Stationary Object Detection in Video

Filip Leiding

Master’s Thesis in Computing Science, 30 ECTS credits
Examiner: Fredrik Georgsson
Supervisor: Mikael Rännar

August 24, 2015
Abstract

Computer Vision is an expression which summarise the area of which image and video analysis is made automatically by computers after given instructions for a specific purpose. VCA (Video Content Analysis) is a subcategory which handles analysis on video files or cameras. The progress within VCA has been driven by the necessity of effective video surveillance in areas where high security is demanded. Object detection within VCA can be used not only for security but for notification of movement or stationary objects to provided sufficient measures.
Acknowledgements

I would like to thank Johanna Björklund and Rickard Lönnéborg at CodeMill for taking me in and letting me do my thesis at their company. I would also like to thank my supervisor at CodeMill, Ludvig Wadenstein for being there for me when I had questions for struggled with a programming issue. And last but not least I would like to thank my supervisor Mikael Rännar at Umeå University for giving me recommendations an correct my faulty grammar in this report.
## Contents

1 Introduction ................................................. 4  
   1.1 Purpose ............................................. 4  
   1.2 Related work ....................................... 4  

2 Video Content Analysis - Object detection ............... 6  
   2.1 Image representation .................................. 6  
   2.2 Background subtraction ................................ 7  
   2.3 Non-recursive background modelling ..................... 8  
      2.3.1 Frame differencing ............................... 8  
      2.3.2 Median filtering .................................. 8  
      2.3.3 Non-parametric modelling ......................... 9  
   2.4 Recursive background modelling ....................... 9  
      2.4.1 Approximated median filtering ................... 9  
      2.4.2 Kalman filtering .................................. 10  
      2.4.3 Gaussian mixtures ............................... 10  
   2.5 1-model background subtraction ....................... 14  
   2.6 2-model background subtraction ....................... 16  
   2.7 Comparison ........................................... 16  

3 Implementation ............................................. 20  
   3.1 System overview ...................................... 20  
      3.1.1 Graphical user interface ......................... 21  
      3.1.2 Background subtraction ......................... 25  
      3.1.3 Foreground sampling ............................. 26  
      3.1.4 Detection and notification ....................... 26  
      3.1.5 Saving to file .................................. 27  
   3.2 Program and libraries .................................. 27  
      3.2.1 Python ........................................... 28
Chapter 1

Introduction

1.1 Purpose

This thesis is a project executed together with CodeMill AB, [16] which is an IT consulting firm location in Umeå in northern Sweden. CodeMill specialises in the field of media and broadcast. This thesis was formed out of a demand for automatic notification when certain objects have been left at improper places. Places where this type of problem could occur is for example at loading docks when new sets of cargo have been delivered and needs to be taken care of, emergency exits where objects are blocking way and immediately needs to be removed for safety reason. The last and most critical example is abandoned bags at airports which can potentially be very dangerous. The questions to be answered in this thesis are as follows:

1. How can one detect changes in otherwise static environments?
2. When does an object become static or non-static?
3. How can one filter out static data from non-static data?

1.2 Related work

With the increasing demand for automated surveillance programs and algorithms, researchers today have abandoned the use of simple, unadaptive background subtraction models since the areas of usage constantly changes
over time. Now the focus lies on improving the existing models to fit the modern needs of automated surveillance. Many researches such as S.C Cheung et al.[2], Medha Bhargava et al.[11] and Sen-Ching S et al.[7] are trying to improve the background models responsiveness and accuracy for functioning in crowded areas such as train stations or rush hour traffic sites. Another side of the modern needs is the ability to distinguish objects from its shadow to reduce the number of faulty detections when the environment light changes or when an objects shadow enter the scene but not the object itself. Kaew Tra Kul Pong et al.[12] and Thanarat Horprasert et al.[13] have been working with this kind of improvement. Some other researchers have been trying to improve the overall function of background subtraction by using new and different approaches. Rubén Heras Evangelio et al.[10] are using a 2-model background model together with a finite state machine to detect static objects and Antonio Albiol et al.[5] are using spatio-temporal maps to detect stationary objects within predefined areas of a scene. Thi Thi Zin et al.[6] and YingLi Tian et al.[8] have done research about object detection suited for real surveillance application where security is the main purpose. In the paper Wallflower: Principles and practice of background maintenance by K. Toyama et al.[3] they have developed a new background subtraction algorithm which they clam are one of the best algorithms to use, at least when the paper was written.
Chapter 2

