Face Detection and Recognition in Video-Streams

Jannik Boll Nielsen

Kongens Lyngby 2010
IMM-B.Sc-2010-14
Abstract

Using the Viola Jones face detection algorithm and Active Appearance Models, it is shown that high success rate face-recognition in video-streams is possible with a relatively low processing time. Limitations in the Active Appearance Model Software by Stegmann et al. [15] forced us discard the original thought of doing a recognition based on parameters from a match, using a general model of the human face. Instead we decided to build an Active Appearance Model representing the subject being searched for, and use this model to do a recognition based on statistical thresholds of deviance.

Tests have proven very high success rates. Detection rate of faces in the video-footage reached 98,38% with only 1,60% erroneous detections, recognition rates per frame reached 66,67% with 0% erroneous and finally the overall sequence recognitions proved a rate of 88,90% while maintaining 0% erroneous recognitions.

The test results clearly indicates that Active Appearance Models are capable of doing high quality face recognitions. Extending the software in order to search for more than one face can easily be done, the computing time however will be doubled whenever the number of Active Appearance Models are being doubled. Had it been possible to do the parameter based recognition, the computing time of the recognition would have remained the same, however recognition of multiple faces would not have any noticeable effect on the computing time.
Both the Viola Jones face detection algorithm and Active Appearance Models have over the last years proven as very robust tools in the domain of image analysis. The speed of the Viola Jones algorithm as well as the flexibility of the Active Appearance Models has already found way to into modern technologies in form of face-detecting cameras and intelligent medical equipment able to recognize elements in the human body. Based on this, the idea of detecting and recognizing human faces using such methods seems very appealing and not as challenging as it has been in the past, using older methods of recognition and detection.

Using the OpenCV C-programming package [6] face-detection will be performed with the Viola Jones algorithm. Recognition of the detected faces will then be performed using the Active Appearance Model software by Stegmann et al. [15]. Since I am studying Electrical Engineering, I have chosen to write all code in the C programming language in order to emphasize, that such software can easily be implemented in a hardware solution. This makes the project very relevant for electrical engineers as well.

I would like to thank my supervisor, Professor Rasmus Larsen, for doing a great job in supervising this project of mine. Also I would like to thank Associate Professor Rasmus Reinhold Paulsen, Assistant Professor Line Katrine Harder Clemmensen and PostDoc Anders Lindbjerg Dahl for their weekly supervision which has also been invaluable and beyond any expectations.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Existing Face Detection Methods</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Existing Face Recognition Methods</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Delimitation</td>
<td>4</td>
</tr>
<tr>
<td><strong>2 Theory</strong></td>
<td>7</td>
</tr>
<tr>
<td>2.1 The Viola Jones Detector</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Continuity Filtering</td>
<td>11</td>
</tr>
<tr>
<td>2.3 The Active Appearance Model</td>
<td>14</td>
</tr>
<tr>
<td>2.4 The Active Appearance Model Search</td>
<td>18</td>
</tr>
<tr>
<td>2.5 Recognition using Active Appearance Models</td>
<td>20</td>
</tr>
<tr>
<td>2.6 The Active Appearance Model Threshold</td>
<td>21</td>
</tr>
<tr>
<td>2.7 OpenCV</td>
<td>23</td>
</tr>
<tr>
<td><strong>3 Implementation</strong></td>
<td>25</td>
</tr>
<tr>
<td>3.1 Limitations of the Active Appearance Model software</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Snapshotting Faces</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Building an Active Appearance Model</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Recognition using Active Appearance Models</td>
<td>28</td>
</tr>
<tr>
<td><strong>4 Results and Discussion</strong></td>
<td>31</td>
</tr>
<tr>
<td>4.1 Detection Results</td>
<td>31</td>
</tr>
<tr>
<td>4.2 Recognition Results</td>
<td>32</td>
</tr>
<tr>
<td>4.3 Discussion</td>
<td>33</td>
</tr>
</tbody>
</table>
## CONTENTS

5 Conclusion ................................................. 37  
  5.1 Conclusion ........................................ 37  
  5.2 Future Work ........................................ 37  

A Test Data .................................................. 39  

B AAM Training-set ......................................... 45  

C Software Manual ......................................... 47  
  C.1 Calling the Software .................................. 47  
  C.2 Changing the Settings ................................. 48  
  C.3 Reading the Output .................................... 48  
  C.4 Known bugs ............................................. 48
List of Figures

1.1 Cameras with face-detectors ........................................... 2
2.1 The Integral Image .................................................. 8
2.2 Using an Integral Image ............................................ 8
2.3 Common features .................................................. 9
2.4 Common features .................................................. 9
2.5 Frontal face ..................................................... 11
2.6 Common features generated from frontal faces ................. 11
2.7 Erroneous Detections ............................................. 12
2.8 Continuity Filter Flow-table ..................................... 14
2.9 Errors after filtering ............................................. 15
2.10 Geometrical facial features ..................................... 16
2.11 AAM Search ................................................... 20
2.12 The $\chi^2$ distribution ......................................... 22
3.1  Snapshot scaling and translation .......................... 27
3.2  Variations in an AAM ................................. 30

4.1  Face Detection Results ................................. 32
4.2  ROC Curve of Face Detections ......................... 33
4.3  Face Recognition Results per frame ................. 34
4.4  Face Recognition Results per sequence ............. 35
4.5  ROC Curve of Face Recognitions .................... 35

B.1  Training images for AAM ............................. 46
List of Tables

A.1 Viola Jones Detections 1 ........................................ 39
A.2 Viola Jones Detections 2 ........................................ 40
A.3 Viola Jones Detections 3 ........................................ 40
A.4 Viola Jones Detections 4 ........................................ 40
A.5 Recognition per frame 1 ........................................ 41
A.6 Recognition per frame 2 ........................................ 41
A.7 Recognition per frame 3 ........................................ 42
A.8 Recognition per video 1 ........................................ 42
A.9 Recognition per video 2 ........................................ 43
A.10 Recognition per video 3 ....................................... 43
A.11 Recognition per video 4 ....................................... 44
Introduction

1.1 Background

A lot of video footage exists in archives, much of this includes people. The footage is however not always properly documented and does therefore often not contain proper information about which people have been filmed. As the contents of video archives is ever increasing, it is needed to find ways of annotating relevant people included in the many footages. To speed up this extensive process, facial recognition and matching, could be applied, using modern computers.

In some cases only one, or a short list of people are needed to be recognized, for example when looking for a certain actor or politician. Here, the issue is not who is there, but rather if a certain person is there. A 'White List' containing the faces of the requested people could be matched with the faces found in video archives, in order to decide if any of these faces are present and if so, who appears where.
Among the algorithms for detecting faces within an image, 3 important ones are the Viola-Jones Object Detection Framework [19], the Schneiderman and Kanade Statistical Detection Approach [11] and the Neural Network-Based Face Detection [4]. These methods all have a high efficiency rate in detecting faces, but only the Viola Jones algorithm has proven fast enough for real-time detections [7].

Schneiderman and Kanade uses an approach where a product of histograms from subsections of an image, are used in order to statistically determine if faces are present. Using multiple statistical templates, multiple orientations of faces can be detected, making the detector more robust. Unfortunately this detector suffers from the low processing speed and is therefore hard to implement in real-time applications.

The Neural Network based detector uses a set of neural networks trained for determining the presence or absence of faces based on pixel intensities. The major disadvantage of this detector, is that the neural networks must learn what is not a face, which is very hard to teach considering how many pixel-intensity combinations are not a face, compared to the number of combinations that are. Furthermore also the speed of this detector is not high enough for real-time applications.

The Viola Jones detector has the advantage, compared to the Neural Network based, that it does not need to know what is not a face. It only needs to know what can be a face. The speed of this detector, is the only of the 3 mentioned that actually allows for real-time detection. This detector will be explained in-depth later in this paper.

Figure 1.1: Both of these cameras automatically detect and focus on faces.

Already the Face Detection algorithms are being implemented in a wide variety of products. Most cameras produced today, have a face detection algorithm im-
implemented to assure proper focus. Recently, also cameras using a smile detector have been presented, assuring happy faces in pictures. In figure 1.1 two cameras from Canon and Sony respectively are shown, both of these cameras use face detection algorithms to detect faces and focus on these.

