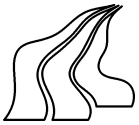
AALBORG UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE



FrB. vej 7E, 9220 Aalborg Ø, Tlf: +45 96 35 80 80 , Fax: +45 98 15 98 89

Theme:

Computer Vision

Title:

Speaker Attention for
Video Conferencing
using Multiple Visual Cues

Project period:

February 5th 2001 – June 6th 2001

Group:

N3-214

Group members:

Peter Zinck Nielsen
Bjarke Andersen

Supervisors:

Moritz Störring
Ole-Christoffer Granmo

Number of copies: 7

Pages: 168

Appendices: 4

Day of completion: June 6th, 2001

Abstract:

This report describes the investigation of methods to be used in a video conferencing system, where a person gets the attention by raising his or her right hand. A system is designed and implemented and a number of video recordings made to make it possible to do experiments on the methods.

To constrain the search area for hand raises, the faces in the videos are found and tracked. To find the faces, we first detect the skin-colours in the images using either lookup tables (LUTs) or Gaussian models. Methods which make it possible to adjust to changes in illumination colour are also investigated. A list of face candidates is made and each face candidate verified by looking at the size, solidity, similarity to a nose-eye template, and elliptic shape. Face trackers are updated and new trackers started based on the face list. Different methods for tracking are investigated and a combination of the Mean Shift algorithm, ellipse fitting, and a Kalman filter is found to be suitable. Based on the face trackers, the areas in which to search for hand-raises are defined. To detect hand-raises the accumulated difference pictures (ADPs) are used. Hand-raises will leave a vertical track in the ADPs, and can therefore be distinguished from other skin-coloured objects passing by in the background or foreground.

Experiments are made to determine the best combination of methods to use and to find out how well the system handles different situations such as illumination change, occlusion, movement in the background, etc. Finally, suggestions for future work are given and the investigations and results made in this report are concluded upon.

Dansk Resumé

Denne rapport omhandler hvorledes det kan gøres muligt for en person i en videokonference at opnå opmærksomhed fra et kamera ved at række højre hånd i vejret. Metoder til dette formål undersøges og implementeres i et system, som gør det muligt at eksperimentere med metoderne i forskellige kombinationer.

For at begrænse størrelsen af de områder i billedet hvori der ledes efter håndsoprækninger, findes ansigterne først og følges over tid. For at finde ansigterne benyttes der hudfarvedetektion ved brug af enten opslagstabeller eller Gaussianske modeller. Ydermere undersøges og eksperimenteres der med metoder, som gør det muligt for systemet at tilpasse sig til skift i lyskildefarven. Resultatet af hudfarvedetektionen segmenteres til en liste af ansigtskandidater, som verificeres ved at tjekke deres rektangulære form og størrelse, deres soliditet, deres lighed med en gennemsnits næse-øje-skabelon og deres elliptiske form. På baggrund af denne liste opstartes nye ansigtsfølgere og allerede igangværende følgere opdateres. Forskellige metoder som kan bruges til at følge ansigter undersøges og en kombination af Mean Shift-algoritmen, ellipsetilpasing og et Kalmanfilter findes velegnet til brug i de miljøer, som vi beskæftiger os med i denne rapport. Ud fra ansigtsfølgernes positioner beregnes de områder hvori der skal ledes efter håndsoprækninger. For at finde håndsoprækninger kigges der på det akkumulerede differesbillede, hvori håndsoprækninger kan ses som et vertikalt spor, hvilket gør det muligt at skelne dem fra andre hudfarvede objekter, som det kan hænde passerer forbi i bag- eller forgrunden.

Ved hjælp af eksperimenter findes der frem til de kombinationer af metoder, som er bedst at bruge. Desuden bruges eksperimenterne også til at finde ud af, hvordan systemet håndterer forskellige situationer såsom skift i lyskildefarve, okklusion, bevægelse i baggrunden m.v. Til sidst beskrives de fremtidige udvidelsesmuligheder for systemet og der bliver konkluderet på de undersøgelser og resultater, der er opnået i rapporten.

Preface

This Master's Thesis is the result of a project made on the DAT6/F10S semester, spring 2001. It describes the investigation of methods for automatic speaker attention in video conferences. Furthermore, a video conferencing system is implemented and used for experiments.

The purpose of the report is to demonstrate that we are capable of using the theories in the area of computer vision to do a thorough analysis of an actual computer vision problem. Furthermore, a theoretical or practical solution to the problem must be presented.

Aalborg University, June 6th, 2001.