Video Content Analysis - Object detection

2.1 Image representation

Video content analysis is the study of visual changes and events in video streams and video files. Video files are collections of images put together in a specified sequence which are shown at a certain speed to make the illusion of motion to the human eye. To make analysis of video, one must go down to the image level and perform analysis on one images at the time. Video content analysis, in particular object detection is made at an even smaller scale than just an image, namely pixel level. To be able to understand the concept in this paper an introduction to digital image representation will follow.

An image is represented by pixels in the digital world and inside computers an image is represented by a 2-dimensional matrix where each element in the matrix is one pixel. The colourspace of the image determines the dimension of the images represented. If the image is in grayscale, each pixel will only contain one value, from 0 to 255 (in 8-bit colourspace per channel) (see Figure 2.2 on the facing page). In RGB (Red, Green, Blue) colourspace, each pixel has three values, one per channel, and this also makes the images representation 3-dimensional. As for the grayscale representation each channel in RGB colourspace can take a value between 0 and 255 (see Figure 2.1 on the next page).
2.2 Background subtraction

One of the basic approaches to detect stationary objects in a video stream is to apply the background subtraction model. This technique is based on a comparison between a stored background image used as a reference of the scene and the next frame in line, described in the paper by Smitha H.[1]. The background will function as a ground layer and everything that was not present in the scene when the background was created is considered a foreground object (object not belonging to the background). This separation between background and foreground is what makes the foundation in object detection by background subtraction. The first background reference is often obtained by taking \( N \) number of frames when the camera starts and take the average or median of the pixel values from all the frames. This is a very simple but effective way to background image. Since almost no scene is constant over time many different models have been developed to adapt the background image when changes in the scene occur. They have to be robust against environmental changes such as illumination but also sensitive enough detect the objects of interest.

An introduction to some of the most used models will be presented below and a comparison between them will be made to distinguish the areas of use for each model. The models can be divided into two subcategories; non-recursive and recursive (no buffer of past frames needed).
2.3 Non-recursive background modelling

Non-recursive modelling is based on using buffer of size \( N \) to store previous frames of the scene. The stored images are used to estimated the background images using the variation of temporal values of each pixel in each frame within the buffer. These techniques can vary in its adaptiveness depending on the size of the buffer. A large buffer implies that the adaption takes longer time and the storages requirement for the using equipment increases and vice versa for a smaller buffer.

2.3.1 Frame differencing

This is the most simplest technique for modelling an adaptive background. Frame differencing takes the current frame \( F \) at time \( t \) and compare it to the previous frame \( t-1 \).

\[
|F_t(x, y) - F_{t-1}(x, y)| > T
\]  

(2.1)

Where \( x \) and \( y \) represent the spatial location of a pixel within the frame and \( T \) is a threshold deciding if the current pixel is significantly different from the previous frame. If the pixel value is greater than \( T \) it will get a value of 1 and if not a value of 0 in the binary mask that is created. This technique is very quick in adapting to changes in the scene but it also have a large problem which is if an object is unified in colour, only the outer lines of the object will be marked as moving. This is because frame differencing only compares the previous frame and thus the object with its unified colour will still be in the same region and the pixel colour will not have changed significantly.

2.3.2 Median filtering

Median filtering is based upon frame differencing but the difference between them is that median filtering uses a larger buffer of images and instead of comparing two pixel values it takes the median of all the frames in the buffer. For a pixel to be detected as moving in this technique, it must have stayed in the background for more than half of the buffer size. The median only works if the images is in grayscale where the medoid is used if the imaged is in a coloured space.
2.3.3 Non-parametric modelling

Non-parametric modelling is similar to the other non-recursive methods but instead of having a fixed buffer of frames for estimating the background update, it uses all of the previous frames to make the estimate parameter independent. This method was constructed by Elgammal et al. [14] by using all the previous frames to estimate the pixel density function $f(P_t = u)$:

$$f(P_t = u) = \frac{1}{N} \sum_{i=t-N}^{t-1} K(u - P_i) \quad (2.2)$$

The $K$ represents the kernel estimator presented by D. W Scott in [15] where the kernel must be a symmetric distribution. In [14] the kernel distribution is the normal distribution.