1.3 Existing Face Recognition Methods

So far, multiple methods of recognizing faces have been developed, some more successful than others. Among the traditional methods are the geometric and photometric approaches. The geometric approach measures the face’s geometric features, such as eyewidth/distance, jaw-size and so forth, while the photometric approach is a view based approach, that extracts a line of statistical, photometric features and evaluates those [16].

Among the methods developed, the most studied are the PCA (Principal Components Analysis, or Eigenface method), the LDA (Linear Discriminant Analysis) and the EBGM (Elastic Bunch Graph Matching) [16].

Recently new approaches to the face recognition problem has appeared. These new approaches involve 3D facial recognition, using 3D models of faces generated by for example stereo-cameras[9], and recognition based on skin texture analysis.

The 3D model approach can achieve very high recognition rates reaching up to 97% success and close to 0% erroneous recognition [5]. The major drawback of this type of recognition is the amount of preparation needed in order to do a match: The face must be recorded in 3D, which is not always possible depending on the application. If one is able to control the recognition environment e.g. a login-facility, this method is definitely a possibility, however if recognition is to be performed on random images, the usage is limited.

The method of recognising using Eigenfaces is based on projecting a face into a subspace spanned by trained Eigenfaces. Based on the subspace-position of the face and a priori knowledge of the subspace-position of the faces being searched for, a statistical evaluation can be performed in order to classify or reject the face as a match [18]. This method has also proven successful with correct recognitions of close to 100%, although unable to classify roughly 40% of the the faces presented [18].

The companies Google and Apple have both lately presented new image organizer software for private use, that enables users to build and search photo-collections based on face recognition. The company L-1 specializes in recognition

\[\text{Google Picasa and Apple iPhoto}\]
systems including face-recognition-based. This company is one of the leading companies in the world when it comes to face-recognition and presents a lot of solutions enabling customers to recognize faces. Some applications even provide real-time recognition [13].

1.4 Problem Statement

This project involves two key subjects.

- Use the Viola Jones algorithm through relevant C-programming packages (OpenCV) and thereby extract faces from a number of given video recordings.
- Use Active Appearance Models software (either API or Standalone) to try and match the faces found, via. the Viola Jones algorithm, with up to 10 images contained in a "White List" in order to decide, whether or not one, or more, of these "White Listed" people appears in the video footage.

While using the Viola Jones face-detector, it is important to be able to distinguish the faces, to a certain degree, in order to decide if the face found is in fact the same as the one in previous frame (but moved slightly), or if it is actually a new face. Without this filtering, an overflow of duplicates would probably appear.

Since the face-detection is done on multiple frames, one could expect that a 'high quality Viola Jones match' would be found at some point. If this is the case, a very high success-threshold could be used, which would likely make the final AAM-White-List-matching easier.

1.5 Delimitation

The Viola Jones algorithm uses a trained classifier cascade in order to detect faces. Training such a cascade is beyond the scope of the project and is in fact a rather extensive project in itself. One must therefore rely on finding an already trained cascade. Cascades are available on many websites on the internet, one example is [12] which provides multiple cascades including frontal face, profile and head + shoulders.
Also building a very general Active Appearance Model is a rather time consuming process, requiring a very large collection of annotated faces containing large variation in appearance. Due to this fact, one must find a compromise between the quality of the Appearance Model and the time spent training it.

Finally, since this is a "proof of concept", little weight is put on the quality of the code and is not considered important for the purpose of this paper.
In this chapter relevant theory concerning the Viola Jones Face Detector and Active Appearance Models will be explained. Initially the Viola Jones Detector algorithm, as well as its cascade of classifiers, will be presented. Subsequently an approach to filter out erroneous detections based on time-space constraints will be presented in the section Continuity Filtering. This theory is the basis of all face detection executed in the recognition software. Afterwards, a theoretical explanation of the Active Appearance Models, including the search and the goodness of fit estimation, is presented. Where the previous sections explains the concepts of detecting faces, these sections concerns theory of recognizing faces from the detections. Finally a brief introduction to the usage of OpenCV is made.

2.1 The Viola Jones Detector

2.1.1 The Algorithm

The Viola Jones Face Detection Algorithm uses a so-called Integral Image to speed up the calculation of rectangular features. The Integral Image resembles
"Summed area tables" and is defined as as the sum of all values left of and above the current coordinate:

\[ ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \]

Where \( ii(x, y) \) is the integral image and \( i(x, y) \) is the original input image.
In figure 2.1 this procedure is illustrated on a 3*3 pixels image. This entire integral image can be calculated in a single pass [19].

As stated, it is very easy to calculate rectangular sums within the input image, using its integral image. This is illustrated in figure 2.2 where the sum of the shaded rectangle, is easily calculated using the values A, B, C and D from the integral image. Thus the sum of the shaded rectangle is given by:

\[ \sum ShadedRectangle = D - (B + C) + A \]

Any given sum can therefore be calculated using only 4 references.
By using the integral image, one can very easily check for features at arbitrary locations within an image, since the calculated sums give a direct indication of the color-intensity at a given area.
Some common features are presented in figure 2.3 [7]. In these features the
2.1 The Viola Jones Detector

Figure 2.3: Common features

required rectangles range from 2 to 4, requiring from 6 to 9 integral image references.

Most of the features presented above are contained within a face, though every feature is so common that you cannot decide whether or not a face is present depending only on that one feature, due to this, these features are called weak classifiers. In order to detect a face properly using these features, Viola-Jones presents the Cascade of Classifiers [19].

This cascade of weak classifiers, very common (but always present in faces) features, tries to rule out if a face is present. It does this by testing sub-windows in the image for more and more specific features, while rejecting windows as soon as they fail a step in the cascade.

This detection cascade is illustrated in figure 2.4.

According to Viola-Jones, a detection window containing features need not be

Figure 2.4: Common features

larger than 24*24 pixels, and even though this seems like a small resolution it actually allows the construction of roughly 160,000 features [7].

The idea is that this cascade of weak classifiers combined, results in a strong classifier, which is able to detect faces with a very high success-rate. Assume a cascade consists of 10 stages with features, each with a detection rate on faces at 0.99 and a false detection rate of 0.5, where $F$ is the overall false
positive rate\(^1\) and \(D\) is the overall detection rate.
The overall detection rate for the cascade can then be calculated as:

\[
D = \prod_{i=1}^{10} 0.99 \\ D = 0.9
\]

While the overall false positive rate gives:

\[
F = \prod_{i=1}^{10} 0.5 \\ F = 0.001
\]

This shows that the combination of a long line of weak classifiers, results in a strong classifier, able of rejecting non-faces very fast, in this case detecting 90% of faces while only erroneously classifying 0.1% of the noise as a face. What is good about this detection method, is the fact that most sub-windows in an image are discarded very quickly and does therefore not require much processing power. At the same time, promising sub-windows are being processed more, allowing a detection of high quality.

\[\text{Theory} \] 

2.1.2 The Cascade

The classifier-cascade used by the Viola Jones Detector is a cascade of weak classifiers generated using the a modified AdaBoost algorithm. When generating such a cascade, an AdaBoost algorithm is fed with literally thousands of images containing faces and again thousands containing no faces [7]. The algorithm is then able to extract common facial-features, expressed as classifiers. Through a thorough training, often taking days of computing power, a classifier-cascade can be so well trained that a Viola Jones detector using this cascade could achieve >90% [7] success rate in positive images\(^2\).

In figure 2.5 an example of a training face is presented. From this and a lot of other faces, common features are extracted. The 2 most dominant pairs of features are shown in 2.6. It can easily be seen that these represent the eyes/forehead contrast, forehead/hair contrast and the nose/eye contrast. It is obvious that these features are all very characteristic for human faces.

\(^1\)The rate at which something is recognized as a face, while it in fact is not
\(^2\)Images that does in fact contain a face
2.2 Continuity Filtering

It can be seen that after implementing the Viola-Jones face detector, even though the detector has a very high detection rate on frontal faces, it still does contain some noise in form of erroneous detections (false positives). An example of this is shown in figure 2.7 where a face detection was computed on a video-frame. Even though the actual face of interest was detected, another, false detection, was also returned.

