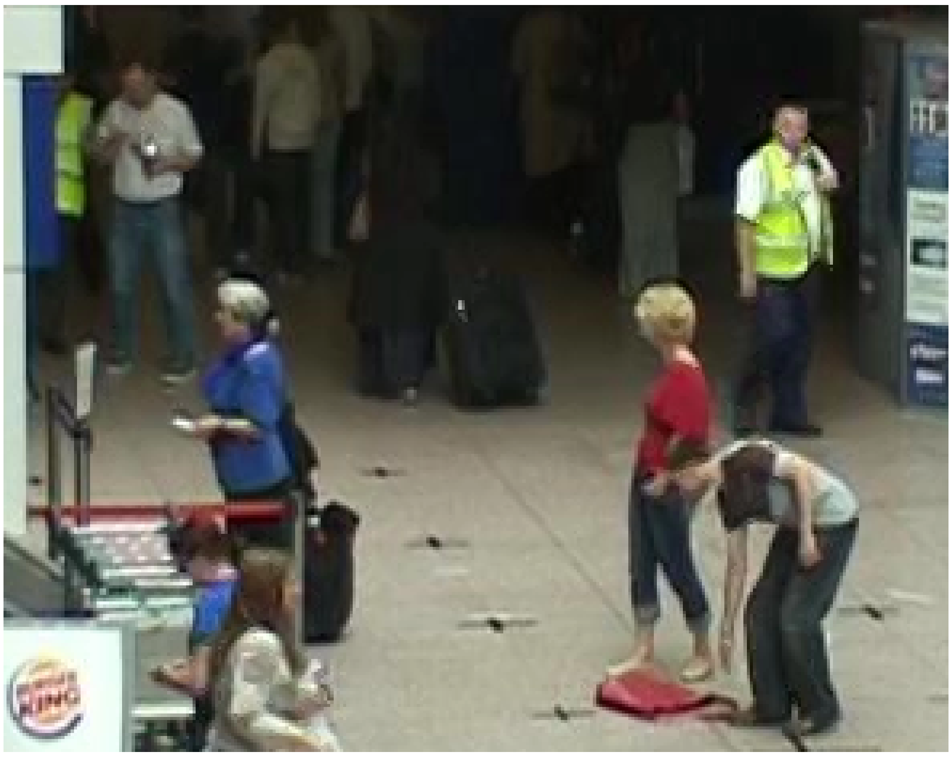


Figure 3.4: After deinterlace filter moving objects are sharper.

half the frame rate and frame count. Weave uses the frame parity information in the source clip to decide which field to put on top. If the output does not come out right, ComplementParity beforehand or SwapFields afterwards might fix the problem. All Avisynth filters keep track of field parity, so Weave will always join the fields together in the proper order. From version v2.56 this filter raises an exception if the clip is already frame-based. AssumeFieldBased can be used to force weave a second time. Prior versions did a no-op for material that was already frame-based.[5]

DoubleWeave

If the input clip is field-based, the DoubleWeave filter operates like Weave, except that it produces double the number of frames: instead of combining fields 0 and 1 into frame 0, fields 2 and 3 into frame 1, and so on, it combines fields 0 and 1 into frame 0, fields 1 and 2 into frame 1, and so on. It does not change the frame rate or frame count. If the input clip is frame-based, this filter acts just as though the clip was separated into fields with SeparateFields first. Weave is actually just a shorthand for DoubleWeave followed by SelectEven. It is recommended to use a filter like SelectOdd or Pulldown after using this filter, unless a 50fps or 60fps video is really wanted. It may seem inefficient to interlace every pair of fields only to immediately throw away half of the resulting frames. But actually, because Avisynth only generates frames on demand, frames that are not needed will never be generated in the first place. If the clip processed is field-based, like a video-camera footage, this filter won't probably be needed. But when processing NTSC video converted from film and usage of the Pulldown filter is planned, usage of DoubleWeave is needed first.[5]

3.3.7 Conditional and other meta filters

Meta filters can be used to control other filters execution. The basic characteristic of conditional filters is that their scripts are evaluated at every frame instead of the whole clip. This allows for complex video processing that would be difficult or impossible to be performed by a normal Avisynth script.[5]

4. FOREVID

ForeVid is forensic video processing software created for Finnish National Bureau of Investigation (NBI). It is based on open source components such as Avisynth. All source codes are open source and GNU licensed and therefore the software is freely modifiable. Version management and development of the ForeVid software is done by the NBI. The purpose of this software is to offer functioning tool for forensic video analysis.

ForeVid uses Avisynth as the opening and editing tool for videos. The ForeVid can be described as a binder that binds all components together and offers interface for the user. Avisynth fits for this purpose perfectly, it has numerous filters for opening and editing videos. Most of the filters are open source and free to use. The user can also download, buy or develop more filters and easily add them to the ForeVid. ForeVid software is created using Python programming language. The GUI is created using PyQt 4.0 which is Qt GUI-library moderated for Python. When video information is being acquired from Avisynth the data is given in C-type DLL-files. Some information from C-type data must be changed into Python compatible format for usage in Python. Python has tools for exporting images and PDF files. Video exporting is managed through the FFMPEG-software that allows usage of Avisynth scripts in video compression process. Structure of the ForeVid is shown in figure 4.1.

4.1 Tools used

In this section the main tools that were used in the creation of the ForeVid. The language used was Python 2.6 which is easy to learn and offers flexibility needed for software of this magnitude. ForeVid is based on Avisynth that enables video frame editing. Moving frames in the Python program is presented in section Video editing with Avisynth. C-language was used in some cases where DLL-files were required. For Video compression purposes FFMPEG offered a great tool, due to its ability to compress videos from Avisynth script. PDF tool offered a toolkit for generating PDF files from the case in hand. Main tools have detailed presentations below.

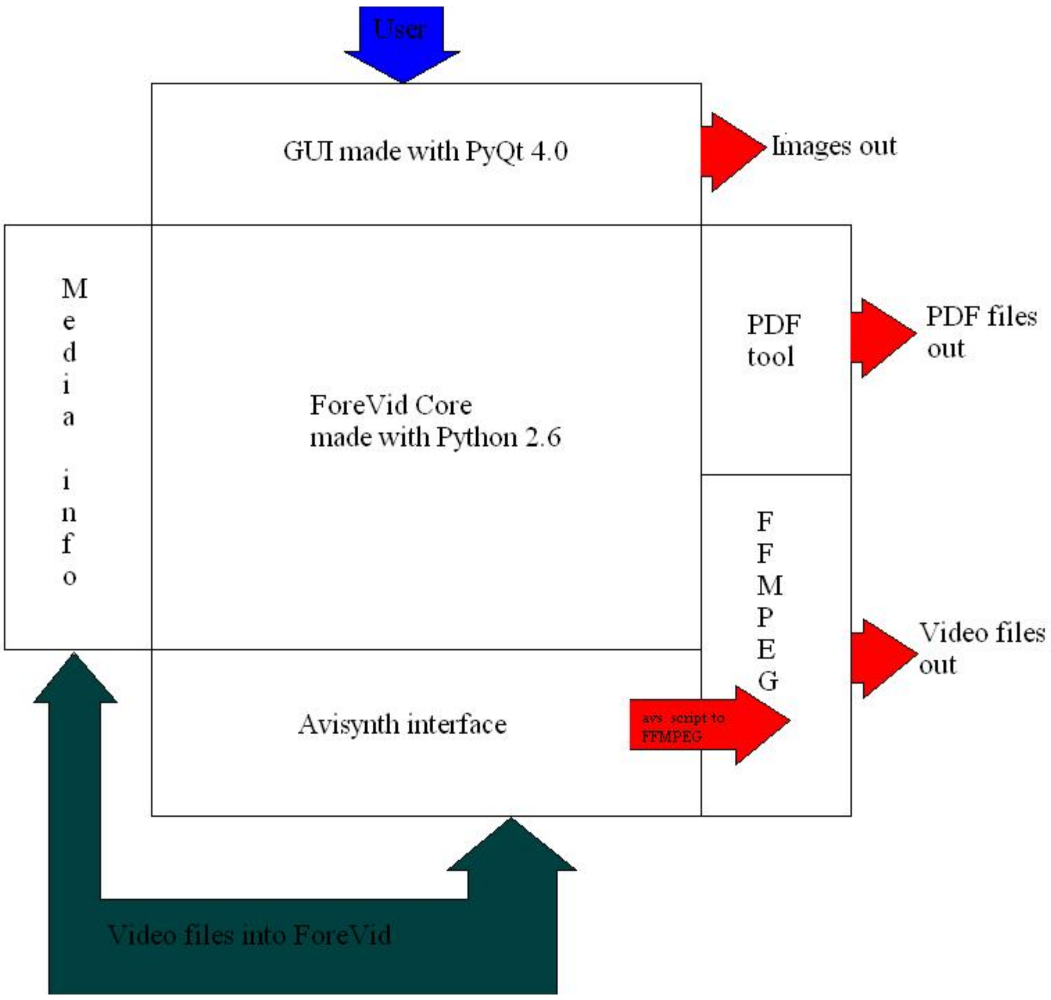


Figure 4.1: Basic blocks of ForeVid.