If the current pixel does not come from the chosen distribution, it will be declared as a foreground pixel.

2.4 Recursive background modelling

Recursive modelling in contrast of non-recursive is that recursive modelling does not store frames from the past in a buffer for comparison. Instead the background is updated with every new frame. This means that the background is affected by frames from the distant past and can have errors lingering for a very long time. Despite that, these types of techniques do not require any larger storage capabilities. Most of these techniques implements weighting variables to discount the past frames faster to reduce the error time of pixels.

2.4.1 Approximated median filtering

This method is based upon its non-recursive twin where comparison between the current and the previous frame is made. Only this time an estimate is made from the median in every new frame, where the estimate is increased by one if the new frame is bigger than the estimate and decreased by one if lower (on pixel level). This only works on grayscale images just as the non-recursive model. If you line up all the frames that have passed half of them would be behind the one with the median pixel and half would be in front, which by definition is the median. Note that this will occur individually for
every pixel in the frame. So the same frame could be on either side in the line depending on which pixel you look at.

### 2.4.2 Kalman filtering

Kalman filtering is a linear prediction which uses the current frame values and mixes them with a prediction of the current frame with help of the previous frame to get an estimated frame. The estimated result at time $t_i$ is

$$
\hat{x}(t_i) = \bar{x}(t_i) + K(t_i) \cdot [z(t_i) - H(t_i) \cdot \bar{x}(t_i)]
$$

where the prediction term is defined as

$$
\bar{x}(t_i) = A(t_i) \cdot \hat{x}(t_{i-1})
$$

$A(t_i)$ is the system matrix which is a constant matrix defined as

$$
A = \begin{bmatrix}
1 & a_{1,2} \\
1 & a_{2,2}
\end{bmatrix}
$$

where the values of $a_{1,2} = a_{1,2} = 0.7$ as used in [4]. $H(t_i)$ is called the measurement matrix and is also a constant matrix.

$$
H = \begin{bmatrix}
1 & 0
\end{bmatrix}
$$

$z(t_i)$ is the input matrix, the current frame from the camera and $K(t_i)$ is the gain matrix. This matrix is derived from the error covariance matrix and if the gain is high, the noise of the input is low and vice versa. So the Kalman filter procedure gets its estimated matrix by weighing the difference between the prediction matrix and the current input matrix. So values which have a large difference from the input matrix will get a lower weigh which means that errors do not linger for a long time. See a graphical explanation of the process in Figures 2.3 on the facing page, 2.4 on the next page and 2.5 on page 12.

### 2.4.3 Gaussian mixtures

Gaussian distributions operates on single values so to be able to explain how it works in the context of images we need to go down to pixel level. This model is called mixtures because each and every pixel is compared to
Figure 2.3

Figure 2.4
Figure 2.5
a set of distributions, ranging from 3 to 5 distributions mostly. Why more than one distribution is used is to be able to ignore objects that belongs to the background but are not stationary, such as swinging leaves, snow and rain but even reduce the detection of shadows. Several distributions makes it possible for multi modal backgrounds which means that a pixel can take several different colour values and still not be classified as a foreground object. Every pixel at each frame is compared to the set of mean values

\[ \mu(K) = \{\mu_1, \mu_2, ..., \mu_K\} \]  

(2.7)

and a set of variances

\[ \sigma(K) = \{\sigma^2_1, \sigma^2_2, ..., \sigma^2_K\} \]  

(2.8)

where \( K \) is the number of Gaussian distributions used. The distribution created with the \( K \) gaussian distributions can be described as

\[ Z \sim N \left( \sum_{i=1}^{K} \mu_i, \sum_{i=1}^{K} \sigma^2_i \right) \]  