Since the detected faces are to be processed by a face-recognition algorithm afterwards, the amount of false detection-outputs is needed to be minimized, so that the recognition algorithm is spending as little computation power as possible on irrelevant areas, while searching for face-matches.

In order to solve this, the fact that the software is processing video-streams and not still-images is being used. Many of the erroneous detections are expected to only be instantaneous, in the sense that a minimal change of camera an-
Figure 2.7: The Viola-Jones algorithm successfully detects the face of interest, unfortunately it also detects something that is not a face.

gle or lightening condition would obscure the false detection. By searching for
time-dependant continuity in detection positions and -sizes, one is able to filter
out the majority of the noise the Viola-Jones algorithm produces in a still-image.

An algorithm was designed, enabling this feature.
Pseudo-code is as follows:

1. A 2 dimensional array is created, where the length of the first coordinate is
determined by the amount of detections allowed to be processed per frame
and the length of the second coordinate is determined by the amount of
frames a face is allowed to skip, in order to still be classified as "the same
face as before".

2. Start processing first frame as follows:

   - Detect all faces. if there are no faces, go back to (2) and process next
     frame.
   - For each detected face,
     - add it to the last column in the array.
     - If any previous detection in the array has a size and position
       within threshold: copy its score, increment it by 1, and assign it
       to this detection.
2.2 Continuity Filtering

- Delete the first column and shift all other columns one left. (i.e. increment the time passed)
- Go back to (2) and process next frame.

3. While processing the detections, only current frames within the array containing a score above a certain threshold is then marked/saved/further processed.

The iteration process is illustrated in figure 2.8 where a detection frame-jump of maximum 5 frames is allowed. Whenever a face is detected that is in the proximity of one of the detections in the previous 5 frames, the score of this face is incremented. In this way, the score indicates whether a certain detection has a high or low probability of actually being a face.

One could do some statistical considerations concerning this method of filtering. Assuming that a face-detector has an erroneous detection rate $P_{error}$, one can estimate the probability of the same error appearing multiple times under the assumption that the scene has changed considerably between each sample. If it is assumed that it takes $n$ frames in order for the scene to not be identical to the previous and the detector registers a face $m$ times in a row, then the number of uncorrelated detections will be $\frac{m}{n}$ and the total probability of observing the same error at the same location multiple times, called $P_{total}$, would be:

$$P_{total} = \prod_{1}^{\frac{m}{n}} P_{error} \Leftrightarrow$$

$$P_{total} = (P_{error})^{\frac{m}{n}}$$

In the implementation, a score threshold of 15 detections was set. Assuming, rather crudely, that it takes 5 frames for detections to be uncorrelated, one obtains $\frac{m}{n} = \frac{15}{5} = 3$. With an erroneous detection rate of $P_{error} = 0.1$ from the detector, the total probability of an error appearing 15 times will be:

$$P_{total} = (0.1)^{3} \Leftrightarrow$$

$$P_{total} = 0.001$$

Indicating a 0.1% probability of misclassification. This is however a very crude approximation since frames will always be somewhat correlated. Moreover, the
Figure 2.8: Flow-table showing detections throughout the current and last 5 frames. The detected faces’ x,y positions are plotted as well as the associated score(superscript). The score is incremented when ever position and size(not shown here) equals one of the preceding detections within a certain threshold, here $\approx 10px$

erroneous detections are not just randomly distributed noise, but do have relation to the image, though might not as strong as an actual face.

By using this continuity-filtering algorithm, the erroneous detections are drastically reduced, however there are some that this algorithm does not remove. Tests showed that especially flat textures with certain patterns, such as Zig-Zag wallpapers can trigger the Viola-Jones detection continuously. This limitation however, is caused by the Viola-Jones algorithm is not removable through post processing. An example of this is shown in figure 2.9.

Another approach that could have been used in order to filter out noise, is the Kalman Filter. The filter predicts positions of the detections over time based on a weighted average between models and measurements. It does this by estimating the uncertainty of a detection and adjusting the weighting accordingly. The Kalman Filter is very useful in this domain and is already used in many tracking applications. One major disadvantage in this specific problem dealt with in this paper, is the fact that scenes change in the videos. Therefore one has to register any scene-changes present and reset the Kalman filters accordingly. This makes the implementation of the Kalman Filter rather extensive, and due to the fact that the primarily proposed filter works flawless in this application, the Kalman Filter was not used. Had the footage been continuous as with surveillance cameras the Kalman Filter had been the obvious approach.

### 2.3 The Active Appearance Model

The Active Appearance Model is a model able to represent any object it is trained for, in a compact parametrised form, by describing the variations of the
2.3 The Active Appearance Model

Figure 2.9: Even after filtering for continuity, the software returns a wrong detection with a very high score. When inspecting the pillar, it does make sense that it fools the Viola-Jones detector when comparing to the features that describe a face: Two dark regions separated by a light (eyes and nose), a broad light region above the previous (forehead), etc.

most dominant features within the training set. The model consists of information regarding the variation of shape as well as information regarding the variation of texture intensity.

The shape variations of a training set, are taught to the AAM, by annotating the most important geometrical features describing the objects within the set. If the object to represent was for example a face, then the contour of the eyes, mouth, nose and jaw would be the most distinct features to mark. Using Generalised Procrustes Analysis the set of annotated geometrical features can be aligned in order to remove any translation, scaling or rotation present within the training set’s samples [17] and thereby supply a set containing annotations affected only by their actual variance. Applying Principal Component Analysis to this set, one obtains a way of describing the shape variability in a compact way. In figure 2.10, originally shown in [15], it is illustrated how the sets of shape-annotations are having their scale, rotation and translation variations removed, thereby resulting in a plot containing only the variance of the annotated features.

The texture variations of a training set are found by warping every element of the set, to match the mean-shape of the set found using the GPA explained
above. From this shape-free reference the pixel-intensities are sampled \cite{15}. The collection’s pixel-intensities from the training-set are then also treated with a Principal Component Analysis, supplying yet another compact description of the set, this being the variations of the texture.

From the above two statistical descriptions of a training-set any arbitrary object belonging to the same family\footnote{It would of course not make sense to try and represent a banana using a model trained from a set of human faces.} as the training set, can be approximated by the model.

Let $s$ be the shape of this arbitrary object and let $t$ be the texture of it. Then $s$ and $t$ can be described by:

\begin{align}
    s &= \bar{s} + \Phi_s b_s \\
    t &= \bar{t} + \Phi_t b_t
\end{align}

Where $\bar{s}$ is the mean shape from the training-set and $\bar{t}$ is the mean texture from the training-set. $\Phi$ are matrices build from the column eigenvectors describing respectively the shape and the texture of the set. And finally the $b_s$ and $b_t$ are the Principal Component parameters unique to the object represented by the set.

From (2.3) and (2.4) it can be seen that:

\begin{align}
    b_s &= \Phi_s^T (s - \bar{s}) \\
    b_t &= \Phi_t^T (t - \bar{t})
\end{align}
Instead of having two distinct parametrisations \((b_s \text{ and } b_t)\), these can be combined to form one single vector, \(c\). However in order to obtain this shape-texture combined parametrisation \(c\), a final PCA must be performed in order to accommodate for any correlation there might be between shape and pixel-intensities. This is done in the following:

\(b_s\) and \(b_t\) are combined into \(b\) using a weighting represented by the diagonal matrix \(W_s\).

\[
\begin{align*}
    b &= \begin{bmatrix} W_s b_s \\ b_t \end{bmatrix} = \begin{bmatrix} W_s \Phi_s^T (s - \bar{s}) \\ \Phi_t^T (t - \bar{t}) \end{bmatrix} \\
    &= \begin{bmatrix} \Phi_{c,s} \\ \Phi_{c,t} \end{bmatrix} c
\end{align*}
\]  

(2.7)

Stegmann et al. mentions that the typical weighting is done so the shape Principal Scores \((b_s)\) are weighted by the square root of the ratio between the sums of the texture and shape eigenvalues [15].