4.1.1 Python 2.6

Python is a high level programming language which is used as the backbone for programming ForeVid. In this task version 2.6 was used because 3.0 had slightly different kind of command structure and it was not yet as well supported as 2.6, which had much more tutorials and examples. Python installation comes with language editor but the original editor is quite bulky. It is recommend to get a more effective editor like eclipse.[16]

Python is an easy to learn but still very powerful object-orientated language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Its simplicity makes it easy to learn but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle. Python has elegant syntax and dynamic typing, which together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.[16]

4.1.2 PyQt 4.0

PyQt constitutes Python bindings developed by Riverbank Computing Limited for the Qt cross-platform GUI/XML/SQL C++ framework. Qt is developed by Nokia's Qt Development Frameworks, formerly Trolltech framework and runs on all platforms supported by Qt including Windows, MacOS/X and Linux. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets. There are two sets of bindings: PyQt v4 supports Qt v4 and the older PyQt v3 supports Qt v3 and earlier. The bindings are implemented as a set of Python modules and contain over 300 classes and over 6,000 functions and methods. Together PyQt v4 and PyQt v3 support all Qt versions since 1.43 and all Python versions since 2.3. PyQt is dual-licensed under a commercial license and the GNU GPL (version 2 and 3). This means that it can be used freely as long as the product is not commercial. Unlike Qt, PyQt v4 is not available under the LGPL. PyQt4 is a set of Python bindings for Qt 4 that exposes much of the functionality of Qt 4 to Python. It contains over 600 classes that cover graphical user interfaces, XML handling, network communication, SQL databases, Web browsing and other technologies in Qt. With Qt come QTDesigner, that can be used to draw GUI. PyQt is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer. ForeVid GUI was created in text form. When starting to use Python and PyQt, internet is the best place to start. There are lots of tutorial and forums with information for all kinds of situations. Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components. PyQt combines all the advantages of Qt and Python. A programmer has all the power of Qt, but is able to exploit it with the simplicity of Python.

The most used Python modules in ForeVid were QtCore and QtGui. The QtCore module contains the core non-GUI classes, including the event loop and Qt's signal and slot mechanism. It also includes platform independent abstractions for Unicode, threads, mapped files, shared memory, regular expressions, and user and application settings. The QtGui module contains the majority of the GUI classes. These include a number of table, tree and list classes based on the model-view-controller design pattern. Also provided is a sophisticated 2D canvas widget capable of storing thousands of items including ordinary widgets. PyQt4 has Qt module that consolidates classes contained in all of the modules into a single module. This has the advantage that the user doesn't have to worry about which underlying module contains a particular class. It has the disadvantage that it loads the whole of the Qt framework, thereby increasing the memory footprint of an application.[14][15]

4.1.3 FFMPEG

FFMPEG is video compression tool that can create a video file, having Avisynth script as it's input. FFMPEG has many settings for video processing and it offers great amount of flexibility. FFMPEG is a fast video and audio converter. It can also grab from a live audio/video source. In ForeVid it is used for converting videos with changes made in Avisynth. It can take Avisynth script and turn it into new video based on the original video. The command line interface is designed to be intuitive, in the sense that FFMPEG tries to figure out all parameters that can possibly be derived automatically. Usually the only specified target is bitrate. FFMPEG can also convert from any sample rate to any other, and resize video on the fly with a high quality polyphase filter. By default, FFMPEG tries to convert as losslessly as possible: it uses the same audio and video parameters for the outputs as specified for the inputs. FFMPEG has been licensed under the GPL.[13]

4.1.4 The Open Source PDF library

The ReportLab Toolkit is a library for programmatically creating documents in PDF format. It's a robust, flexible, time-proven, industry-strength solution. It fits in ForeVid environment because it is written in Python and it is open-source. It makes possible to quickly and easily create or automate complex or data-driven documents. The ReportLab Toolkit has evolved over the years in direct response to the real-world reporting needs of large institutions. It's in production use across the world as the trusted and proven foundation of existing enterprise solutions.[12]

The library implements three main layers:

- A graphics canvas API allowing to 'draw' PDF pages and also create many of the special features of PDF files (outline entries, links and so on)
- A charts and widgets library for creating reusable data graphics, including many common business and financial charts

- A flexible page layout engine - PLATYPUS ("Page Layout and TYPography Using Scripts") - which builds documents from components like headlines, paragraphs, fonts, tables, images, and vector graphics.

4.2 Functional description

When the ForeVid program is launched, initial screen is displayed to the user for 3 seconds. After that the project menu appears. The project menu is shown in figure 4.2. In project menu the user can create, open or delete projects. Case that is being studied can be created as a project in project menu. Project menu shows current projects that are in the particular working directory. When new project is being created the program creates a blank project that only has its folders and state files created. If the user chooses to open a project, the project's last state is being loaded and the user can continue where he left last time. A Project saves it's state in closing situations so users do not need to save their projects. There is only one saving state so previous states cannot be recovered directly. If a project is deleted it cannot be restored. In the project menu is also an option for setting up a working directory. Working directory is a directory where project folders and files are created. It should be kept in mind that ForeVid doesn't save video files in a project, so removing or deleting an opened video file corrupts the project.

After the project menu closes, ForeVid loads files that have been imported to the project earlier. In this case loading screen is displayed to the user while loading is performed. Loading screen informs the user that the program is loading files and that it is working properly. When all files are loaded to a project, the main working window is displayed.

The main working window is shown in figure 4.3. In the center of the window there is the video displaying area. On the left side is the playlist where all videos in the particular project are listed. Video can be added to a project by dragging it to the playlist or using the file import selection from the dropdown menu at the toolbar. Dropdown menus are located at the upper section of the working window just below the title bar. Below the dropdown menus is the toolbar which includes some quick buttons for frequently used functions. On the right side is a marker list, it contains markers that are made in the project. Marker positions can be changed using buttons below the marker list. Moving cursor on top of a marker brings up the marker text of that marker. Activating marker by a double click selects that marker point as the current video frame. Marker text is shown below the video display area if the marker is activated. This is presented in figure 4.4.

