SUSPICIOUS ACTIVITY DETECTION

Computer Science
Honours Documentation 2010
Author: Dane Brown
Student Number: 2713985

Supervisors: Mr James Connan and Mr Mehrdad Ghaziasgar

Department of Computer Science
University of the Western Cape

A mini-thesis submitted in partial fulfilment of the requirements for the degree of
B.Sc. Honours.
ABSTRACT

The project serves to reduce the crime rate in South Africa. It aims to increase the efficiency of security guards performing surveillance. The project alerts the user of suspicious activity in multiple parking lots. The project itself is scalable enough to incorporate other functionality with relative ease. It logs the suspicious activity for future reference. It operates in real-time and has an effective accuracy of greater than 90% for detecting positive suspicious activity when set up in a specific parking lot. The final decision to decide whether an offence has been committed is still up to the user.
PLAGIARISM DECLARATION

I, Dane Brown, certify that this project is my own work. I understand what plagiarism is and I have used quotations and references to fully acknowledge all the words and ideas of others, which we have used in our project. I have not copied anyone else's project. I have also not permitted anyone to copy my project.

Signature: _ _Dane Brown_ _
ACKNOWLEDGEMENTS

First and foremost I am ever grateful to my Lord and Saviour, Jesus Christ to whom I owe my life. Without him I would not have been able to produce this honours project or even be for that matter. I would also like to thank my parents for giving me the opportunity to study at university. Next I would like to thank my two classmates that greatly influenced my performance in this year in honours, Mr Simon Kleinsmith and Mr Stafford Joemat. I would like to thank Telkom for granting me a bursary in undergraduate level to enable me to pay for my studies and the University of the Western Cape.

I am wholeheartedly grateful to my supervisors Mr James Connan for guiding me to reach my initial milestones in the first semester and Mr Mehrdad Ghaziasgar for enabling me to create a feasible quality product at the end of this honours year. To Mrs Verna Connan and Miss Fatima Jacobs, thank you for your support with admin work this year. I would also like the thank Daniel Leenderts for providing the necessary equipment for our lab and the rest of the computer science staff for putting in all the effort to prepare us at undergraduate level to enable us to excel at postgraduate level.
TABLE OF CONTENTS

ABSTRACT ......................................................................................................................... 2
PLAGIARISM DECLARATION ............................................................................................... 3
ACKNOWLEDGEMENTS .................................................................................................... 4
CHAPTER 1 ......................................................................................................................... 7
  INTRODUCTION ............................................................................................................. 7
    1.1 Computer Vision and Image Processing ............................................................... 7
    1.2 Current Research .................................................................................................... 7
CHAPTER 2 ......................................................................................................................... 9
  USER REQUIREMENTS DOCUMENT ............................................................................ 9
    2.1 Users view of the problem ..................................................................................... 9
    2.2 Description of the problem .................................................................................. 9
    2.3 Expectations from the software solution ............................................................. 11
    2.4 Not expected from the software solution ............................................................ 12
CHAPTER 3 ......................................................................................................................... 13
  REQUIREMENTS ANALYSIS DOCUMENT ................................................................... 13
    3.1 Designer's interpretation ...................................................................................... 13
    3.2 Breakdown of the problem .................................................................................. 14
    3.3 Complete Analysis of the problem ...................................................................... 14
    3.4 Current Solution .................................................................................................. 17
    3.5 Suggested Solution ............................................................................................. 17
CHAPTER 4 ......................................................................................................................... 18
  USER INTERFACE SPECIFICATION ............................................................................ 18
    4.1 Start Screen .......................................................................................................... 18
    4.2 How the user interface behaves .......................................................................... 19
CHAPTER 5 .............................................................................................................................................. 22

HIGH LEVEL DESIGN ..................................................................................................................... 22

5.1 High Level Data Dictionary ................................................................................................. 22

5.2.1 Relationship between objects .......................................................................................... 24