(2.9)

This new distribution can have a similar look as this illustration below (see Figure 2.6)

Figure 2.6: Gaussian mixture using 3 distributions

The probability of a pixel being inside this distribution can described by setting up a confidence interval as follow

\[ 1 - \alpha = P = \left( -\frac{\lambda_{\alpha/2}}{\sigma} < \frac{X - \mu}{\sigma} < \frac{\lambda_{\alpha/2}}{\sigma} \right) \]  

(2.10)
where $P$ is the probability the the pixel is inside the quantiles and $\alpha$ is the area of the distribution outside the quantile limits. The $X$ is the value of the current pixel. The quantile limits are usually set by how many standard deviations the result can differ from the mean value. One standard deviation away from the mean value corresponds to an $\alpha = 32\%$, two standard deviations $5\%$ and three standard deviations $0.3\%$. The chosen alpha usually lies between 2 and 3 standard deviations from the mean value. If the pixel is inside this interval it counts as the background, otherwise it is detected as foreground.

### 2.5 1-model background subtraction

The simplest approach to detecting stationary objects is to use the 1-model background subtraction which only uses a single comparison of the scene (see Figure 2.7 on the facing page). The model is based on a comparison between the current frame and the stored background frame. The background $BF_t(x, y)$ where $t$ is the current frame and $x$ and $y$ are the spatial coordinates for the pixel, is created at first by taking $N$ number of frames at startup and take the median values of the pixels to create an image as the reference BF. To separate new objects from the objects of the background image, a foreground $FF_t(x, y)$ is created. This is done by taking the difference between the current frame and the background frame and create a mask for the foreground image $M_t(x, y)$. The pixels in the mask can take the number 0 and 1 depending on the value from the difference.

$$|FF_t(x, y) - BF_t(x, y)| > T$$  \hspace{1cm} (2.11)

Where $T$ is the threshold chosen and $M_t(x, y) = 1$ if the difference is above $T$ and $M_t(x, y) = 0$ otherwise. This will create an image which is black and white, where the white areas are objects that does not belong to the background. For an object to be specified as stationary, it must be at the same spot for a period of time. A sample of $M$ frames must be collected and a foreground mask for each frame must be created. Since the masks are in binary form, the logic operator AND, denoted as $\land$, will create a sampled mask

$$S = (M_{t-n}(x, y) \land M_{t-n+1}(x, y) \land \ldots \land M_t(x, y))$$  \hspace{1cm} (2.12)

where $S$ only contains the objects that have been detected in all of the sampled masks. Those pixels that are still white in the mask correspond to
the stationary object in the scene. This method is very basic and therefore it has some restrictions. It can not distinguish permanently stationary objects from temporary stationary objects, for example a person who stops to tie a shoe. It would alert the system for every object that is standing still for a moderate period of time. To solve this problem a similar but slightly more advanced model was invented, described in the next section.

Figure 2.7: 1-model and 2-model background subtraction illustration
2.6 2-model background subtraction

In this method, the subtraction of the background will occur at two different rates. One of the background images is updated every frame and the other one is updated every $L$ frames (see Figure 2.7 on the previous page). Masks for both backgrounds are created at respectively rate. The short term background $SB_t(x,y)$ will be compared to the current frame $CF_t(x,y)$ and every pixel will either increase in intensity or decrease depending on the result of the comparison.

$$|CF_t(x,y) - SB_t(x,y)| > T \quad (2.13)$$

Equality between pixels leave the pixel unchanged. This enables the $SB_t(x,y)$ to change quickly in scenes where the lighting conditions change rapidly. The long term background $LB_t(x,y)$ will do the same process at every $L$ frames and is compared to the current $SB_t(x,y)$ to gradually adapt to the environment of the scene. By increasing or decreasing the intensity of the pixels, stationary objects will slowly become a part of the background and moving objects will still only be part of the moving foreground. By having two backgrounds updating at different intervals, detection of temporary stationary objects and objects that were part of the background before but has been moved can be made. This method can only be applied to frames who are in the grayscale colourspace.