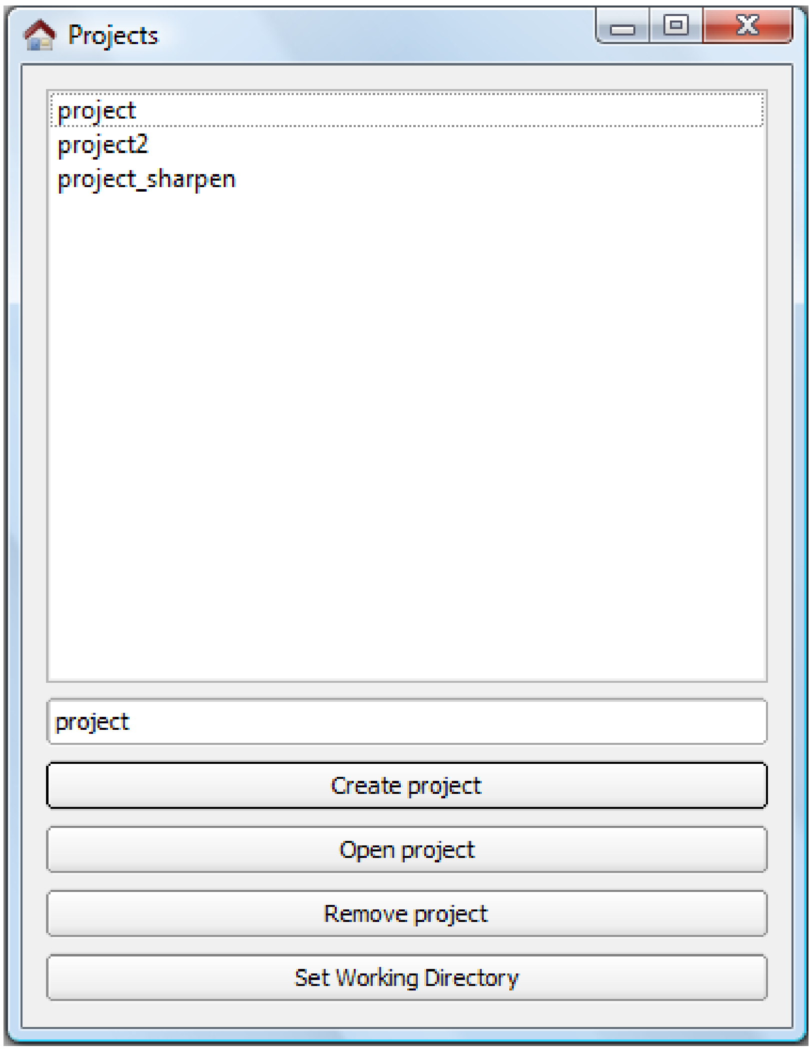


Figure 4.2: The project menu of ForeVid. This is where projects are controlled.

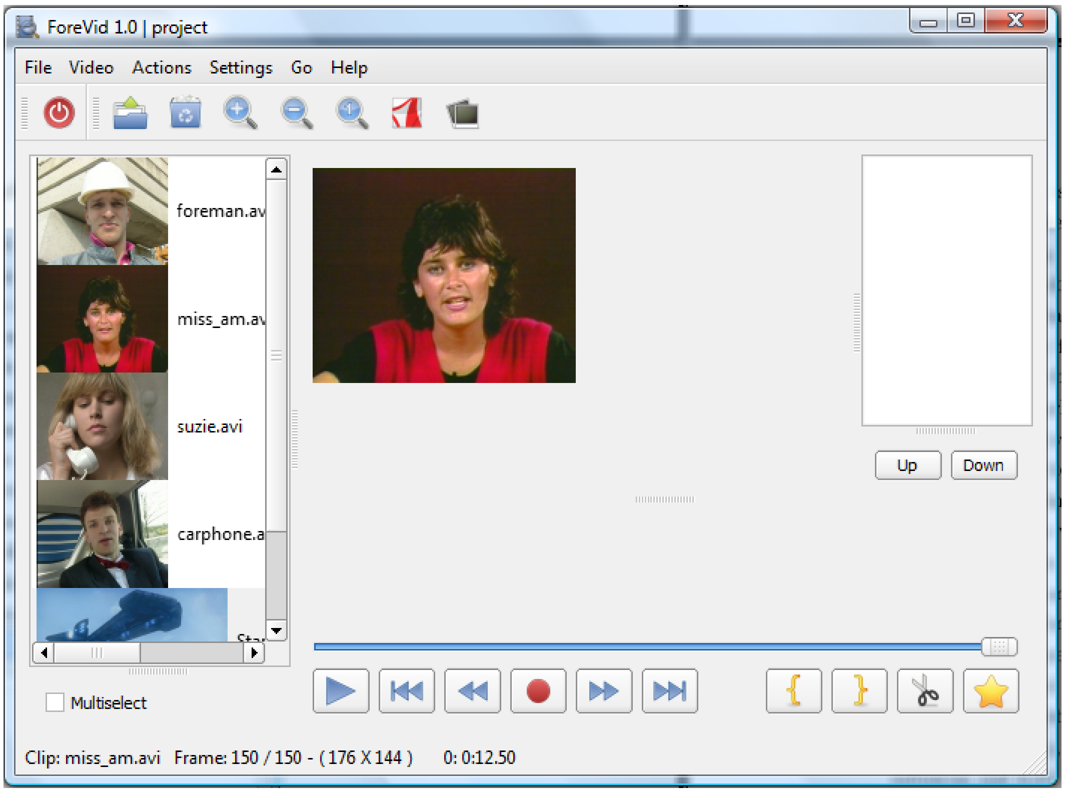

ForeVid 1.0 | project
File Video Actions Settings Go Help
foreman.av
miss_am.av
suzie.avi
carphone.a
Up
Down
Multiselect
Clip: miss_am.avi Frame: 150 / 150 - (176 X 144) 0: 0:12.50

Figure 4.3: Main working window of ForeVid.

Below the marker text area is the video sliding bar, showing current video progress. It can also be used for moving in the video by dragging the slider or clicking on the preferred position. Below the slider is the video control bar where all quick buttons for video watching and time line editing are located. Markers can be seen in figure 4.4. At the bottom of the screen is the status bar which is used for displaying information about the video and process status. There are six menu selections on the menubar: File, Video, Action, Go to, Setup, Help

The file menu holds all video importing and exporting selections. File import asks the user to locate the right file by a graphical interface. The program remembers the folder the last file was picked up from, helping the user if he needs to import more files later. File export is a submenu. In the file export the user can export images from markers, frame sequences or export edited videos or export PDF files from markers. All exports have two options: exporting into fixed project folder or exporting into a folder selected by the user. In the user selected folder option the program remembers it and suggests it at the next time. File menu includes also a selection for returning to the project menu. Current project state is saved if project is changed. There is also exit selection in the file menu.

Video menu includes all selections for processing a video like filter adding and

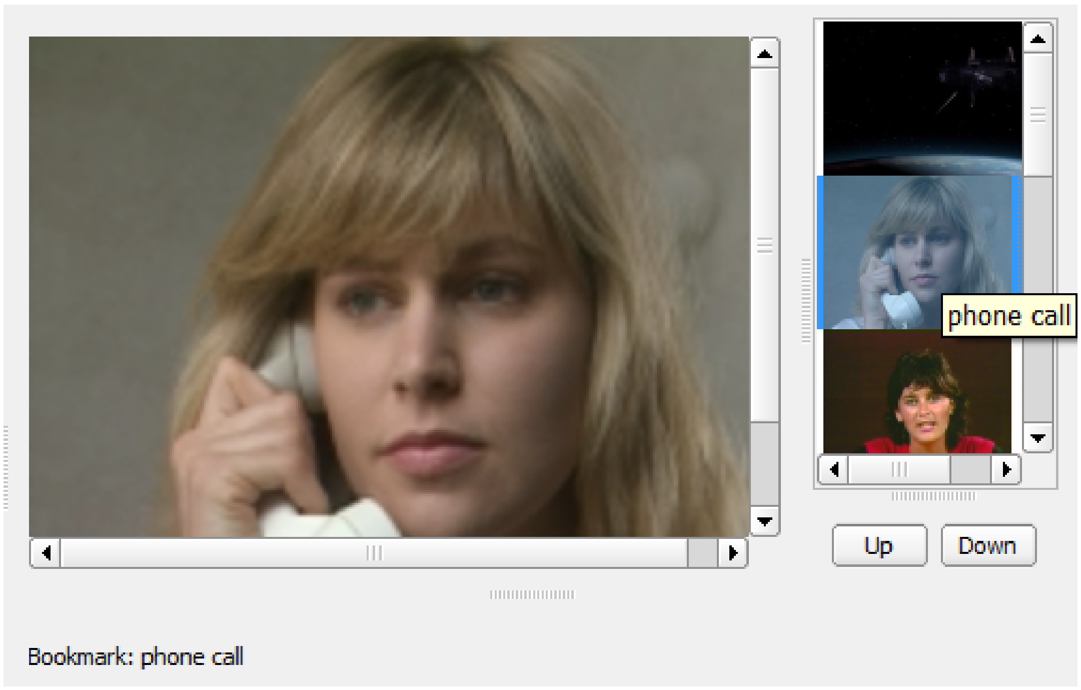


Figure 4.4: The marker list of ForeVid.