5.2.2 Full Solution Component Diagram .................................................................................. 25

5.2.3 Tracking of Cars and Humans diagram .......................................................................... 26

CHAPTER 6 .............................................................................................................................................. 27

LOW LEVEL DESIGN ...................................................................................................................... 27

6.1 Low Level Data Dictionary .................................................................................................. 27

6.2 Why not machine learning classification ............................................................................. 28

6.3 Detailed Methodology ......................................................................................................... 29

6.4 State Diagram ....................................................................................................................... 34

CHAPTER 7 .............................................................................................................................................. 35

TESTING .............................................................................................................................................. 35

7.1 Stress Testing ......................................................................................................................... 35

7.2 Correctness Testing .............................................................................................................. 36

CHAPTER 8 .............................................................................................................................................. 37

USER MANUAL ..................................................................................................................................... 37

8.1 Starting up the system ............................................................................................................ 37

8.2 Using the system in demonstration mode .............................................................................. 41

CHAPTER 9 .............................................................................................................................................. 46

CODE DOCUMENTATION ............................................................................................................. 46

CHAPTER 10 ......................................................................................................................................... 47

CONCLUSION ...................................................................................................................................... 47

REFERENCES ...................................................................................................................................... 48
CHAPTER 1

INTRODUCTION

1.1 Computer Vision and Image Processing

Computer Vision is used in various ways to manipulate images. It is used as a medium for the computer system to communicate with images. Information can be extracted from an image in order to be used for a specific function. Vehicles and people should be extracted separately in order to monitor their specific behaviour. Human behaviour is particularly emphasized. OpenCV is the open source library used for this purpose. Its main focus is real-time image processing. The following research will be based on the C/C++ language using the OpenCV library. [1]

1.2 Current Research

Security has always been a big issue in a crime infested country like South Africa (RSA). [2] Security measures such as police recruiting and training are in place in many areas, but have not always proven to be particularly effective or useful. A good starting point to combat crime is the usage of computer vision techniques to promote the successful detection of suspicious activity.

The proposed system defines loitering next to a vehicle as the primary suspicious action. There are also two extra forms of suspicious behaviour. The first one involves accelerating over a certain limit in a parking lot and the second type is parking a car without getting out in a predefined time limit. Normal activity is defined as an owner parking his car, getting out of the car and leaving the area, and returning after a while to leave the parking lot.
Carjacking is one of the most common crimes in SA. There were ±16000 break-ins of cars during the year 2009. [3] This makes the rate of carjacking in South Africa eighteen times greater than the USA and one of the highest in the world! Carjacking’s are costing tax payers billions of Rands while the criminals reap the rewards.

Various organisations employ some form of security system, especially when it houses expensive or vital equipment. The proposed system will be used in tandem with the existing security system. Typically the parking lot’s video security team monitor different areas. The security guard working the video security shift should focus on multiple screens for hours to monitor for suspicious activity and declare a possible break-in. The system merely serves to assist mechanism him. When the application detects suspicious behaviour, it alerts the user performing surveillance by attracting his/her attention to a specific monitor by means of displaying an alert window. The system also plays an alert sound in the case where speakers are connected and the user does not want to stare at the monitor at all. The users decide for themselves whether it is worth investigating or not. The system then logs that alert to a text file accessible using the graphical user interface (GUI) described later on.
CHAPTER 2

USER REQUIREMENTS DOCUMENT

This chapter focuses on viewing the problem from the user’s perspective. The solution is based on the vital information acquired from the user.

2.1 Users view of the problem

The user requires a system that can be incorporated with the existing surveillance equipment. The system must alert the user when it detects suspicious activity, but the GUI must allow for the system to continue running in case of more suspicious activities occurring. It also needs to log previous break-ins and false positives for future reference and possible system extensions. For this to be possible, video analysis and manipulation needs to occur in real-time.