Coupling the shape and texture eigenspaces through the third PCA then results in the combined parameters describing variances in shape and texture as well as their common correlation:

\[
\begin{align*}
    b &= \Phi_c c = \begin{bmatrix} \Phi_{c,s} \\ \Phi_{c,t} \end{bmatrix} c \\
    &= \begin{bmatrix} \Phi_{c,s} \\ \Phi_{c,t} \end{bmatrix} W_s^{-1} \Phi_{c,s} c
\end{align*}
\]  

(2.8)

With this, (2.7) and (2.8) can be combined, resulting in a statistical description of the arbitrary object being:

\[
\begin{align*}
    s &= \bar{s} + \Phi_s W_s^{-1} \Phi_{c,s} c \\
    t &= \bar{t} + \Phi_t \Phi_{c,t} c
\end{align*}
\]  

(2.9) (2.10)

Where the object is now described using only one set of parameters, describing the full coupled appearance.

What (2.9) and (2.10) tells, is that adjusting only the values of the vector \(c\), one is able to model any given appearance belonging to the same family as the training-set.

This leads to second aspect of Active Appearance Models: Finding the best match between a model and an underlying image by adjusting the appearance parameters in \(c\). However, before continuing to this matter, a brief comment on how to build an Active Appearance Model is now presented.
2.3.1 Building an Active Appearance Model

In order to extract parameters describing shape- and textural-features, an Active Appearance Model must be built. This model must be trained using a wide variety of faces presenting different lightening conditions, facial expressions and orientations. The model must be as general as possible enabling it to match any arbitrary pose, expression and texture of an unknown face.

M. B. Stegmann presents in his work FAME - A Flexible Appearance Modelling Environment [15] a collection of different annotated faces. With this collection it is possible to train an Active Appearance Model of the human face, containing parameters describing face-positions, textural- and shape-features and expressions, based on the actual variations present within the collection of faces. This can be done using the console-software also available through [15].

The annotation process is rather time consuming due to the fact that it is done manually. Edwards et al. therefore suggests doing this annotation semi- or fully automated, since the environment of the training faces can be fully controlled [17].

2.4 The Active Appearance Model Search

The Active Appearance Model Search is basically how a model is able to fit itself to an underlying image, i.e. how to iterate through the model-parameters in order to obtain the minimum error possible, between model and image. Multiple ways of doing this exists, however only one will be explained here: the Fixed Jacobian Matrix Estimate [17], which is the one used in the AAM software by M. Stegmann and therefore also used in this project [15].

In order to solve this optimization, an error-vector \( r(p) \) is introduced. This vector is an indication of the pixel-intensity-error between the model and the picture below. \( r(p) \) is defined as:

\[
r(p) = t_{image}(p) - t_{model}(p)
\]  \hspace{1cm} (2.11)

Where \( t_{image}(p) \) and \( t_{model}(p) \) are the texture vectors describing respectively the image texture under the model and the model-texture.

If \( p^* \) is a set of parameters in the proximity of the optimum parameter-set: \( p^+ \), the Taylor Expansion of the error is then:
2.4 The Active Appearance Model Search

\[ r(p^* + \delta p) \approx r(p^*) + \frac{\partial r(p^*)}{\partial p} \delta p \quad (2.12) \]

Where the differential component \( \frac{\partial r(p^*)}{\partial p} \) is:

\[ \frac{\partial r(p^*)}{\partial p} = \frac{\partial r}{\partial p} = \begin{bmatrix} \frac{\partial r_1}{\partial p_1} & \cdots & \frac{\partial r_1}{\partial p_Q} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial p_1} & \cdots & \frac{\partial r_M}{\partial p_Q} \end{bmatrix} \quad (2.13) \]

In order to obtain the minimum error \( r \), a \( \delta p \) must be found that fulfils:

\[ \arg\min_{\delta p} ||r(p^* - \delta p)|| \quad (2.14) \]

Using the approximation from equation (2.12), a least-squares solution can be found:

\[ \delta p = -Rr(p) \quad (2.15) \]

Where the matrix \( R \) is given by:

\[ R = \left( \frac{\partial r}{\partial p} \cdot \frac{\partial r}{\partial p}^T \right)^{-1} \frac{\partial r}{\partial p} \cdot \frac{\partial r}{\partial p}^T \quad (2.16) \]

Calculating the Jacobian Matrix from equation (2.13) is a rather consuming process, however according to [17], due to the fact that the AAM works in a normalized reference frame, this component is approximately constant. Therefore one only needs to calculate this matrix once, instead of after every iteration, speeding up the search process greatly.

\[ \frac{\partial r(p^*)}{\partial p} \approx \frac{\partial r(p^+)}{\partial p} \quad (2.17) \]
Not only does [17] state that $\frac{\partial r(p^\ast)}{\partial p}$ is approximately constant, they also assume, rather crudely [14], that $\frac{\partial r(p^\ast)}{\partial p}$ is constant throughout the entire training-set, allowing one to compute this matrix only once during the training process and thereby increasing the search speed even further.

The iterative search explained above, is illustrated in figure 2.11 where the model tries to fit to the underlying image using 9 iterations.

![Figure 2.11: An AAM search. Using least squares method, the optimal set of parameters are found resulting in the minimum error possible.](image)

2.5 Recognition using Active Appearance Models

G.J. Edwards et al. presents in [3] a way of extracting identity features from a face using a very general Active Appearance Model trained from a very large training-set. Initially Edwards et al. models the AAM from a large collection of faces. In this model, Edwards et al. suggests there must be a combination of parameters describing the inter-class variation\(^4\) and another combination describing the intra-class variation\(^5\). Knowing the location of the inter-class parameters, one can do an AAM search

\(^4\)The identity of the face, these variations distinguish one identity from another.

\(^5\)These variations are changes within any identity, such as expression, pose and lighting conditions.
on a face and extract these values. Thereby obtaining a vector containing an unique ID for the face. This ID should theoretically be the same for any foto of the face.

Another approach to recognize faces, is by modelling a specific AAM to represent the face being searched for. The AAM is then trained from a collection of images containing the face in question. The collection need not be as large as when making a general face model, but must contain common variance to the face such as expression, pose, lighting conditions and maybe even age-related variation. Such a specific model will be able to match very well to images containing the correct face, and based on an overall goodness of fit estimate, one can determine if a face represented by the model is present or not.

2.6 The Active Appearance Model Threshold

When the Active Appearance Software by Mikkel Stegmann [15] has finished its search, it returns a Mahalanobis Distance measure to indicate the quality of the match. This distance measure is calculated in the the truncated principal component space and is defined as:

$$D_M = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

(2.18)

Where $x = (x_1, x_2, x_3, ..., x_n)^T$ is a set of values describing a point in an n-dimensional space, being compared to a point-distribution with a mean of $\mu = (\mu_1, \mu_2, \mu_3, ..., \mu_n)^T$ and a covariance matrix $\Sigma$, which is a diagonal matrix with the eigenvalues of the principal components in the diagonal.

Assuming that the principal component scores follows a normal distribution, the square the Mahalanobis distance-measure is a sum of squares of independent identically distributed variables following a normal distribution: $N(0,1)$. The square of the Mahalanobis distance must therefore follow a $\chi^2$ distribution:

$$D_M^2 = (x - \mu)^T \Sigma^{-1} (x - \mu) \in \chi^2(n)$$

(2.19)

Stegmann chooses to normalize this distance by dividing with the number of modes, $n$. 

From this, a statistical estimate of a threshold, distinguishing between members and non-members of the Active Appearance Model can be computed by finding an appropriate $\chi$ and thereby calculating the corresponding Mahalanobis Distance:

$$D^2_{threshold} = \chi^2(\alpha(v)) \Leftrightarrow$$

$$D_{threshold} = \sqrt{\chi^2_v(\alpha(v))} = \chi_{\alpha(v)}$$

Where $v = n$ is the number of freedom degrees\(^6\) in the Active Appearance Model containing $n$ modes, and $\alpha$ is the probability of the right side of the $\chi^2$-distribution as shown in figure 2.12.