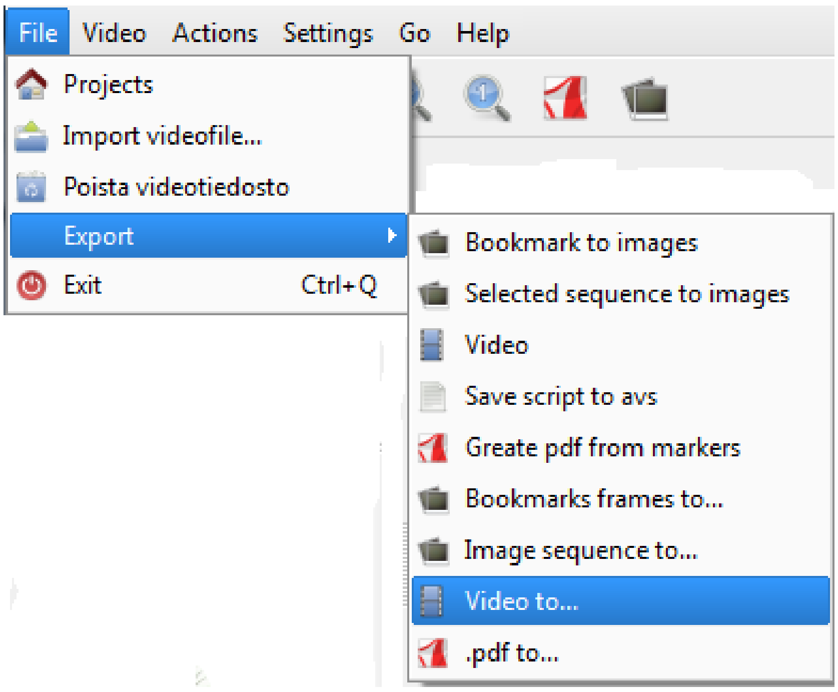


Figure 4.5: Filemenu selections for ForeVid.

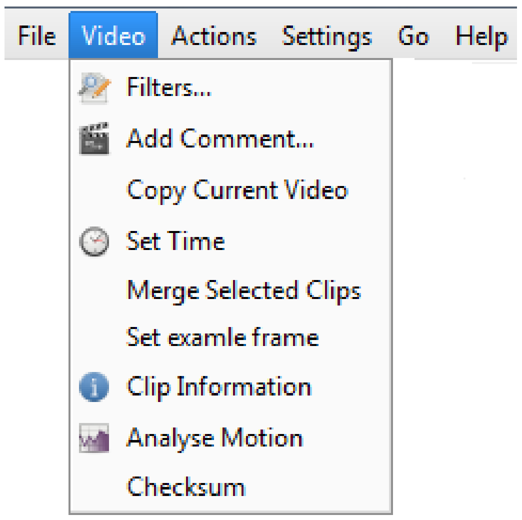


Figure 4.6: The filemenu selections for ForeVid.

removing, motion detecting, header screen, etc. Filter adding is one of the most important functions in ForeVid, it allows Avisynth filters to be applied to the video. Filter removing is also performed in the filter menu. The filter set includes all filters like trim, blankscreen, etc. Filter function brings up a window that has multiple filters which can be selected. When the user selects a filter, a window is created for that filter's attributes. If filter has no attributes, attribute window is not created. More filters can be added to the basic set quite easily by incorporating their source code.

On the toolbar the user can find all quick buttons for functions that are frequently used. The functions on the toolbar are quit, video import, clip deleting, filter adding, zoom out, zoom in and return to original size. The action menu includes all selections that are used for performing an operation to a clip. There are also some time line editing and marker managing functions. These menus can be seen in figure 4.3.

Goto menu is used for moving within the clip. There are functions for playing the video or a whole video list. There are also multiple selections for moving to different locations within the clip. Setup handles some parameters that can be changed. User can select which mediafilter is used for opening videos. Saving format for the image as well as the language of the environment can also be selected here. Currently available languages are English and Finnish. ForeVid is built so that further adding of languages is easy.

Current version of the program can be found in the help menu. When a video is added to the project, the program checks if the video can be opened. If the file does not open, the program informs the user about this and shows the info window containing additional information about that file. If a file is opened successfully, a video icon appears in the play list and the imported video becomes active. When a video is activated it's first frame is shown in the display area. Only one copy of a video can be opened in the project, but the video can be copied inside ForeVid for duplicates for editing purposes.

4.3 Introduction to the source code

ForeVid has thousands of lines of code. In the following a fast overview is offered to clarify the coding process of the ForeVid. Like mentioned before, ForeVid uses video processing tools of the Avisynth to open and edit the video. First thing to code is the plugin for the Avisynth. For this "AvsClip" class was created. AvsClip uses "avisynth"-class which was modified from the AvsP program[17]. In "avisynth" class avidll=ctypes.windll.Avisynth is defined creating interface to the DLL-file. These classes create binding between Avisynth and ForeVid. For controlling data stream from Avisynth, "AvsClip" class has C type structures and functions. An example of creating a C type structure is given below:

```
class BITMAPFILEHEADER(ctypes.Structure):
    _fields_ = [
                ("bfType", WORD),
                ("bfSize", DWORD),
                ("bfReserved1", WORD),
                ("bfReserved2", WORD),
                ("bfOffBits", DWORD)]
```

An example of creating a C type function looks like:

```
CreateFile = ctypes.windll.kernel32.CreateFileA
WriteFile = ctypes.windll.kernel32.WriteFile
CloseHandle = ctypes.windll.kernel32.CloseHandle
```

"AvsClip" class also contains following information: filter list, filter values, marker list, name of the clip, zoom values and number of the current frame. Information that "AvsClip" class has is used to identify the clip in ForeVid. All this information is needed to allow multiple clips in a session. The most used function in the "AvsClip" class is the "GetFrame function". This function asks a frame from Avisynth by using a frame number. Frame information is given as C-type data. Another important function is "addFilter" and it uses "Invoke" to apply filters to an opened clip. The function is used as:

```
addFilter(Avisynth command, Avisynth clip, parameter1, parameter2,...,parameter6)
```

In "addFilter" Invoke is used as follows:

```
avsfile = self.env.Invoke(filter,args,0)
arg0.Release() #release the clip
self.clip = avsfile.AsClip(self.env)
```

In here "filter" contains Avisynth filter command and args is a vector containing all the parameters. In the main program of ForeVid the video clip is opened as follows:

```
self.AVI = omaAvi.AvsClip(fileName, ",fitHeight=None, fitWidth=None, oldFrame-
count=1, keepRaw=True)
```

"omaAvi" is the file where "AvsClip" class is presented. "fileName" is a char type value containing file path and name. Another problem was to get the frame information in Python compatible form. Solution can be found from the "drawFrame" function of the main program. In the code there are lines for error checking and setting up the right clip. Finally in the seventh line the _GetFrame-function is called. The function activates the frame that was called and it can be pointed in the computer memory. Using the pointer data is converted from C-type data to Python compatible form. Image information comes out as mirrored so " mirrored()" function of "QImage" class is used to correct the image. Finally "vaihda"-function of the "videoWindow" is used to change the current frame shown. "videoWindow" is a widget object created for showing the video frame. Rest of the "drawFrame" function is setting up GUI parameters and an option for showing original clip and/or histogram. The "drawFrame" function is shown below:

```
def drawFrame(self,frame):
    if len(self.creatureList) > 0:
        self.AVI = self.creatureList[str(self.lastVideo)]
        self.pituus = self.AVI.Framecount
        self.slider.setRange(0, self.pituus)
        self.viive = 1/self.AVI.Framerate
        self.AVI._GetFrame(frame)
```

```
a = ctypes.string_at(ctypes.pointer(self.AVI.pBits),
self.AVI.Width*self.AVI.Height*self.kanavia)
B =
QImage(a,self.AVI.Width,self.AVI.Height,QImage.Format_ARGB32)
B = B.mirrored()
B = B.scaledToHeight(self.AVI.Height*self.AVI.getZoom()[0],1)
self.videoWindow.vaihda(B)
self.statusBar().showMessage(self.dict.getWord('Clip: ')+'%s
'%(self.lastVideo)+self.dict.getWord('Frame:')+' %.0f / %.0f -
(%.0f X %.0f ) %s'%(frame,self.pituus,self.AVI.Width,
self.AVI.Height,self.time(self.AVI.getTime()[0],
self.AVI.getTime()[1], self.AVI.getTime()[2])))
self.checkMarker()
if self.showOrig:
   if self.isChanged:
      self.showOriginal()
      self.orginalAVI._GetFrame(frame)
      a2 =
      ctypes.string_at(ctypes.pointer(self.orginalAVI.pBits),
      self.orginalAVI.Width*self.orginalAVI.Height*self.kanavia)
      B2 =
      QImage(a2,self.orginalAVI.Width,self.orginalAVI.Height,
      QImage.Format_ARGB32)
      B2 = B2.mirrored()
      B2 =
      B2.scaledToHeight(self.orginalAVI.Height*self.AVI.getZoom()[0],1)
      self.videoWindow2.vaihda(B2)

if self.showHisto:
   B.save(QString("temp\hist.bmp"),"BMP",70)
   self.hist.refresh()
```