The proposed hardware and software requirements are:

- AMD Opteron 8 core 2378 contained in an X2200 sun server.
- DDR2 800 MHz 8gig server ram
- Graphics card with multiple monitor support
- Kubuntu 10.04 operating system
- Opencv libraries

2.2 Description of the problem

When burdened with the job of focusing on multiple security monitors it can become tedious, and security guards start to make errors and miss the important intervals when monitoring is imperative. This monitoring can continue for hours without any suspicious activity occurring, but it can easily take the security guard by surprise when a break-in occurs. If he misses a few seconds of footage, it can
prove to be the difficult between catching a culprit and letting him escape. Hiring more security guards costs money and does not always solve the problem. A system needs to be built to assist security guards to do their job more efficiently.
2.3 Expectations from the software solution

The system needs to successfully detect suspicious activity. The following objectives need to be achieved:

1) Image Acquisition
   i. A digital image is produced by a security camera.

2) Image processing and analysis
   i. Re-sampling in order to assure that the image coordinate system is correct.
   ii. Noise reduction in order to assure that sensor noise does not introduce false information.

3) Feature extraction
   i. Frame differencing with binary thresholding to remove stationary objects.
   ii. Regions of interest (ROI) such as blobs.

4) Detection
   i. Motion History Image (MHI) to store blobs movement pattern.
   ii. MHI is stored in a cyclic buffer - a single, fixed-sized array as if it were connected end-to-end

5) Object tracking
   i. Using the MHI extracted from the image to determine the movement between images.
2.4 Not expected from the software solution

Detecting burglary of vehicles is not part of the project scope. A human independent suspicious behaviour monitoring system is a possible system, but will not be implemented in this project due to time constraints.
CHAPTER 3

REQUIREMENTS ANALYSIS DOCUMENT

In this chapter the requirements stated in the previous chapter is analysed and the problem is looked at from the designer’s point of view. It focuses on the system and software requirements.

3.1 Designer's interpretation

The performance of the Central Processing Unit (CPU) is quite important because the system will run in real-time. The difficulty lies in the analysis of human movement. The MHI and basic blob tracking must be manipulated in such a way as to tell the difference between loitering next to a vehicle and the owner just getting into his vehicle to drive away. Frame differencing is the foundation for all of the detection. [4] When loitering is detected, it causes an alert signal to trigger, which will be elaborated on later in this mini-thesis.

Figure 3.1: Distinguishing cars from pedestrians
3.2 Breakdown of the problem

Surveillance cameras monitor parking lots and the individual footage for each camera is displayed on multiple screens. The person monitoring the security camera feeds analyses each screen day and night. This simple but tiresome process can easily cause the security guard to lose concentration and miss important disorder in parking lots. The system must analyse the videos in real-time. When suspicious activity is detected the security guard is alerted on the specific screen in which the activity is being detected. Security then decides whether they should pursue the culprit.

3.3 Complete Analysis of the problem

1) **Recording the parking lot activity in real-time:**
   i. Security cameras are required to record activity in each parking lot. This live feed is plugged into the server which redisplay the original live feed in real-time, whilst doing image processing in the backend.

![Figure 3.3.1: Recording the parking lot activity in real-time](image)
2. *Car and Human tracking*

i. We track cars with yellow circles and humans with smaller red circles. Tracking starts once the MHI has enough (three) frames in the cyclic buffer (array). Then we can get the general motion direction of the object (car or person).

*Figure 3.3.2: Car and Human tracking*
3. **Behaviour classification**

i. Now that we can track motor vehicles and people, the only thing left to do is classify certain movements as suspicious activity and the rest as normal activity.

a) Normal activity is parking the car and walking away from the parking lot. Then returning and driving away.

b) Suspicious activity is the loitering around a vehicle. When the red blob is spotted close to the car and fidgets with various parts of the car it gets added to a counter. When the counter reaches a certain peak the alert signal is sent to that particular monitor as seen in the figure.