As an example, the Active Appearance Model describing the actor Jarl Friis Mikkelsen, presented later in this paper, has 14 parameters which gives a total of 14 degrees of freedom. In order to be able to classify 95\% of the images of Jarl Friis Mikkelsen, $\chi^2$ is determined using $\alpha = 0.05$ and $v = 14$:

---

\(^6\)Due to the fact that $\mu$ and $\Sigma$ are based on a large and independent training-set
\[ D_{\text{threshold}}^2 = \chi^2_{0.05}(14) \iff (2.23) \]
\[ D_{\text{threshold}}^2 = 23.68 \iff (2.24) \]
\[ D_{S,\text{threshold}}^2 = \frac{D_{\text{threshold}}^2}{n} = \frac{23.68}{14} = 1.69 \iff (2.25) \]
\[ D_{S,\text{threshold}} = 1.30 \iff (2.26) \]

This concludes that theoretically at least 95% of images taken of the subject will be classified as true, if a Mahalanobis-threshold of 1.30 is used to distinguish. The result of these calculations corresponds nicely to the best threshold found empirically: 1.2. The threshold was determined by manually tuning in intervals of 0.1 until the best results were found.

## 2.7 OpenCV

The procedure of detecting faces using the Viola-Jones algorithm is basically a straightforward task in OpenCV. Among the many other functions, OpenCV comes with the `cvHaarDetectObjects` that uses a trained cascade of classifiers to detect an object within the image.

```c
CvSeq* cvHaarDetectObjects(
    const CvArr* image,
    CvHaarClassifierCascade* cascade,
    CvMemStorage* storage,
    double scale_factor=1.1,
    int min_neighbors=3,
    int flags=0,
    CvSize min_size=cvSize(0,0)
);
```

The function requires an image, a trained cascade containing Haar-Like Features\(^7\) and allocated memory. The last arguments are optional: `Scale factor` indicates how much the size of the detection-window should increase after every pass through the image. The `neighbors` argument groups clusters of detections

\(^7\)Haar-Like features are the rectangular intensity features described while introducing the Viola-Jones algorithm. The features can quickly be calculated in an image using the associated integral image as a lookup table.
greater than a given threshold, into one single detection, clusters below the threshold are discarded. The flags argument only has one option at this moment which is 0, enabling a Canny Edge Detector to discard some areas of the image that are unlikely to contain the searched object. Finally the size sets the minimum size the detector should search for objects at.

When the detection function has been computed it returns a sequence of values, that contain information about the various detections made, including total count, specific positions and specific widths and heights.

It is obvious that this OpenCV function does the exact operation described by Viola-Jones: Scans a range of sub windows using a cascade of weak classifiers, in order to detect a certain object. Therefore, given a well trained classifier cascade, this detector can be used to fast and accurately detect faces as proposed by Viola-Jones [19].

Before using the cvHaarDetectObjects function, some preparations has to made on the image in order to maximize the success rate. Firstly the input image must be converted into grey scale, since the features are based on light and dark areas. Furthermore Gary Bradski and Adrian Kaehler [1] recommends equalizing the histogram in order to level out the light intensities in the image.

After these initial conversions an image is ready for processing. Depending on the amount of possible faces and the dimensions of the image, the processing time can vary a lot. In non-HD video footage, where the resolution is fairly low (VHS = 350x480px) this detection process can be computed in real-time on a modern computer.
In this chapter the reflections made, as well as the obstructions faced, during the implementation are presented. Due to an unforeseen limitation of the Active Appearance Model software, some new approaches to the recognition had to be made. This is explained in the first section. The following sections concern how one, using the previously explained theory, in practice snapshots faces from video-footage, builds the Active Appearance Model and recognizes faces using the model.

3.1 Limitations of the Active Appearance Model software

It was expected that the Active Appearance Model software presented by Stegman et al. [15] was able to output a vector containing the resulting parameters from the AAM search. This however was not the case and due the fact that accessing the API and merging it with this software was beyond the scope of this project, another approach to face-recognition using Active Appearance Models was taken. Instead it was decided to base the recognition on subject-specific models as explained in the previous chapter, knowing that the speed of the recognition-software would be affected.
3.2 Snapshotting Faces

When processing the video-footage and detecting faces, the software must save key-snapshots in order to supply the following Active Appearance Model with the best information available. Basically this means, that every face detected in a video-sequence must be saved. However, due to the instability of a Viola Jones Detector, this would quickly result in an overflow of snapshots.

A way to optimize the snapshot-process, is by utilising the continuity filtering presented earlier. Theoretically this would allow the software to only take one snapshot per person present in the footage. Due to some unavoidable breaks in the continuity, such as scene changes or faces turned away from the camera, this is however not possible and one must expect a varying amount of snapshots per person. Even with these breaks, the method still minimizes the amount of snapshots taken greatly though.

Another issue is the fact that the Viola Jones Detector does not return a quality estimate of the face detected. This means that the snapshots saved can be of very poor quality, and very possibly another and much better face could appear later within the same stream of continuous faces. Therefore one can not only rely on taking one single snapshot when a face is detected, but must however continue taking snapshots in certain intervals throughout the span of the detection, in order to maximize the chance of getting the optimum snapshot of the face.

Finally one must secure consistency in the size of snapshots saved, the size of faces within these snapshots, as well as the position of the faces within the snapshots. This must be done in order to optimize the initialisation the AAM. Luckily the Viola Jones Detector solves two of these problems: The size of faces and the position of faces. Due to the nature of the detector, it will always return faces with almost same size and position relative to the window of the detector. With this consistency secured, one can scale and translate the detection window freely in order to obtain a snapshot fulfilling the requirements of the AAM (e.g. the jaw must be visible). This is illustrated in fig. 3.1. The last thing to ensure, is the size of the saved snapshots. This is easily done by defining a certain size, that one wishes all snapshots to be saved in, and then resize snapshots accordingly before they are saved.

When implementing the function taking snapshots as explained above, one must consider a range of parameters: In what interval should the detector save snapshots in continuous face-streams? How much must the Viola Jones detection be scaled and translated? What dimensions should the saved snapshots have? And finally, in what format should it be saved?

In this implementation, the user is able to set a snapshotting interval as well as a cap for the amount of snapshots taken. The default values are snapshots every 15th detection (per continuous face) with a maximum of 5 snapshots. In order to increase recognition-rates the interval and cap can be increased, however, this is
3.2 Snapshotting Faces

Figure 3.1: To the left is the raw detection of the Viola Jones Detector. To the right, the faces have been scaled and translated so that all information needed by an AAM is present. Lines have been drawn to illustrate the consistency in scale and position of the detector.

The decision about what dimensions the saved snapshot should have, mostly depends on the size of the images used to train the Active Appearance Model, i.e. there is no need to save images with higher resolution than the one provided by the AAM, on the other hand, one should provide the AAM with images that contain enough information, to be able to recognize the face. In this project, a size of 200x200 pixels is used, mostly due to the fact that this is roughly the resolution of the faces in the footage.

The format of the saved snapshot basically depends on what formats the AAM processes best. Here, the AAM console interface only takes .BMP formats, therefore all snapshots are saved as bitmaps.
3.3 Building an Active Appearance Model

In order to train an Active Appearance Model representing the actor Jarl Friis Mikkelsen, a collection of different images of the actor has to be made. All these images must show the actor’s face in different expressions, directions, lighting conditions, as well as additional facial features such as moustache etc. This is done in order to ensure that the model is able to match any expression or appearance the actor could be expected to have.

In the final Active Appearance Model a total of 29 images was used. From these 29 images, 18 were snapshotted from a piece of video-footage containing the actor and 11 were found on Google. It was very hard to find proper images using Google, since most images were very biased in pose and expression. On the contrary, it was very easy to obtain good images of the actor’s face from the video-footage, since a single run of the Viola Jones detector could output a very large collection of faces, from which proper images could be hand picked. The second approach is therefore recommended to anyone interested in replication this recognition-method.

The most time-consuming element in the building-process, was the hand annotation of the faces. Could this process be automated as Edwards et al. suggests [17], a lot of man-hours could be saved and the recognition software would be more or less fully automatic.

In figure 3.2 the model is presented, where its 3 most significant parameters are being changed.

3.4 Recognition using Active Appearance Models

In the initial recognition tries, a small model only consisting of 15 images was used to recognize the actor. The recognition proved possible with a Mahalanobis Distance Threshold of 2.0. Unfortunately a lot of misclassifications appeared due to the rather high distance threshold. Extending the model with additional images representing the actor improved the recognition considerably. The distance threshold could be lowered to 1.3 which filtered out most of the wrong recognitions. This indicates that the more trained model one has, the lower threshold can be used. However, one must always expect that some false positives can appear. For persons looking very much like the actor, the software will probably not be able to differentiate.