PDF files are created by the following method. Function "makePdf" is called to create a PDF file. "makePdf" uses class named "ForevidPdf". The function creates a PDF file and inserts marked frames as images with marker text into to the file one by one. The image is obtained from the "AvsClip" class as presented earlier. "self.markerL" is a list containing name of the clip, number of the marked frame and marker text. The "ForevidPdf" class has a function "addFigure" that inserts an image and text into the PDF file. the "build()" function builds up the actual file to be saved to the hard drive. Value of the "self.lastVideo" is stored in the help parameter so that current situation can be restored after PDF file creation. The "makePdf" function is shown below:

```
def makePdf(self):
    if len(self.markerL)>0:
      if len(self.folder) < 1 and not self.toFolder:
        directory = self.path[0]+self.project[0]+'\\docs\\'
      else:
            if len(self.folder)/<4:
              directory = self.folder[:2]
            else:
                  directory = self.folder+'\\'
                  pdf = ForevidPdf(unicode(directory)+
                  unicode(self.project[0])+".pdf",str(self.project[0]))
                  help = self.lastVideo
      for k in self.markerL:
          self.lastVideo = k[1]
          self.frame = k[0]
          self.AVI = self.creatureList[self.lastVideo]
          self.AVI._GetFrame(self.frame)
          a =types.string_at(ctypes.pointer(self.AVI.pBits),
          self.AVI.Width*self.AVI.Height*self.kanavia)
          B =
          QImage(a,self.AVI.Width,self.AVI.Height,QImage.Format_ARGB32)
          B = B.mirrored()
          pdf.addFigure(B,unicode(k[2]))
          self.lastVideo = help

      pdf.build()
      self.statusBar().showMessage(self.dict.getWord('Created:
      ')+unicode(self.project[0])+".pdf")
    else:
          reply = QMessageBox.warning(self, 'Warning',
          self.dict.getWord("There aren't any bookmarks"), QMessageBox.Ok)
          self.toFolder = False
```

One of the main goals of the software is to create video files. For this task a working video compression library is needed. FFMPEG compression tool offers solution to this problem. FFMPEG has been presented in earlier sections. In the following it is shown how it was implanted into ForeVid. FFMPEG takes parameters from the command line. First task was to figure out how commands in FFMPEG work and what kind of parameters it can have. Next step was to find out how Python can handle command line streams. In ForeVid a "aviSave" class operates as Python wrapper for FFMPEG. "aviSave" has fixed profiles that have been presented earlier in video compression section. These profiles are read from an XML file. The main commands of the "aviSave" class are:

```
self.process = QProcess()
self.process.setProcessChannelMode(QProcess.MergedChannels)
...

def startCommand(self):
    ...
    command = 'ffmpeg\\ffmpeg -i "'+self.avsFile+'" -r '+str(self.fps)
    + ' -f ' + format + parameters + ' -y"'
    +self.path+self.fnameEdit.text()+'"'
    self.process.start(command)
    ...
```

It is desirable to get information about the saving process, so the output stream of standard output is printed. "SIGNALS" function as triggers that are waiting for the signal they are given. Key commands of the function are presented below:

```
self.connect(self.process, SIGNAL("readyReadStandardOutput()"), self.readOutput)
self.connect(self.process, SIGNAL("readyReadStandardError()"), self.readOutput)


def readOutput(self):
    self.output.append(str(self.process.readAllStandardOutput()))
```

4.4 Example cases solved by ForeVid

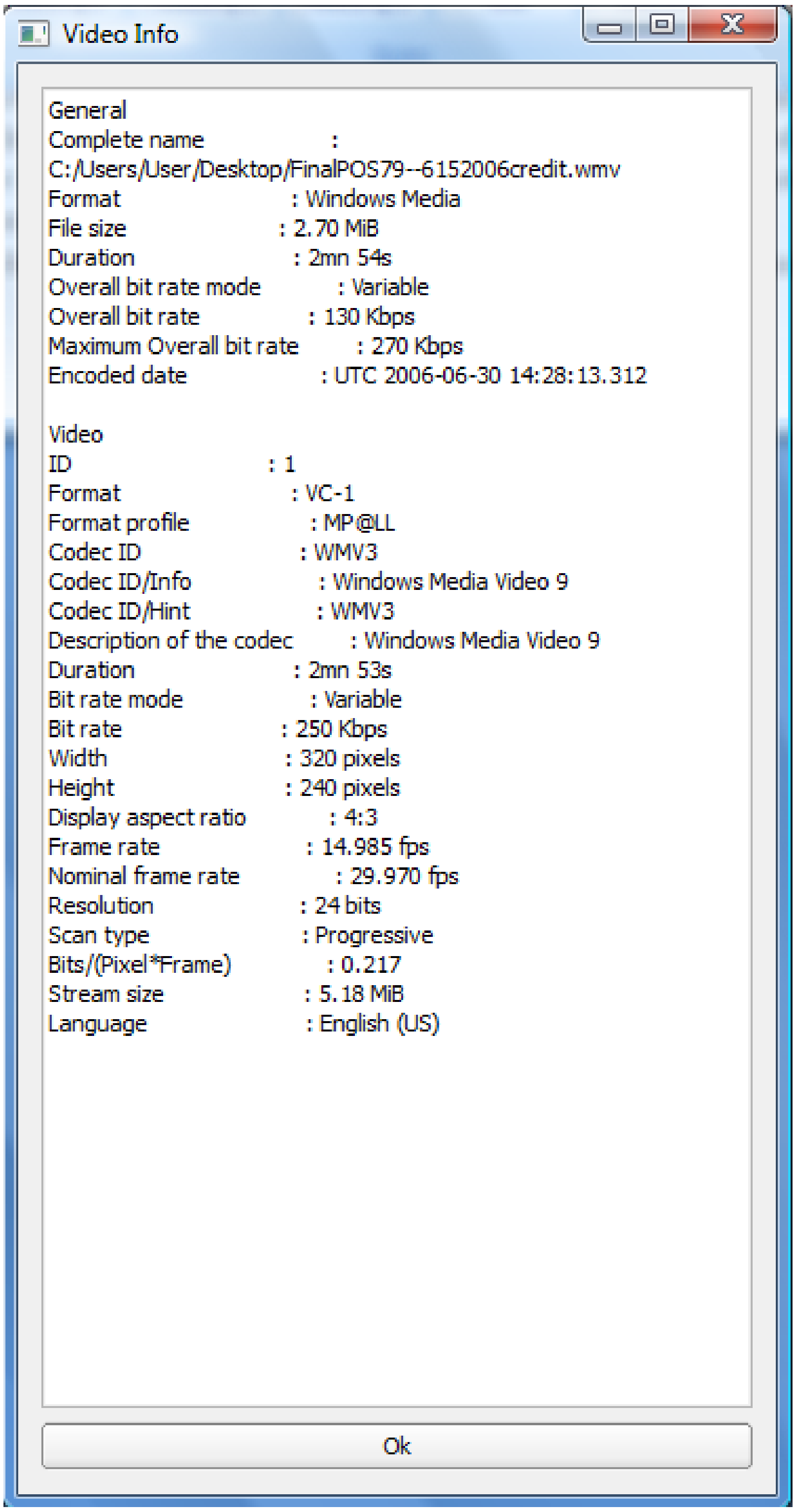
In this section we introduce some of the features and abilities of ForeVid. The main goal of this section is to show basic usage of ForeVid, not making any fancy tricks. Two example cases are presented: the first one is from a store where two suspects are using stolen credit card, in the second one a video clip from attempted ATM robbery is shown.