![Figure 3.3.3: Behaviour classification](image)

Figure 3.3.3: Behaviour classification

c) There are two more suspicious activity types:

- Accelerating too fast out of the parking lot.
- Remaining in the car for a few minutes when parking a car.
3.4 Current Solution

At present there are two official systems that are solutions to the problem. The first one is very similar to the suggested system that will be further explained after these current solutions. The second solution makes use of advanced hardware, rather than emphasizing computer vision techniques.

1. **Samurai**
   
   CCTV cameras are not used to their full potential. The Samurai system alerts the security operator when it detects unusual activity. It was implemented in December 2009 by the euro police. Samurai still needs to be refined and has funding to do so until the end of 2011 to prove its worth. [5]

2. **Future Attribute Screening Technologies (FAST)**
   
   This crime solution is a pre-crime detector. It consists of a battery of sensors that detect security threats from a distance. There are sensors for facial expressions, pulse rate, breathing rate and body heat. The system was found to be 78% accurate using a sample size of 140. This system is fully automated. The cost however restricts it to first world countries that have billions of rands in security capital. [6]

3.5 Suggested Solution

The suggested solution aims to be a cost effective, modifiable and accurate. The focus of this project is a system that will work effectively at detecting suspicious activity in parking lots. Since it is modifiable it will be able to expand into several other systems that all generally detect suspicious activity. It uses the powerful open source library called Opencv to achieve this. [1]
CHAPTER 4

USER INTERFACE SPECIFICATION

In this chapter we explore the user interface for the system. The user interface was built using Kubuntu's native Kdialog. A shell script was written that uses these dialogs to prompt the user for the next action he requires. We will show its typical operation by means of use-case diagrams.

4.1 Start Screen

This is a dialog-based user interface used for its simplicity, speed and stability.

Figure 4.1: Behaviour classification
4.2 How the user interface behaves

1. Loading a video

For demonstration purposes the option to load videos manually was used. The user can also use this feature to play old recordings so that the system helps them find suspicious behaviour. If no camera is detected, the system does not stop. It still allows you to select a pre-recorded video.

![Figure 4.2.1: Behaviour classification](image-url)
2. **Display the alert**

Mplayer is used to show the alert signal and produce the sound alarm too. Each screen will have its own alert video so that the user knows immediately where the suspicious activity is occurring.
3. *Show the log*

When the system finds something suspicious, it immediately saves the parking lot number, the type of suspicious activity and the exact date and time that it was detected.
CHAPTER 5

HIGH LEVEL DESIGN

We finally start looking at the methodology followed in constructing this system. In this chapter we will look at it from a high level of abstraction, while the low level view will follow in the next chapter. Since the system was programmed in C/C++ we do not have an Object Oriented Analysis so we will not include a class diagram.

5.1 High Level Data Dictionary

An object is given with its description to help define the diagrams that follow.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mencoder</strong></td>
<td>Mencoder, a companion program to MPlayer, is a free command line video decoding, encoding and filtering tool. [7]</td>
</tr>
<tr>
<td><strong>OpenCV</strong></td>
<td>Opencv is the open source library used for this purpose. Its main focus is real-time image processing.</td>
</tr>
<tr>
<td><strong>Haar-like features</strong></td>
<td>These features, rather than using the intensity values of a pixel, use the change in contrast values between adjacent rectangular groups of pixels. The contrast variances between the pixel groups are used to determine relative light and dark areas. Two or three adjacent groups with a relative contrast variance form a Haar-like feature. [8]</td>
</tr>
<tr>
<td>Boosting Algorithms</td>
<td>Boosting algorithms are used with pre-classifiers to increase their classification strength. Haar-like features and adaptive-boost is a common combination used for near-real-time to real-time systems. [9]</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Greyscale</td>
<td>It is used to convert colour images to its greyscale equivalent. The conversion, to a shade of grey from a colour image, is established by calculating the effective brightness or luminance of the colour. This value is then used to create the shade of grey that corresponds to the desired brightness.</td>
</tr>
<tr>
<td>Frame Differencing</td>
<td>Get the absolute difference between two images and using the output as the new image. Used to detect between consecutive frames.</td>
</tr>
<tr>
<td>Threshold</td>
<td>Segmenting the image into a foreground and a background. The output is a binary image (black and white image). [10]</td>
</tr>
<tr>
<td>Motion History Image (MHI)</td>
<td>Creates a shadow of previous movement where brighter areas are more recent movement. Integral to data collection in this project for analysing movement in the parking lot. [4]</td>
</tr>
<tr>
<td>Blob Detection</td>
<td>Detects points and/objects in an image that are either brighter or darker than the surrounding points and/or regions using MHI. The object size can also be determined. [4]</td>
</tr>
<tr>
<td>Region of Interest (ROI)</td>
<td>Uses Coordinates to extract part of an image and use that as a key image.</td>
</tr>
</tbody>
</table>
5.2.1 Relationship between objects