Another consideration to do, is whether to average out the Mahalanobis distances per sequence or simply classify using the lowest distance found. Theoretically the model should only fit perfect to the subject it is trained for thereby
resulting in a very low Mahalanobis distance, but due to different sorts of noise in video-footage, one could expect that a multiple snapshots of the subject would be obscured by the noise and would in fact pull the distance-average above the distance-threshold. Classification based on lowest score is therefore a good approach and is implemented as the standard classification method in this software. It can of course be changed to be based on the average-distance instead if the user wishes this.

One major issue at this point, is the fact that the AAM-search is rather slow. An average search on a 200x200 pixel image takes 1.5 seconds. When using the default snapshotting settings in the software, there are roughly taken 60 snapshots per minute in interview-like footage. This results in a rather long processing time afterwards $\approx 1.5$ minutes of processing per minute of video-footage. Since the processing time by the AAM-search is probably not very optimizable, the major improvements could probably instead be found in the way the snapshots are taken. Lowering the limit to the amount of snapshots taken of a continuous face or increasing the interval between every snapshot, could be one way forward. However, every time the amount of snapshots is decreased, the odds of having a good quality face-image is also decreased. Therefore one should be careful not to decrease the amount of snapshots too much.
Figure 3.2: Screenshot of the Active Appearance Model describing the Danish actor Jarls Friis Mikkelsen. The screenshots shows the mean-face (upper left) and 3 versions, where 3 of the most significant parameters have been changed 1 standard deviation.
In this chapter the performance of the recognition software will be described in form of test results. Initially the detection performance will be presented and subsequently the recognition performance. Finally will the results be discussed.

4.1 Detection Results

In order to test the performance of the Viola Jones Detector with the Continuity Filter applied a set of sequences were picked containing a total of 700 frames. The face detection results are presented in figure 4.1, where it can be seen clearly, that the filter radically reduces the false detections while maintaining a very high detection rate.

In order to get an indication of what continuity threshold should be used, a ROC curve is also presented in figure 4.2. This curve is generated from the same collection of data and indicates that the optimum threshold lies around 15.

Not only has the detector and filter proven very high detection rates, but also the speed of the detector and filter combined is mentionable. Even with the filter applied, the detector is still able to process video-streams in real-time.
3.2 Results and Discussion

Figure 4.1: It can be seen that the Viola Jones detector yields very high detection rate (100%). Unfortunately it also returns an unwanted amount of erroneous detections. Increasing the Continuity Score Threshold decreases the false detections decently but at the expense of true detections.

and even faster, allowing the user to monitor the detection process real-time and making the software more user-friendly. Unfortunately one cannot increase this threshold infinitely. Whenever the score threshold is increased, the risk is increased that a face present in a piece of video-footage will not reach this score before disappearing, this would result in a lot of misses and one must therefore be careful when increasing this threshold.

4.2 Recognition Results

While testing the recognition software performance a total of 1000 snapshots of faces was collected from varying video-footage. Within this collection, 200 of the snapshots contained the actor that the recognition software was trained for (i.e. the actor the AAM represents). The software processed all the 1000 snapshots multiple times using varying Mahalanobis Distance Thresholds, and based on the thresholds determined if the actor was present or not. The results of these tests proved relatively high per-face/frame recognitions as shown in figure 4.3.
Subsequently 15 videos were processed using the complete piece of software, including the continuity filtered Viola Jones detector and the Active Appearance Model matching. From these 15 videos, the actor was present in 9. All videos contained many different people ($\approx 60$ unique face sequences within the videos) and also contained large variation in the actor’s appearance. Even with this large appearance-variation and many faces not being the actor, the recognition software proved very successful. In figure 4.4 the success rates per video-sequence are shown as a function of the Mahalanobis Distance Threshold.

Finally two ROC curves describing the per-frame recognitions and the per-face recognitions respectively are shown in figure 4.5. Usually one would look for the best compromise between true recognitions and false recognitions, however due to the fact that the software should rather have a low recognition rate than have any false-recognitions it is important that a suitable low threshold is selected. Therefore one should according to curve number 2 select a threshold of 1.3. Using this threshold yields a very high recognition rate of $\approx 90\%$ and thereby proving that recognition using Active Appearance Models is in fact achievable.

4.3 Discussion

Results from above clearly indicates high success rates of both detections and recognitions. As stated in chapter 3, improving the Active Appearance Model would most likely enable one to lower the Mahalanobis Distance Threshold even...
Figure 4.3: This figure illustrates the success rates of recognition per frame. Increasing the distance threshold allows for higher recognition rate but at the expense of increased erroneous recognition rate.
4.3 Discussion

Figure 4.4: Even though the per-frame recognition rate is fairly low, it yields much higher recognition rate when integrating over sequences. Tests proved upto 88.9% recognition rate without any erroneous detections.

Figure 4.5: The first ROC Curve shows the correlation between correct and erroneous detections per frame depending on the Mahalanobis Distance. The second ROC curve show the correlation per sequence. Even though the first curve suggests the best compromise is a distance of 1.37, one must keep in mind that when in the domain of recognition, no false-positives are tolerated. The second figure clearly indicates that a Mahalanobis Distance of 1.30 fulfils this requirement, allowing ≈90% recognition without any false-positives.
further, as well as allowing better matching of images containing the actor. This would increase success rates even further while keeping the false recognitions at zero and is therefore a good way of improving the software.

Even though the recognition results looks promising, one must keep in mind that the statistics are based on a fairly small set of data. In order to properly test this software, even more sample videos of the actor are required as well as footage not containing him. The charts presented in this chapter are therefore not fully trustworthy and should only be compared to the data that was available.

The speed of the AAM software is one major disadvantage at this point. Although the actual search algorithm is not optimizable, the fact that the search is performed on all snapshots of a face-sequence, and not stopped whenever a score is lower than the threshold, is an issue. This is caused by the AAM software that does not return any Mahalanobis-distances before all images have been processed. If one merged the AAM API into this software, much more specific AAM-calls could be performed, allowing the searches to stop whenever a match is found and thereby lowering the processing time greatly. Usage of the API would also allow for parameter based recognition as previously presented. With this said, one must keep in mind that not all footage is as face-intense as the footage that has been tested here. All of this footage consisted of interview-like material with a constant presence of faces. In many scenarios much less face-intensity will probably be present, resulting in less processing time by the software. Also as processing power of computers is constantly increasing, one could expect that this matter is not any issue in just a few years.

Implementation of the Kalman Filter combined with a scene-change-detector might also help reduce some of the splitting of face-sequences happening whenever a face-tracking is lost due to for example head-turning or a hand covering the face. This would likely reduce the amount of snapshots to be processed and thereby reduce the processing time even further.

Finally it must be mentioned, as it has been briefly in the Implementation Chapter, that the manual annotation of faces for the Active Appearance Model is a very time-consuming process. Semi- or full-automation of this process would be a major improvement. An approach to this, could be building a super-general face model from a very large training set. Letting this very flexible model fit onto a collection of images describing the subject for the recognition-AAM would yield automatic annotation and allow for automatic building of AAMs, by only supplying a collection of faces. Of course this requires the construction of a massive AAM, but this only needs to be done a single time. Moreover the model can be built in steps allowing it to aid its own training-process.
Conclusion

5.1 Conclusion

Using the OpenCV C-programming package, a continuity filtered Viola Jones face-detector was successfully implemented. The detector obtained a success rate of 98% with a reduction of erroneous detections from 30% down to 1.6%. We thereby directly proved that continuity filtering is able to remove the majority of all noise present in the Viola Jones detector.

Subsequently we successfully proved that Active Appearance Models are very applicable in the domain of face-recognition and that recognition rates close to 90% are achievable. Even though our implementation only contained 1 Active Appearance Model, the concept allows for the use of multiple models in order to do white-list matches as indicated in the problem statement.