CASE1: Surveillance video shows a couple using stolen credit card.

Background: The credit card was reported stolen from a parked truck. The victim told deputies that her wallet and credit cards and about 20 DVDs were stolen from her truck. Before she was able to cancel all of her credit cards, some were used at businesses in Orange City, Sanford and Lake Mary. Shortly after the break-in, a white woman who appeared to be in her late teens was captured by a surveillance camera in Target purchasing merchandise with one of the stolen credit cards. She had red hair and was wearing blue jeans and a green shirt. She was accompanied by a man of Hispanic or Middle Eastern descent who had short, dark hair and was wearing blue jeans and a black shirt.[18]

Processing: first ForeVid is started like presented before. When the project screen appears, a name for this project (case number, for example) is given. In this case the name is "Appendix A" for obvious reasons. As can be remembered a case can contain multiple videos, however for simplicity this particular example has only one video. When the project is set, it is time to import the video clip. "File=>Import video file" function or quick icon in the toolbar can be used. In each case the file selection window is presented. When video file is imported, an icon should appear in the video clip list. However, in this case an error box is presented saying: "Could not open file as video!". Video information box from that file is presented. Video information box is shown in figure 4.7. The video information box gives all the information from the file that was opened. From this particular file we can see that it is in fact a video clip. The reason that it didn't open is that the media filter in use didn't support this clip. Avisource filter is the default filter in use, but in this case we need to change it to DirectShowSource from: "Settings=>Set opening filter for clip". Now we can open this video clip and icon appears in the video clip list.

First we take a look at the clip to see what is going on in it. It shows the scene described earlier. Lights in the clip are good so there is no point in changing them. Resolution of the video clip is small (320x240) as typical. For this I double the resolution using Lanczo4Resize filter. Now I have clip that has resolution of 640x480. The image looks blurry, but image quality is almost as good as it gets. In this example the value used for sharpen was 0.5, but like said before, the changes were not significant. Next step is to take a couple of still images from the suspects and take a PDF file describing what is happening. One of the still images is presented in figure 4.8. Regarding the still shots, the fast way is to use the still shot button (red big dot) in the video control bar. Second way is to use "File=>Export"



```
General
Complete name                :
C:/Users/User/Desktop/FinalPOS79--6152006credit.wmv
Format                       : Windows Media
File size                    : 2.70 MiB
Duration                     : 2mn 54s
Overall bit rate mode        : Variable
Overall bit rate             : 130 Kbps
Maximum Overall bit rate     : 270 Kbps
Encoded date                 : UTC 2006-06-30 14:28:13.312

Video
ID                           : 1
Format                       : VC-1
Format profile               : MP@LL
Codec ID                     : WMV3
Codec ID/Info                : Windows Media Video 9
Codec ID/Hint                : WMV3
Description of the codec     : Windows Media Video 9
Duration                     : 2mn 53s
Bit rate mode                : Variable
Bit rate                     : 250 Kbps
Width                        : 320 pixels
Height                       : 240 pixels
Display aspect ratio         : 4:3
Frame rate                   : 14.985 fps
Nominal frame rate           : 29.970 fps
Resolution                   : 24 bits
Scan type                    : Progressive
Bits/(Pixel*Frame)           : 0.217
Stream size                  : 5.18 MiB
Language                     : English (US)
```

Figure 4.7: When ForeVid cannot open a file or the media filter in use is wrong video information box is presented with error message. Video information box can also be seen from video drop box by selecting video information.

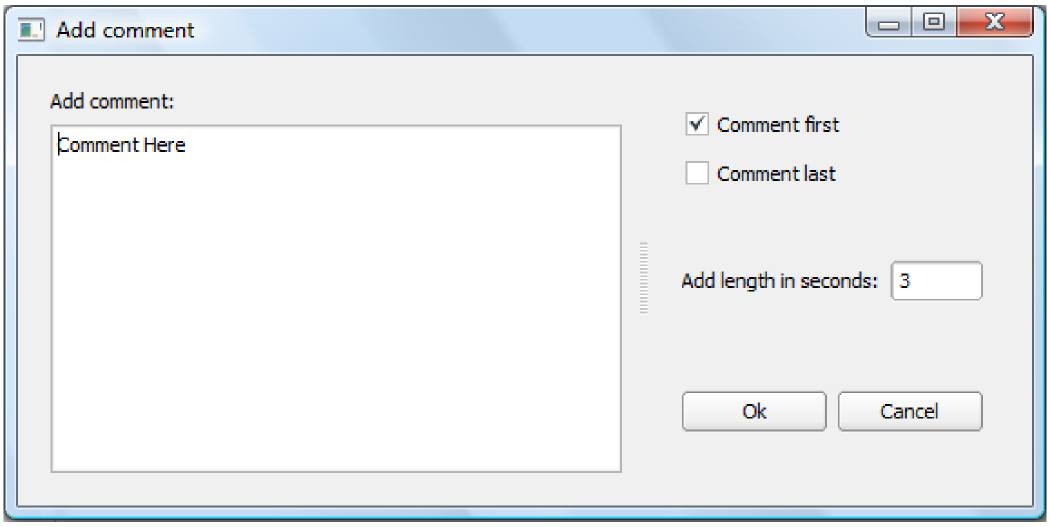
Figure 4.8: One of the main jobs in forensic usage of ForeVid is to take still images. That's why it has been made as easy as possible.

and select to take images from specific frame sequence or turn markers into images. PDF file is created using markers. Comment of the marker is placed as image text. Of course it is possible to change comment later from "Actions=> Edit bookmark". When markers are all set and commented right we can start to create the PDF file. It can be created by using the toolbar or "File=>Export" sequence. In all options it is possible to determine where the file is saved. In this example I use the toolbar option. Created PDF file can bee seen in appendix A.

CASE2: Video clip from attempted ATM robbery

Background: This video shows two suspects trying to rob ATM inside a store [18]. The video includes video footage from two different cameras. First camera is recording inside of the store and the second one outside the store. Footage quality is not the best possible, but that is the real world situation.

Processing: In this case it is presented how to create new clip. Because there are two clips merged together, they need to be separated. First use "copy current video" function from the video menu. Copy function was created because ForeVid



Add comment:

Comment Here

[✓] Comment first

[ ] Comment last

Add length in seconds: 3

Figure 4.9: When comment is added, the user can determine for how long the comment screen is being shown and where it is being positioned.

prevents user from opening same file multiple times. After this there are two clips, the original and the copy. Next thing to do is to cut out unwanted material. In the first clip which is the original one, cut out the footage outside the store. In the copy do the opposite. After cutting process there are two clips, one outside of the store and one inside of the store. Next both videos are to be resized into bigger size. In this case size is doubled just like in case 1. For outside footage using "levels" filter for light adjustment gives better result. When using "levels" the user can adjust "input high" and "input low" attributes, same goes for "output low" and "output high" attributes. These are best used in situations where increasing light level make lighter spots too light. The user can determine so that only light in darker spots is increased. Ihe next step is to add comment screen in the beginning of the clip, between the clips and in the end. Comment can be added from "Video=>Add comment". Add comment window is then opened and the user can select where the comment is positioned: in beginning of the clip or in the end of the clip. Add comment window is presented in figure 4.9. First we make comment in beginning of the clip. The first text "ATM robbery" is typed and below that text "Inside the store" then "beginning of the clip" selection is set on and second setting is left for 3 seconds. Now we get 3 second clip in front of the original clip of saying:

ATM robbery
Inside the store

This message is being shown for 3 seconds just as was specified. One frame from the comment screen is being presented in figure 4.10. Text size is fixed so text appears smaller if the resolution is higher. When this is done it is time to do the next clip