2.7 Comparison

Under this section a comparison between the different background modelling methods will be made and be summarised with a table describing each methods important features (see Table 2.1 on page 19). To start the comparison I will first separate recursive and non-recursive methods and compare the methods within these two categories and then do a comparison between the two groups.

The non-recursive models are very similar to each other since both median filtering and non-parametric modelling is based on the frame differencing technique. The frame differencing model and the median filtering model both compares the current frame with the previous frame or buffer of frames while the non-parametric modelling technique uses all of the previous frames to make an estimate and then compare the estimate to the current frame.
The base function is as said very similar between these three and the biggest difference between them is that only the non-parametric model works on colour images while the other two only work for grayscale images (with a small exception for the median filtering where the medoid can be used for colour images). The non-parametric model is the most sophisticated but requires a large buffer for storing all of the previous frames, which can be quite many if the recording goes on for a longer period of time.

**Non-recursive modelling**

**Pros:**

- Overall quick adaption to changes in the scene.
- No lingering error from past frames (except for non-parametric modelling).

**Cons:**

- Require storage capabilities, larger means slower adaption.
- Only works for grayscale images (except for median filtering using medoid).

The recursive models are not as similar to each other as the non-recursive models are. All of these techniques have different approaches which makes them harder to compare. The approximated median filtering has the same base as the median filtering, but this time no storage at all is used, instead a threshold is used to either increase or decrease the background pixel value by one. This unfortunately makes this model useless for colour images. The Kalman filtering uses a prediction matrix and the current matrix to update the background reference matrix and thus it does not need to store frames for the estimate. To get this technique to work, some initialisation and predefined variables are required. Gaussian mixtures is the most sophisticated model presented in this paper because of its multi modal property and its ability to work on both grayscale and colour images. Gaussian mixture uses a statistical approach to determine the similarity of the current pixel and
the background pixel. This model also requires some predefined variables to work. These variables must be set to determine the sensitivity of the model. The Kalman and the Gaussian models have the colour images capability in common while the approximated median filtering only work for grayscale. The only thing all of these three models have in common is that they only compare the background frame with the current frame and that the background image is updated after each frame.

Recursive modelling

Pros:
- Most of them work on colour images.
- Require no storage of frames.

Cons:
- Can have lingering errors for a long time.
- Require some initialisation before start.

So what does these two major categories of techniques have in common? Well, all of the techniques are used to update the background image to make it adaptive to changes in the scene, but in different ways as explained above. As summarised in the table (see Table 2.1 on the next page) the major difference between the two groups is that the majority of the recursive techniques have a tendency to be a bit slower than the non-recursive techniques. This could be explained by looking on how much computation that needs to be done at each new frame. In the non-recursive techniques a lot of information is stored, so the amount of computation needed is much lower compared to the recursive models where nothing is stored. This means that all of the information needed to make the background update is computed at every frame. This makes the recursive models somewhat slower because of redundant computations. To say that recursive techniques are better than the non-recursive ones is not a valid statement until the context of usage has been revealed and the type of hardware to use have been decided. When these two variables are known, then one can compare the effectiveness of these different groups of techniques.
<table>
<thead>
<tr>
<th>Technique</th>
<th>Multi-modal</th>
<th>Shadow Det.</th>
<th>Adaptive Rate</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Differencing</td>
<td>No</td>
<td>No</td>
<td>Fast</td>
<td>Non-recursive</td>
</tr>
<tr>
<td>Median filtering</td>
<td>No</td>
<td>No</td>
<td>Fast</td>
<td>Non-recursive</td>
</tr>
<tr>
<td>Non-parametric modelling</td>
<td>No</td>
<td>Yes</td>
<td>Slow</td>
<td>Non-recursive</td>
</tr>
<tr>
<td>Approx. median filtering</td>
<td>No</td>
<td>No</td>
<td>Fast</td>
<td>Recursive</td>
</tr>
<tr>
<td>Kalman filtering</td>
<td>No</td>
<td>No</td>
<td>Slow</td>
<td>Recursive</td>
</tr>
<tr>
<td>Gaussian mixtures</td>
<td>Yes</td>
<td>Yes</td>
<td>Slow</td>
<td>Recursive</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison summary of background models.
Chapter 3