In the figure below the relationship between the objects are identified.

![Figure 5.2.1: Relationships between objects](image)

*Figure 5.2.1: Relationships between objects*
5.2.2 Full Solution Component Diagram

In the figure below a component diagram shows the components and sub-components of the full solution to the system.

![Component Diagram]

**Figure 5.2.2: Full solution**

The complete system has both hardware and software components:

- **Hardware:** Cameras that support 25 fps or better with hi-resolution support. A Sun x2200 server will allow 8 parking lots to be monitored simultaneously running off one server. Low-end or better video card with multiple monitor support. [11]

- **Software:** The software resides on the server. This consists of all the computer vision libraries used and constructed. The backend is the bulk
of the system containing all of the processing. The frontend merely gives options to display the desired video or log.

5.2.3 Tracking of Cars and Humans diagram

In this figure the key component to the system is slightly elaborated. It shows a high-level view of how cars and humans are tracked in the system using computer vision methods.

![Tracking of Cars and Humans diagram](image)

*Figure 5.2.3: Blob Tracking*
### CHAPTER 6

**LOW LEVEL DESIGN**

#### 6.1 Low Level Data Dictionary

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greyscale</td>
<td><code>Image buf[last]</code> is converted to greyscale and stored in <code>Image img</code></td>
</tr>
<tr>
<td>Frame Differencing</td>
<td><code>Image buf[idx1]</code> and <code>Image buf[idx2]</code> are subtracted from each other.</td>
</tr>
<tr>
<td>Threshold</td>
<td><code>Int diff_threshold</code> holds the value of the threshold to convert the greyscale image to a binary image and stores the image in <code>Image silh</code></td>
</tr>
<tr>
<td>Motion History Image (MHI)</td>
<td><code>Image silh</code> puts images into the motion buffer <code>Image mhi</code> using <code>int timestamp</code>, which calculates the time each blob gets detected</td>
</tr>
</tbody>
</table>
Blob tracking

Each time a blob is detected its coordinates are saved to create a ROI, which enables it to be tracked. The orientation is measured using Image orient and the function cvCalcMotionGradient. Next the component height and width is determined by the ROI. It is stored in int comp_height and int comp_width respectively. A big comp_height + comp_width is a car and a smaller sum is a person.

6.2 Why not machine learning classification

1. **Speed**

   The first method we used was the haar-feature classification method. See 5.1 High Level Data Dictionary. We had to train over 2000 samples for there to be a respectable accuracy. However, we only got it working at 15 frames per second. [8] This was adequate to run one camera thread after dropping a few frames, but it was not a scalable solution.

2. **Accuracy**

   An accurate classification method such as support vector machines was another consideration, but once again it is not scalable to be used on many parking lots. It can easily be retrained, but it does not give us real-time performance, especially when running more than one camera. [12] Since it was not considered in this system, we will not explain it in further detail.
We need a system that is accurate to a holistic object level. MHI is reliable at tracking blobs and blazingly fast (faster than real-time running on the same server).