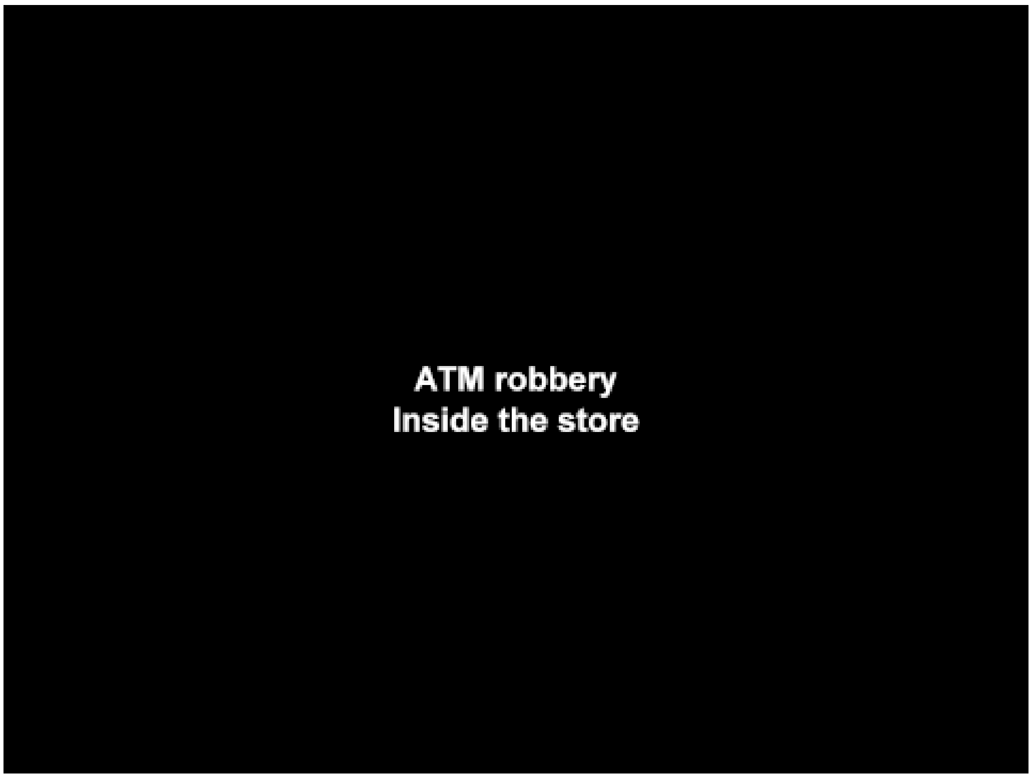
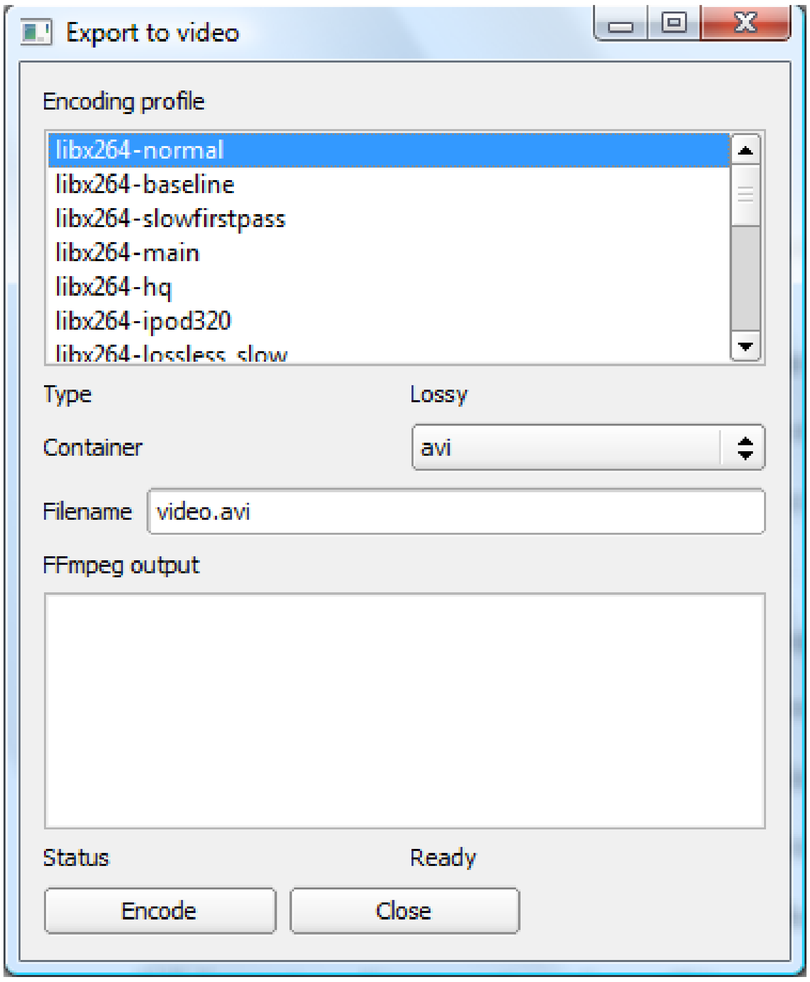


Figure 4.10: Comment frame can give information for the viewer.

from outside the store. This works like in the first clip and the following text is added in the beginning of the clip:

ATM robbery
Outside the store

Now two clips with comment frames have been added in front of the video. Next thing is to merge those two clips back together. This is done by selecting "Video=>Merge selected clips". After this an icon of the new clip appears in the video list. Next ending comment is placed in end of this new clip. This new clip is not real file on the hard drive it is simply a script binding those clips involved together. In last part the video is saved so it becomes something real. This can be done from "File=>Export". Selections for video saving are saving it directly to the project folder or asking the saving folder from the user. Saving window is presented in figure 4.11. Simplified version of the video created in this project can be seen in figure 4.12.



Export to video

Encoding profile

libx264-normal
libx264-baseline
libx264-slowfirstpass
libx264-main
libx264-hq
libx264-ipod320
libx264-lossless_slow

Type Lossy

Container avi

Filename video.avi

FFmpeg output

Status Ready

Figure 4.11: In the video saving window the user can select the profile to be used.

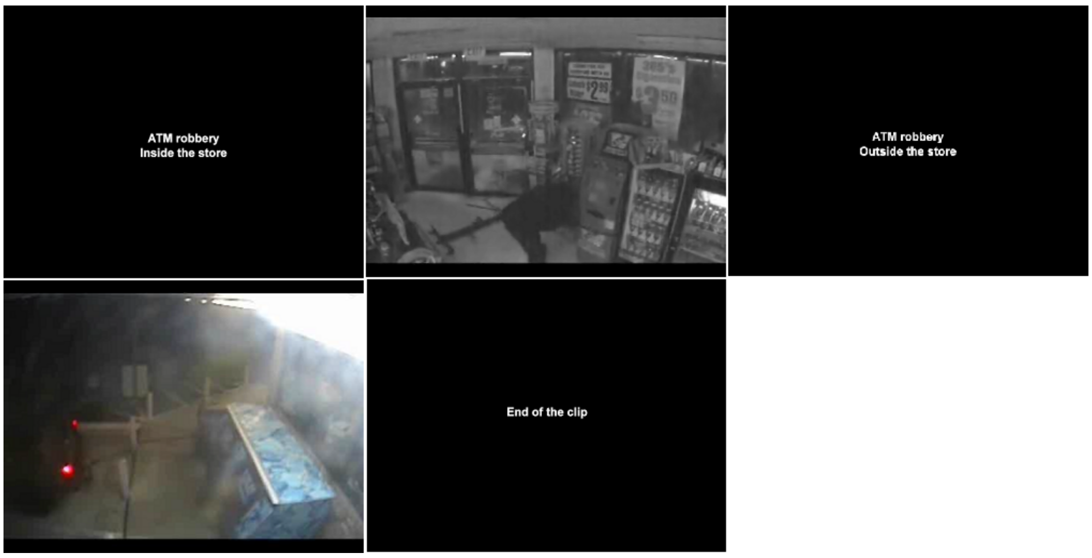


Figure 4.12: Video created in case 2 is presented here. Every frame presents one part of the clip. Reading order is from left to right.

5. CONCLUSIONS AND FUTURE WORK

Development of the surveillance technology has been exponential. Surveillance technology has become more common all around the world continuously, because the amount of video material to handle grows rapidly. That makes video material more and more important in forensic investigations. Typically video data is processed in the same way as images, frame by frame. However, time dependence of the frames in a video offers additional possibilities for video data processing. One can, for example, improve video quality and decrease the noise level for more detailed images. If we can recognize object movement in the video we can try to remove shaking camera effect or even use more advanced operations like super-resolution.