5.2 Future Work

As stated in previous chapter, an integration of the AAM framework would allow for even faster recognition as well as open up for the possibility of doing parameter-based recognition as suggested by Edwards et al. [3]. Therefore this
is an obvious place to optimize this solution. Another upgrade mentioned is
the automatization of the AAM training, if this was implemented, the software
would become much more user-friendly and would not require any preparations
before recognition.
Aside from the above examples of improvements, one final obvious optimization
is simply optimizing the already written code.
Appendix A

Test Data

In tables A.1, A.2, A.3 and A.4 test results concerning the Viola Jones Face Detector are presented. In these tests the continuity Score Threshold was increased. Afterwards in tables A.5, A.6 and A.7 the test results from per-face recognitions are presented. In these tests the Mahalanobis Distance Threshold was increased. Finally in tables A.8, A.9, A.10 and A.11 the overall recognition test results are presented. Also here the Mahalanobis Distance Threshold was increased.

Table A.1: Viola Jones Detections, part 1

<table>
<thead>
<tr>
<th>Minimum Score:</th>
<th>1</th>
<th>Minimum Score:</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faces found</td>
<td>517</td>
<td>Faces found</td>
<td>476</td>
</tr>
<tr>
<td>Faces not found</td>
<td>0</td>
<td>Faces not found</td>
<td>1</td>
</tr>
<tr>
<td>Wrong detections</td>
<td>154</td>
<td>Wrong detections</td>
<td>72</td>
</tr>
<tr>
<td>Total detections</td>
<td>671</td>
<td>Total detections</td>
<td>549</td>
</tr>
<tr>
<td>Total faces</td>
<td>517</td>
<td>Total faces</td>
<td>477</td>
</tr>
<tr>
<td>Hit %:</td>
<td>100,0%</td>
<td>Hit %:</td>
<td>99,8%</td>
</tr>
<tr>
<td>Error %:</td>
<td>23,0%</td>
<td>Error %:</td>
<td>13,1%</td>
</tr>
</tbody>
</table>
Table A.2: Viola Jones Detections, part 2

<table>
<thead>
<tr>
<th>Minimum Score: 5</th>
<th>Minimum Score: 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faces found</td>
<td>444</td>
</tr>
<tr>
<td>Faces not found:</td>
<td>1</td>
</tr>
<tr>
<td>Wrong detections:</td>
<td>29</td>
</tr>
<tr>
<td>Total detections:</td>
<td>474</td>
</tr>
<tr>
<td>Total faces:</td>
<td>445</td>
</tr>
<tr>
<td>Hit %:</td>
<td>99,8%</td>
</tr>
<tr>
<td>Error %:</td>
<td>6,1%</td>
</tr>
</tbody>
</table>

Table A.3: Viola Jones Detections, part 3

<table>
<thead>
<tr>
<th>Minimum Score: 10</th>
<th>Minimum Score: 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faces found</td>
<td>347</td>
</tr>
<tr>
<td>Faces not found:</td>
<td>4</td>
</tr>
<tr>
<td>Wrong detections:</td>
<td>14</td>
</tr>
<tr>
<td>Total detections:</td>
<td>365</td>
</tr>
<tr>
<td>Total faces:</td>
<td>351</td>
</tr>
<tr>
<td>Hit %:</td>
<td>98,9%</td>
</tr>
<tr>
<td>Error %:</td>
<td>3,8%</td>
</tr>
</tbody>
</table>

Table A.4: Viola Jones Detections, part 4

<table>
<thead>
<tr>
<th>Minimum Score: 20</th>
<th>Minimum Score: 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faces found</td>
<td>244</td>
</tr>
<tr>
<td>Faces not found:</td>
<td>15</td>
</tr>
<tr>
<td>Wrong detections:</td>
<td>3</td>
</tr>
<tr>
<td>Total detections:</td>
<td>262</td>
</tr>
<tr>
<td>Total faces:</td>
<td>259</td>
</tr>
<tr>
<td>Hit %:</td>
<td>94,2%</td>
</tr>
<tr>
<td>Error %:</td>
<td>1,1%</td>
</tr>
</tbody>
</table>
Table A.5: Recognition per frame, part 1

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Excluded %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>45,00%</td>
<td>13,00%</td>
</tr>
<tr>
<td>Number of Modes:</td>
<td>14,00</td>
<td>14,00</td>
</tr>
<tr>
<td>Mahalanobis Distance:</td>
<td>1,00</td>
<td>1,20</td>
</tr>
<tr>
<td>Stats:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total TrueFalse:</td>
<td>418,00</td>
<td>418,00</td>
</tr>
<tr>
<td>Total TrueTrue:</td>
<td>26,00</td>
<td>113,00</td>
</tr>
<tr>
<td>Total FalseTrue:</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>Total FalseFalse:</td>
<td>178,00</td>
<td>91,00</td>
</tr>
<tr>
<td>Times present:</td>
<td>204,00</td>
<td>204,00</td>
</tr>
<tr>
<td>Times not present:</td>
<td>418,00</td>
<td>418,00</td>
</tr>
<tr>
<td>Hit %:</td>
<td>12,7%</td>
<td>55,4%</td>
</tr>
<tr>
<td>Erroneous Detection %:</td>
<td>0,0%</td>
<td>0,0%</td>
</tr>
</tbody>
</table>

Table A.6: Recognition per frame, part 2

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Excluded %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5,00%</td>
<td>2,50%</td>
</tr>
<tr>
<td>Number of Modes:</td>
<td>14,00</td>
<td>14,00</td>
</tr>
<tr>
<td>Mahalanobis Distance:</td>
<td>1,30</td>
<td>1,37</td>
</tr>
<tr>
<td>Stats:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total TrueFalse:</td>
<td>418,00</td>
<td>373,00</td>
</tr>
<tr>
<td>Total TrueTrue:</td>
<td>136,00</td>
<td>153,00</td>
</tr>
<tr>
<td>Total FalseTrue:</td>
<td>0,00</td>
<td>45,00</td>
</tr>
<tr>
<td>Total FalseFalse:</td>
<td>68,00</td>
<td>31,00</td>
</tr>
<tr>
<td>Times present:</td>
<td>204,00</td>
<td>184,00</td>
</tr>
<tr>
<td>Times not present:</td>
<td>418,00</td>
<td>418,00</td>
</tr>
<tr>
<td>Hit %:</td>
<td>66,7%</td>
<td>83,2%</td>
</tr>
<tr>
<td>Erroneous Detection %:</td>
<td>0,0%</td>
<td>10,8%</td>
</tr>
</tbody>
</table>
Table A.7: Recognition per frame, part 3

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Excluded %</th>
<th>Number of Modes: 14,00</th>
<th>Number of Modes: 14,00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mahalanobis Distance:</td>
<td>1,55</td>
<td>Mahalanobis Distance:</td>
<td>1,98</td>
<td></td>
</tr>
<tr>
<td>Stats:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total TrueFalse:</td>
<td>229,00</td>
<td>Total TrueFalse:</td>
<td>41,00</td>
<td></td>
</tr>
<tr>
<td>Total TrueTrue:</td>
<td>180,00</td>
<td>Total TrueTrue:</td>
<td>183,00</td>
<td></td>
</tr>
<tr>
<td>Total FalseTrue:</td>
<td>281,00</td>
<td>Total FalseTrue:</td>
<td>469,00</td>
<td></td>
</tr>
<tr>
<td>Total FalseFalse:</td>
<td>4,00</td>
<td>Total FalseFalse:</td>
<td>1,00</td>
<td></td>
</tr>
<tr>
<td>Times present:</td>
<td>184,00</td>
<td>Times present:</td>
<td>184,00</td>
<td></td>
</tr>
<tr>
<td>Times not present:</td>
<td>510,00</td>
<td>Times not present:</td>
<td>510,00</td>
<td></td>
</tr>
<tr>
<td>Hit %:</td>
<td>97,8%</td>
<td>Hit %:</td>
<td>99,5%</td>
<td></td>
</tr>
<tr>
<td>Erroneous Detection %:</td>
<td>55,1%</td>
<td>Erroneous Detection %:</td>
<td>92,0%</td>
<td></td>
</tr>
</tbody>
</table>