3. Complexity

Considering the fact that we are not tracking on detailed features such as face, eyes and mouth, we do not need to use a complex array of classifiers for high accuracy and real-time speed. [13] We therefore did not consider creating strong classifiers using boosting algorithms or any other combination of classifiers See 5.1 High Level Data Dictionary.

6.3 Detailed Methodology

We will now discuss the methodology used to create this system by elaborating on the following key computer vision techniques used:

1. Greyscaling
2. Frame Differencing
3. Threshold
4. Motion History Image
5. Blob tracking
1. **Greyscaling**

When we have the captured frame we use this method to convert colour images to its greyscale equivalent on the fly. The greyscale image shows the effective brightness or luminance of the colour image. Red, green and blue values are multiplied by 0.3, 0.59 and 0.11 respectively. The sum of the product is calculated to give the single pixel value, which is greyscale. [14]

Since our system needs to run in real-time, we need to use techniques that process faster than it would have taken to process the original frame. Greyscale is also necessary because since its only one value rather than three, it is easier to manipulate the frames. [14]

![RGB vs. greyscale](image)

*Figure 6.1.1: RGB vs. greyscale*

2. **Frame differencing**

This method subtracts two images from each other to get the absolute difference. This is used when the still part of an image needs to be separated from the moving part of the image. It is worth noting that
traditional frame differing techniques contained “double vision”. This was solved by taking the second difference of the image. This method is called the absolute difference and was the preferred technique used in this system. [15]

As we have emphasized earlier, our system needs to find the least hardware intensive method to detect suspicious activity. We are also aiming for a respectable accuracy of 90%. In order to achieve these results before we do blob tracking, an acceptable background subtraction algorithm is favoured; however, some are processing intensive. We have, however, discovered that since we are looking at two specific blob types, namely the car and human, we do not need to use a sophisticated background subtraction technique. We simply used the binary threshold to remove the still background and noise, which is explained next. [15]

![Figure 6.1.2: Frame difference and threshold](image)
3. **Threshold**

During the frame differencing process a binary threshold value needs to be used to get rid of unwanted noise whilst preserving the foreground. Setting the threshold to the correct value is important so that we get acceptable results in blob tracking. [15]

4. **MHI**

The motion history image is the final component for blob tracking. This method focuses on accumulating and recognising entire object patterns rather than focusing on detailed features. The advantages of this approach are the low use of memory and blazingly fast yet descriptive representation of capturing motion on a per blob basis. [4] It has coordinates that can be used to measure the width and the height of the blob component. The relative angle at which the blob is traversing is also measured. The MHI is created by layering images over successive image regions using the following formula to update its history:

\[
MHI_\delta(x, y) = \begin{cases} 
\tau & \text{if } \Psi(I(x,y)) \neq 0 \\
0 & \text{else if } MHI_\delta(x, y) < \tau - \delta 
\end{cases}
\]

*Figure 6.14: MHI displayed as a blue image*
5. **Blob tracking**

Now that we have the coordinates and the size of the blobs all that is left to do is define people as smaller blobs than cars and then create constraints for suspicious activity as discussed in 3.3.3. [13]

*Figure 6.1.5: Cars and Humans are distinguished*
6.4 State Diagram

The diagram below shows a more detailed view of the different states the system goes through to detect suspicious activity.
CHAPTER 7

TESTING

The testing system was not a server, but rather a PC that has the following specifications:

- AMD 955 Phenom II 3.2 GHz
- OCZ 1600 MHz DDR 3 4gig ram
- On-board graphics card ATI 3300 HD
- Kubuntu 10.04 as the operating system

7.1 Stress Testing

Since the system would continuously run on the server, we needed to make sure it will be reliable. We created a script to do 100 iterations of the pre-recorded videos that are used for demonstration purposes. A minor bug was found after 10 iterations, but it was easily fixed. We then re-ran the script for 100 iterations and had no further stress problems.