Although there are commercial software available for the task, no free, open-source alternatives are available. This is the main reason of making this thesis and ForeVid, a free, open-source software for the forensic analysis of surveillance videos. Similarly as its commercial counterparts, Forevid contains the necessary features for forensic analysis of surveillance videos, including several options for video playback, processing and documentation. In a typical law enforcement organization, the high licensing costs of commercial software can considerably limit the amount of people having access to a proper analysis software. As the amount of surveillance video material is constantly increasing and licensing costs limiting the amount of people working with professional software, the reliable analysis of video material may become challenging. However, with Forevid these restrictions do not apply. The organization can save money in license fees, and concurrently increase the amount of people working with surveillance video.

NBI of Finland continues development of ForeVid. This thesis handles ForeVid version 1.0 and even when writing this version 1.01 is on it's way. Reception for ForeVid has been good and the software's future looks good. Software is now used in many police station in the country. Feedback from these station give valuable information for future development of the ForeVid. GUI will probably be changed according to the feedback that is given by the users. Also new features will be added as need occurs. Here are many interesting possibilities for the future development of Forevid. First, the selection of available video processing operations can be extended. With more efficient processing operations, the user has more options to enhance low quality video material. Second, the potential of automated video content analysis could be explored. Although automated analysis could not be used for producing forensic evidence, it would give us valuable tools, when reviewing large amounts of surveillance video material.

Source code of the software is freely available. Hopefully there is interest for this kind of work and development of this software continues.

BIBLIOGRAPHY

[1] Arun Hampapur, Lisa Brown, Jonathan Connell, Sharat Pankanti, Andrew Senior, Yingli Tian, Smart surveillance: applications, technologies and implications, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.8004&rep=rep1&type=pdf, May 2010

[2] Gregory K. Wallace, The JPEG Still Picture Compression Standard, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.226 &rep=rep1&type=pdf, May 2010

[3] Athanassios Skodras, Charilaos Christopoulos, Tourradj Ebrahimi, The JPEG 2000 Still Image Compression Standard, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.183 &rep=rep1&type=pdf, May 2010

[4] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard, Ajay Luthra, Overview of the H. 264/AVC video coding standard, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.4000&rep=rep1&type=pdf

[5] Avisynth, URL: http://Avisynth.org/mediawiki/Internalfilters, May 2010

[6] EE Times:Video surveillance rides IP networks, URL: http://www.eetimes.com/showArticle.jhtml?articleID=198500214, July 2010

[7] Video-Surveillance-Guide ,URL: http://www.video-surveillance-guide.com/, July 2010

[8] Ben Rudiak-Gould et al. Avisynth, URL: http://Avisynth.org/, June 2010

[9] Fabrice Bellard et al. FFmpeg, URL: http://www.ffmpeg.org/, July 2010

[10] Laurent Aimar et al. x264 a free h264/avc encoder, URL: http://www.videolan.org/developers/x264.html, May 2010

[11] IP Video Surveillance, URL: http://www.preferredtechnology.com/solutions/videosurveillance/videotransport.html, June 2010

[12] ReportLab's Open Source Libraries, URL: http://www.reportlab.com/software/opensource/, December 2010

[13] FFMPEG, URL: http://www.ffmpeg.org/, December 2010

[14] PyQt-Python Info, URL: http://wiki.python.org/moin/PyQt, December 2010

[15] Riverbank PyQt, URL:
http://www.riverbankcomputing.co.uk/software/pyqt/intro, December 2010

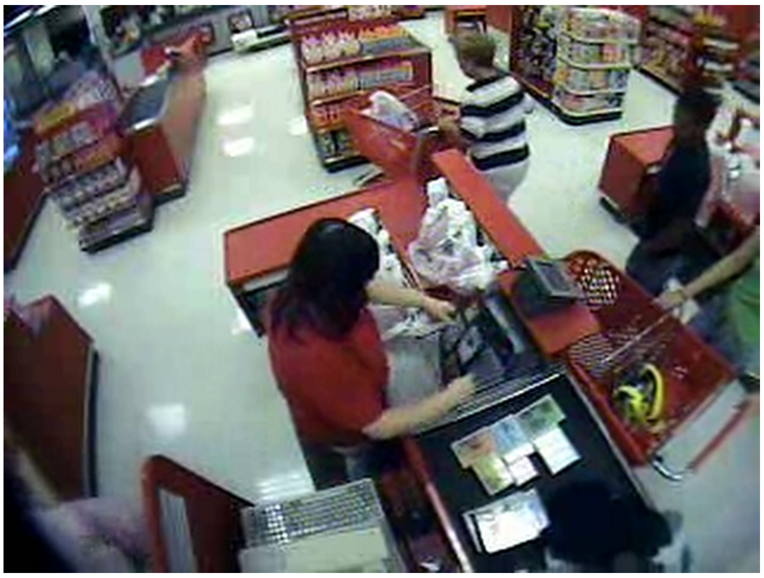
[16] Python Programming Language Official Website, URL:
http://www.python.org/, June 2010

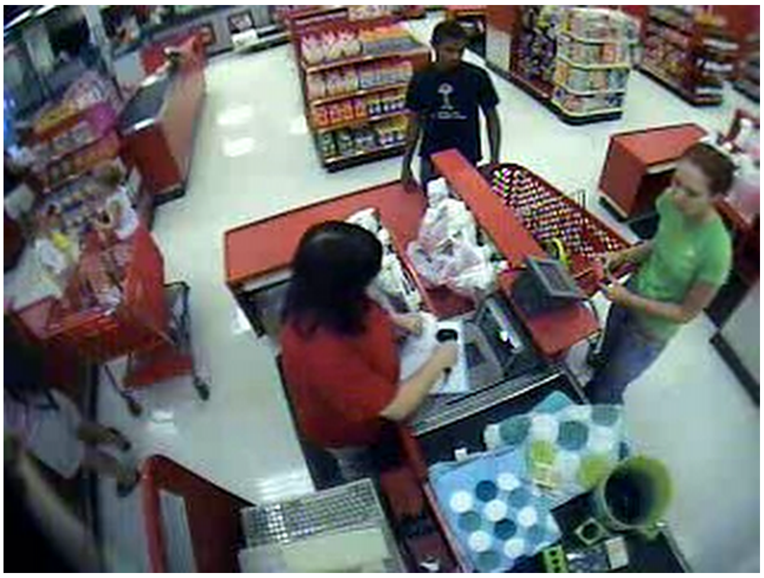
[17] AvsP; Avisynth scrip tool, URL: http://Avisynth.org/qwerpoi/index.html, December 2010

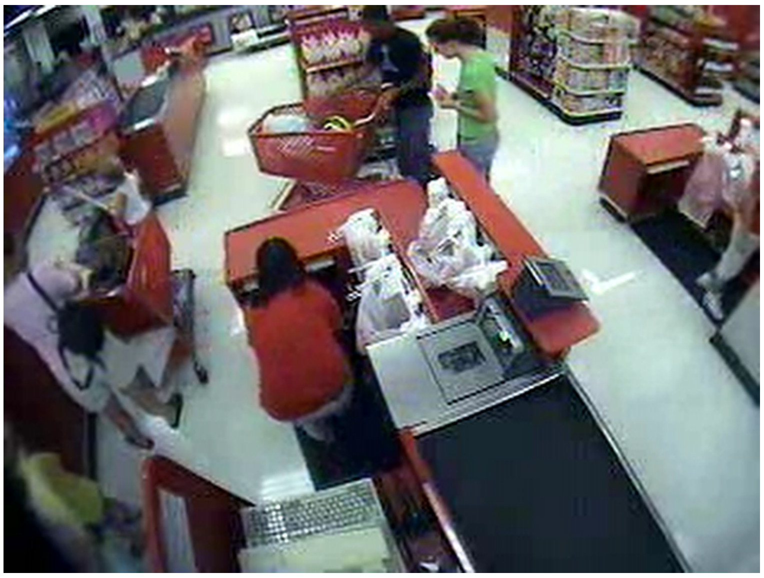
[18] Volusia County sheriff's office; Video Download Page, URL:
http://www.volusia.org/sheriff/press/Video%20Downloads/default.htm, Janyary 2011

[19] Image, Biometric Facial Recognition, URL:
http://mbpgsu.ca/category/science/, April 2010

Appendix A


Suspects arrive at the cash desk with purchases


Suspects pay for the purchases with stolen credit card.



Suspects leave the store.