Implementation

3.1 System overview

Figure 3.1: Program flow chart
3.1.1 Graphical user interface

The implemented GUI is a very simple one with only some basic functionality. Upon startup the user can choose to save the captured frames. This feature can be switched on and off during runtime (see Figure 3.2). Another immediate setting which can be changed either before or during runtime is the learning rate of the background where a high number represents a slow background update. The last setting of the startup window is the pixel area detection which controls the threshold of the area of objects that should be detected as stationary. With lower number, smaller objects can be detected. This unit is measured in square pixel which is calculated after the contours of the detected object have been calculated. When the record button is pressed the program starts to capture from the source the user have entered in the video input source setting (see Figure 3.3 on the next page) which can be found under file in the menu bar.

![Object Detection](image)

Figure 3.2: Graphical user interface startup

The program can take any type of video file or a number which represent the source of a connected camera, 0 is the default number for a connected camera or a built in camera. When a valid source is entered the option to pause, exit or watch (see Figure 3.2) the capturing live comes available for the user. Pressing the pause button will stop the capturing and even the
saving of the captured frames, but the program will still be running. The background reference will be reseted and will become the last frame before the pause button was pressed. This enables the user to pause the capturing, ignore the changes made during the pause and then continue the capturing like the pausing never happened. The exit button will exit the program as well as the exit button in the menu bar. The view button will enlarge the window and display the captured images in real time (see Figure 3.4 on the next page).

![Video Input Source](image)

**Figure 3.3: Video source input**

When live view is enabled, another option for the user comes up, namely to extend the window further for an extend view mode (see Figure 3.5 on page 24). This mode not only shows the current frame but also the applied binary mask and the sampled images when an object has been stationary for a period of time.

The last setting the user can make is to type in their destination for the notification mails (see Figure 3.6 on page 24). These mails are sent to the user when new detection occurs. The user can specify which mail address it should be sent to and which mail address the sender should have. Under that the subject and the actual message can be specified to the user’s preferences. If the used mail server requires login credentials, the optional input fields for username and password can be used. The last input field is the address to the mail server with a targeted port.
Figure 3.4: Live view of the capturing
Figure 3.5: Extended live view with binary mask applied

Figure 3.6: Mail account credentials
3.1.2 Background subtraction

The purpose of the program is to read the next frame from the input source and make a background subtraction against the background reference image. This is made by simply take the current frame and then subtract the background reference image. This is a simple binary operation since the images are stored in matrices so every operation is made for every pixel representation in the matrices. A simple subtraction between the matrices will result in matrix with a lot of noise in it since the subtraction is absolute (see Figure 3.7). This means that even if the pixels just differ with decimals it will be visible.

![Simple unfiltered background subtraction](image)

Figure 3.7: Simple unfiltered background subtraction

This program uses a predefined method for subtraction the images. This method uses morphological filters which filers out pixel which have very small differences (see Figure 3.8 on the next page). These filters provides improved accuracy and less faulty detections when the binary masks are applied.

When the subtraction has been made the resulting matrix is used to make a binary mask which then is used for the foreground separation and the detection of objects not belonging to the background.
3.1.3 Foreground sampling

The foreground mask is applied on every frame and at frame 0 and at frame \( N \), the mask is saved as a sample. When the frame number is \( N \), a comparison is done by taking the masks at frame 0 and frame \( N \) and perform a logical AND operation. This operation creates a new binary matrix with pixels in white only if the corresponding pixels in both of the sample frames are white (see Figure 3.9 on the facing page). This operation will detect if an object has been present in both of the frames when the sample was taken, indication that an object has been left abandoned. Only two samples are taken instead of a whole series of them in shorter interval because if only one of them do not have the object and the other ones has, the object will not be detected. So only the first and last sample frames counts.

3.1.4 Detection and notification

When the sample comparison is made, a built in function in OpenCV searches for white pixels in the matrix and finds the contours of the stationary object.
Figure 3.9: Foreground sampling compared with logical AND

When the contours have been found, another built in function calculates the area within the contours. This area is compared with the threshold specified of the user in the GUI. If the areas is larger than the threshold, a rectangle around the object is drawn on the current frame. When the rectangle has been drawn a notification is sent to the specified email address with the user specified message and subject (see Figure 3.6 on page 24).