It was noted that it ran slightly faster than real-time, but when two pre-loaded videos were simultaneously loaded it created two threads and ran at real-time 25 frames per second with a resolution of 640 by 480.
7.2 Correctness Testing

Once we completed stress testing we wanted to measure the real-world accuracy of the system. Hardware accessibility was, however, a big problem. We had to use a high vantage point to shoot the video, but since the equipment was all electronic and fragile, we used a laptop with a built in web cam.

- Intel core 2 duo pentryn 2.1 GHz
- Corsair 800 MHz DDR 2 4gig ram

The first and biggest problem we encountered was the weaker processor. The system however managed to get 5 out of 5 matches correct, but the last test had some frame skipping when we tracked more than two cars, so it didn’t track as well as the other 4 iterations, but it did not incorrectly classify the behaviour.

Real-time non-real-world tests were also conducted. The camera was positioned just over three metres from the ground. With the parameters adjusted to a different angle and zooming level it could track objects just as well as it did from a high vantage point. The high vantage point was 6-7 metres high. The reason we could still track these objects with very minor changes was due to the fact that we could once again use our fast PC instead of the slower laptop.

This concluded the real-time testing and although a 100% accuracy was noted, which satisfied the >90% requirements mentioned earlier in the document, it did not mean a 100% break-in detection rate. The system was built for suspicious activity detection and not carjacking detection. Well at least not yet.
CHAPTER 8

USER MANUAL

The GUI system contains two main parts:

- Demonstration mode
- Real-time mode

8.1 Starting up the system

At start up you are prompted to either start the system by selecting “yes” or viewing the log by clicking “no”.
1. **Starting the system**

If a video camera is attached to the server then it will start in real-time mode. In real-time mode it immediately starts monitoring for suspicious activity.

With no video camera attached it starts in demo mode. In demo mode it will prompt the user to select a pre-recorded avi format video.
2. *Viewing the log without starting up the system*

Here the user can either view the log file or be given the option to exit again.
If the user chooses to view the log then it will pop-up and be given the option to exit or still start the system.
8.2 Using the system in demonstration mode

In demonstration mode you can select one of five videos as seen from the prompt screen on the previous page.

1. Drive by video

In this first video a car innocently drives past a parking lot. This displays the first Normal activity video.
2. *Two people leaving a parking lot normally video*

The next video displays two friends leaving a parking lot in a non-suspicious manner.
3. **Two men loitering (Primary suspicious activity)**

This is the first and main suspicious activity demo. It shows how two men are tracked around a car and classified as suspicious.
4. *Suspiciously not leaving his vehicle after parking it*

For this video the system was slightly modified to classify twenty seconds as suspicious instead of two minutes.
5. *Accelerating too fast out of a parking lot*

This is a particularly useful extension to the system. It can be used to track down reckless drivers that accelerate so fast that their tyres lose traction.
CHAPTER 9

CODE DOCUMENTATION

The code documentation as well as the installation documentation can be found on the accompanying DVD. It contains a ready to use virtual machine for use with the virtual box software.

The code documentation contains comments to each method as well as to important statements.
CHAPTER 10

CONCLUSION

This documentation has been written to describe the process of producing a real-time suspicious activity detection system that can be used in multiple parking lots. The Incremental Model was the software development process used in this project. The requirements were gathered, analysed, designed, coded and tested. The steps were iterated a number of times. Maintenance was however not part of the scope of this project.

We are happy to state that the system has satisfied the requirements gathered and we are excited about room for future extensions to this project.

We conclude this mini-thesis with special thanks to Mr James Connan, Mr Mehrdad Ghaziasgar and the University of the Western Cape for giving us the opportunity to work with such an exciting and practical project that has great promise for the future.
REFERENCES


   http://tripatlas.com/Carjacking/#South%20Africa


   http://www.cse.ohio-state.edu/~jwdavis/CVL/Research/MHI/mhi.html


   http://www.sun.com/servers/x64/x2200/specs.xml