Table A.8: Recognition per video-sequence, part 1

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Excluded %</th>
<th>Number of Modes: 14,00</th>
<th>Number of Modes: 14,00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mahalanobis Distance:</td>
<td>0,50</td>
<td>Mahalanobis Distance:</td>
<td>0,70</td>
<td></td>
</tr>
<tr>
<td>Stats:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total TrueTrue:</td>
<td>0,00</td>
<td>Total TrueTrue:</td>
<td>4,00</td>
<td></td>
</tr>
<tr>
<td>Total TrueFalse:</td>
<td>7,00</td>
<td>Total TrueFalse:</td>
<td>7,00</td>
<td></td>
</tr>
<tr>
<td>Total FalseTrue:</td>
<td>0,00</td>
<td>Total FalseTrue:</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>Total FalseFalse:</td>
<td>9,00</td>
<td>Total FalseFalse:</td>
<td>5,00</td>
<td></td>
</tr>
<tr>
<td>Times present:</td>
<td>9,00</td>
<td>Times present:</td>
<td>9,00</td>
<td></td>
</tr>
<tr>
<td>Times not present:</td>
<td>7,00</td>
<td>Times not present:</td>
<td>7,00</td>
<td></td>
</tr>
<tr>
<td>Hit %:</td>
<td>0,0%</td>
<td>Hit %:</td>
<td>44,4%</td>
<td></td>
</tr>
<tr>
<td>Erroneous Detection %:</td>
<td>0,0%</td>
<td>Erroneous Detection %:</td>
<td>0,0%</td>
<td></td>
</tr>
</tbody>
</table>
Table A.9: Recognition per video-sequence, part 2

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Number of Modes</th>
<th>Mahalanobis Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65,00%</td>
<td>14,00</td>
<td>0,90</td>
</tr>
<tr>
<td></td>
<td>26,00%</td>
<td>14,00</td>
<td>1,10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Stats:</th>
<th>Stats:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total TrueTrue</td>
<td>6,00</td>
<td>Total TrueTrue</td>
</tr>
<tr>
<td>Total TrueFalse</td>
<td>7,00</td>
<td>Total TrueFalse</td>
</tr>
<tr>
<td>Total FalseTrue</td>
<td>0,00</td>
<td>Total FalseTrue</td>
</tr>
<tr>
<td>Total FalseFalse</td>
<td>3,00</td>
<td>Total FalseFalse</td>
</tr>
<tr>
<td>Times present</td>
<td>9,00</td>
<td>Times present</td>
</tr>
<tr>
<td>Times not present</td>
<td>7,00</td>
<td>Times not present</td>
</tr>
<tr>
<td>Hit %</td>
<td>66,7%</td>
<td>Hit %</td>
</tr>
<tr>
<td>Erroneous Detection %</td>
<td>0,0%</td>
<td>Erroneous Detection %</td>
</tr>
</tbody>
</table>

Table A.10: Recognition per video-sequence, part 3

<table>
<thead>
<tr>
<th></th>
<th>Excluded %</th>
<th>Number of Modes</th>
<th>Mahalanobis Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5,00%</td>
<td>14,00</td>
<td>1,30</td>
</tr>
<tr>
<td></td>
<td>0,50%</td>
<td>14,00</td>
<td>1,50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Stats:</th>
<th>Stats:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total TrueTrue</td>
<td>8,00</td>
<td>Total TrueTrue</td>
</tr>
<tr>
<td>Total TrueFalse</td>
<td>7,00</td>
<td>Total TrueFalse</td>
</tr>
<tr>
<td>Total FalseTrue</td>
<td>0,00</td>
<td>Total FalseTrue</td>
</tr>
<tr>
<td>Total FalseFalse</td>
<td>1,00</td>
<td>Total FalseFalse</td>
</tr>
<tr>
<td>Times present</td>
<td>9,00</td>
<td>Times present</td>
</tr>
<tr>
<td>Times not present</td>
<td>7,00</td>
<td>Times not present</td>
</tr>
<tr>
<td>Hit %</td>
<td>88,9%</td>
<td>Hit %</td>
</tr>
<tr>
<td>Erroneous Detection %</td>
<td>0,0%</td>
<td>Erroneous Detection %</td>
</tr>
</tbody>
</table>
Table A.11: Recognition per video-sequence, part 4

<table>
<thead>
<tr>
<th></th>
<th>Excluded % 0.02%</th>
<th>Excluded % 0.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Modes:</td>
<td>14,00</td>
<td>14,00</td>
</tr>
<tr>
<td>Mahalanobis Distance:</td>
<td>1,70</td>
<td>1,90</td>
</tr>
<tr>
<td>Stats:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total TrueTrue:</td>
<td>9,00</td>
<td>Total TrueTrue:</td>
</tr>
<tr>
<td>Total TrueFalse:</td>
<td>5,00</td>
<td>Total TrueFalse:</td>
</tr>
<tr>
<td>Total FalseTrue:</td>
<td>3,00</td>
<td>Total FalseTrue:</td>
</tr>
<tr>
<td>Total FalseFalse:</td>
<td>0,00</td>
<td>Total FalseFalse:</td>
</tr>
<tr>
<td>Times present:</td>
<td>9,00</td>
<td>Times present:</td>
</tr>
<tr>
<td>Times not present:</td>
<td>8,00</td>
<td>Times not present:</td>
</tr>
<tr>
<td>Hit %:</td>
<td>100,0%</td>
<td>Hit %:</td>
</tr>
<tr>
<td>Erroneous Detection %:</td>
<td>37,5%</td>
<td>Erroneous Detection %: 62,5%</td>
</tr>
</tbody>
</table>
Appendix B

AAM Training-set

Figure B.1 presents the 29 images that were used in order to built the Appearance Model Representing the Danish actor Jarl Friis Mikkelsen.
Figure B.1: This set, consisting of 29 images, was used in order to train an Active Appearance Model for the Danish actor Jarl Friis Mikkelsen.
Appendix C

Software Manual

The software can be downloaded from:

http://www.Boll-Nielsen.dk/FaceRecognition.zip

This zip file contains the full software including C-code and a sample-video.

For the software to build Active Appearance Models as well as the full AAM Framework, please visit:

http://www.imm.dtu.dk/-aam/

C.1 Calling the Software

There are two ways of calling the software. The first and most intuitive being simply running the Detector.exe and entering a path to a video-file when prompted. Another method is by calling the Detector.exe from the command prompt with the first argument being the path to the video-file:

"path/to/detector/detector.exe video/path.avi"
The software accepts both .AVI and .MPEG formats. During face-detection, pressing escape stops the face-detection procedure and forces the software to recognize using only the detections made until escape was pressed.

C.2 Changing the Settings

In order to change the detection and recognition parameters, one can access the Settings.rsf file. This file contains a list of settings allowing one to customize the software to a specific task. Here, one can also change the Detection Cascade and the Active Appearance Model, this enables the software to deal with recognition-tasks not necessarily related to face-recognition since any detector and appearance model can be taken into use.

C.3 Reading the Output

After the software has processed video-footage it informs the user if the subject was present and if so, it will also output timestamps about the subjects appearance.
If one is interested in a more detailed insight in the recognition process, a file named Detections.txt will be created after every run, allowing one to see the complete collection of Mahalanobis Distances calculated. Also snapshots taken by the software are accessible, all snapshots are placed in the processing/ folder together with their associated AAM-matches.

C.4 Known bugs

If a grey-scale Active Appearance Model is taken into use, it is very important that grey-scale mode is also activated in the settings (or vice versa). If this is not done, the software will crash.

Too long or too face-intensive video-footage can cause the software to reach the maximum amount of face-detections allowed, resulting in memory-allocation errors during the detection phase. To avoid this, one should check that a reasonably high Continuity Filtering Threshold is being used. If the error persists, one can split the video-sequence into smaller sequences.
In very rare cases the Active Appearance Model Software encounters image-regions that it is not able to process properly. This results in the software crashing during the recognition phase. In order to avoid this error, one can change the snapshotting-intervals or continuity threshold in the settings and thereby prevent the software from snapshotting that specific frame again.

If a crash cannot be solved using the above solutions, ensure that the correct files and folders exists in the software folder:

```
processing/
Data/
    jarlmodel.amf
    haarcascade_frontalface_alt2.xml
Detector.exe
Settings.rsf
aamc.exe
aamcm.exe
cv200.dll
cvaux200.dll
cxcoer200.dll
cxts200.dll
highgui200.dll
ml200.dll
opencv_ffmpeg200.dll
VisCore.dll
VisMatrix.dll
VisXCLAPACK.dll
```

If any of these files or folders are missing, the software will not run properly. If this is the case, please redownload the software from the url in the start of this appendix.