3.1.5 Saving to file

If the user chooses to save the captured frames, they will be saved after every operation has been made. After the program exits, all of the frames are put together into a video file. The program uses XVID codec and AVI as file format but can be changed to the users liking. If the frames are saved to a file, the program will wrap it up in an MKV container together with the detection chapter text file.

3.2 Program and libraries

Below are short descriptions of the major programs and libraries which was used during the making of this program for object detection. Every extension and plug-in within these libraries will not be listed and explained but can be found at the respective web page.
3.2.1 Python

Python is a very diverse and multifunctional language that is supported almost everywhere on every platform. It can be used both as a scripting language as well as an object oriented language [19], which this program uses it as, or as a mixture of both scripting and object oriented. The possibility to embed other languages inside Python make its usage almost limitless. This language is chosen for its diversity and its support on many major but also minor platforms. This is to make this program as universal as possible without compromise too much.

3.2.2 Tkinter

Tkinter is a graphical user interface (GUI) toolkit which comes embedded in the Python environment when installed. This framework for creating GUI’s are very similar to the Swing package in Java and is very easy to use and creating something small but functional takes very little time. This toolkit is mostly built for small GUI’s and therefore its functions are limited to only support the most basic needs. If creation of big scale and advanced GUIs another library is recommended. This toolkit was chosen for its simplicity since the GUI will only consist of a few buttons and input lines for the user to change some variables and settings during runtime. This toolkit is well documented and have examples of every button and item it can support [20] and also what input and output every method has for easy understanding and usage.

3.2.3 OpenCV

OpenCV (Open Computer Vision) is a set of libraries written in optimised C and C++ with the intention to be computational effective for real-time programs and applications. These libraries is made under the BSD licence which gives the user the right to use it freely under commercial as well as under academic purposes. OpenCV has many well designed methods for VCA which are constructed after well known research papers with robust techniques. The methods used in this paper are mostly for the background subtraction and the background update. Documentation of all the methods and attributes of these can be found at the OpenCV webpage [21]
3.2.4 MKVmerge

MKV files or Matroska files is a media container which can hold video, audio and subtitles in a single file [17]. These files are not of video or audio format, just a simple container for files. To create this container a program called MKVmerge [18] is used which takes a video file and can take a text file containing chapters marks for the video file and then creates a new MKV file. This program is used to mark new detections from the implemented program to be able to search the file for detection events instead of having to go through the whole video.
Chapter 4

Results

4.1 Test Results

During the implementation of the program used in this thesis, test files with objects being left behind and abandoned have been used to make the program work correctly. These files were taken from a website and were used originally for the CAVIAR project [22]. These files have been specifically designed to test different forms of computer vision and video analysis scenarios. The files used in this project are the files where objects are left somewhere in the scene by different people. Below are images taken from test files from the CAVIAR web site (see Figures in figure 4.1 on the facing page).

4.2 Comparison Between Languages

To be able to get an understanding of the performance of this program I have written the program both in Python and in C++ with OpenCL to see if it was possible to make the computations run faster. The programs have been tested on an Intel Core i5 1.3 GHz with a integrated Intel HD Graphics 5000 as graphics unit. Both of the program have been tested with the same kind of test through a live webcam with the same objects to detect. Each test processed 1000 frames. Below is a summary of the different average frame rates at different resolutions (see Table 4.1 on page 32 and Figure 4.1 on page 32).

The difference between the languages when live view was enabled can not be determined at the lower resolution because the web camera can not record
Figure 4.1: Tests made with CAVIAR test files, left side are scene without objects, right side marked the abandoned objects.
<table>
<thead>
<tr>
<th>Language</th>
<th>Resolution (pixels)</th>
<th>Average frame rate (FPS)</th>
<th>Live View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>320x240</td>
<td>30.02/14.95</td>
<td>No/Yes</td>
</tr>
<tr>
<td>Python</td>
<td>640x480</td>
<td>30.0/10.77</td>
<td>No/Yes</td>
</tr>
<tr>
<td>Python</td>
<td>1280x720</td>
<td>18.9/6.22</td>
<td>No/Yes</td>
</tr>
<tr>
<td>C++</td>
<td>320x240</td>
<td>30.3/15.87</td>
<td>No/Yes</td>
</tr>
<tr>
<td>C++</td>
<td>640x480</td>
<td>30.3/12.20</td>
<td>No/Yes</td>
</tr>
<tr>
<td>C++</td>
<td>1280x720</td>
<td>20.41/6.49</td>
<td>No/Yes</td>
</tr>
</tbody>
</table>

Table 4.1: Summary table of average frame rate at different resolution

![Diagram showing performance difference between Python and C++ with OpenCL](image)

Figure 4.2: Performance difference between Python and C++ with OpenCL
faster than 30 frames per second, so the only significant difference was when
the frame rate came up to max of the cameras capability at 1280x720 pixels.
Here the C++ program performs 8% better than the Python language. When
the live view was enabled differences could seen at all resolutions. Here the
difference was between 4, 3 to 13, 3%
Chapter 5

Conclusions

5.1 Conclusion

The intention of this thesis was to answer the three questions stated in the beginning of the report:

1. How can one detect changes in otherwise static environments?

2. When does an object become static or non-static?

3. How can one filter out static data from non-static data?

The first question is about the foundation of object detection, namely analysing frames captured from some sort of video camera. The basics are as explained earlier a subtraction of a reference background image with the current captured frame. After that, several methods were explained that one can use for detection based on the conditions of the environment it is supposed to be used in. The second question is harder to just give one answer to. This is because some objects are becoming static faster than others based on the environment they are in. In some environment such as airports, object might be considered static after several tens of minutes, while at for example train stations during rush hour, object might need to be detected under a minute to prevent theft of belongings. So this question does not have a single answer since it is based of the situation. The last question can be answered by using different methods. The method I described in this report uses the chapters function that exists in most video containers. To create the chapters, a text file with time stamps describing where the different chapters
begin is created. This text file together with the video file can be merged using the MKVmerge external program to create a video container. When the new video file is played, the user is able to search the video with the chapters, which represent the time of detection. This is just one of many ways to be able to search through a video file for times of detection.

5.2 Future Work

To make this program viable in real applications a few improvements and further developments can be made to suit the need in that specific environment. This program provides a stable ground for object detection which can be extended to the users liking. Since the detection time and the background adaption is based on the frames per second, different machines processes the image at different speed up to the max limit set by the cameras ability to capture frames. To solve this issue and make the program hardware independent a function for adjusting the detection settings based on how fast the machine can process a single image and so calculate the frames per second should be developed. Another extension could be to make the program ”smart” and recognise detected objects to avoid detection the same object twice if it has been moved slightly. This would prevent false alarms and make the program more effective for real surveillance usage. The possibilities for extending and further development of this program is as endless as ones imagination and need for some sort of detection software.
Bibliography


[7] Sen-Ching S. Cheung and Chandrika Kamath "Robust techniques for background subtraction in urban traffic video" Center for Applied Sc-
YingLi Tian, Rogerio Feris, Haowei Liu, Arun Humpapur, and Ming-Ting Sun  "Robust Detection of Abandoned and Removed Objects in Complex Surveillance Videos"


Medha Bhargava, Chia-Chih Chen, M. S. Ryoo, and J. K. Aggarwal  "Detection of Abandoned Objects in Crowded Environments"  Computer and Vision Research Center Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712, USA


Thanarat Horprasert, David Harwood, and Larry S. Davis  "A Statistical Approach for Realtime Robust Background Subtraction and Shadow Detection"  Computer Vision Laboratory University of Maryland College Park MD


[19] https://www.python.org
[22] http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/

**Image reference**

[23] Figure 2.2 on page 7 borrowed from

[24] Figure 2.1 on page 7 borrowed from

[25] Figure 2.6 on page 13 borrowed from

[26] Figure 2.7 on page 15 borrowed from
[5]