Peter Zinck Nielsen

Bjarke Andersen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description

Video conferencing makes it possible for groups of people located in different parts of the world to communicate almost as if they were in the same room. This is a great advantage for companies that have offices in many countries, since it can save them a lot of travel expenses and make their employees feel more as a single unit, although they are separated by thousands of miles. Compared to other forms of communication such as emails and telephone calls, video conferencing has the advantage of sending live images. This makes it a lot easier to communicate, because the involved people can make full use of their body languages when explaining something. Figure 1.1 illustrates a typical video conference setup as we imagine it in the system we describe in this report.

One of the primary problems when having a video conference is to make sure that the person who is talking has the attention – i.e. is zoomed in on. If we just have a camera that statically shows a whole group of people, it may be very difficult for other participating groups to see who is talking and important information might be lost. One way of solving this is to let a human control the camera and he must make sure to zoom in on the person who is talking. Of course this is not optimal, since the person doing that is unable to participate in the conference, and at the same time he or she must be paid for doing a rather tedious job. Some commercial systems [30, 29, 41] try to automate the control task by using acoustics-based tracking, where a microphone placed on the camera is used to find the direction of the person who is talking and make the camera zoom in on him or her. Although this seems as a good solution, it has the disadvantage of being sensitive to acoustic noise.

We would like to investigate if the control of the camera can be done in another way, using the video signal which is unaffected by acoustic noise. If this is possible, an approach using both acoustic and visual cues could result in an even more robust system.

One way of discovering the talking person based only on visual information, is to look for the place in the video stream that is most active, since people tend to be more active when they talk than when they are silent [10]. Another solution could be to search for faces in the video stream, and then make use of some kind of hand signaling, e.g. raising the hand, to identify the person that wants to talk. Since humans have limitations in their physical behaviors it should be possible to identify to which head a raised hand belongs. The people participating in the video conference could also be equipped with a button connected to the video conference system. This way a person could press the button

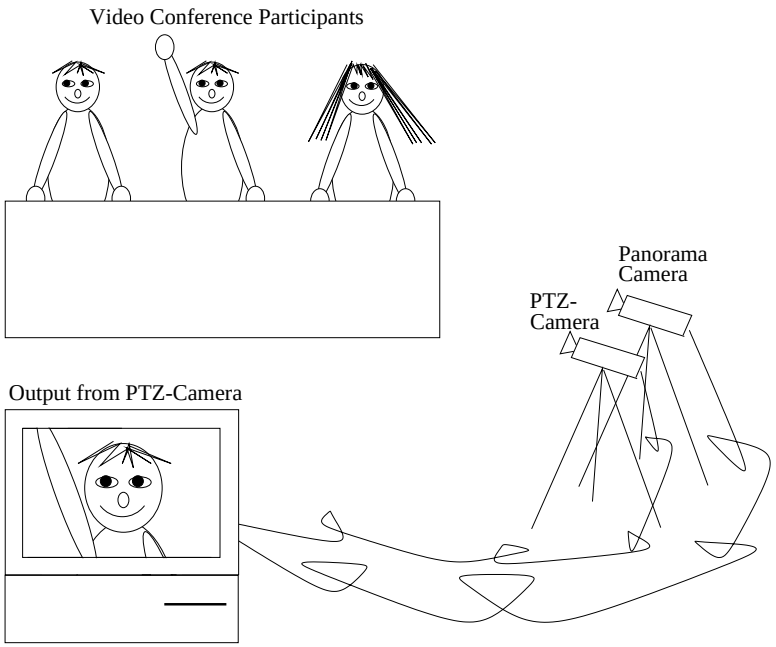


Figure 1.1: Typical Video Conference Setup: One of the cameras provides a "panorama" view of the participants, while another camera is zoomed in on the person who is speaking. It is the image from the latter camera which will be sent to the video conference participants at the other location(s).

when attention is wanted. One great disadvantage of this is that the system will have to be calibrated every time the person using a particular button moves significantly (e.g. more than 30 cm.), otherwise the camera will not know where to look for the person who pressed the button. Furthermore, this increases the hardware complexity of the system since it will consist not only of the cameras and the computer, but also of the buttons.

In this project we have focused on developing a video conference system that makes use of the hand-raise principle to detect the person who wants to get the attention. The system will consist of two cameras; the *panorama camera* and the *Pan-Tilt-Zoom-camera* (PTZ-camera). The position and zoom-level of the panorama camera is fixed and its output is used to search for faces and hand-raises. If a hand-raise is detected, the PTZ-camera is used to zoom in on the face of the person who wants the attention (see Figure 1.1). Throughout the report we will refer to the system as VICOWIJOY (**VI**deo **CO**nferencing **WI**thout a **JOY**stick) or the VICOWIJOY system.

1.2 Existing Systems

A system that has some similarity to the one we have in mind, is described in [10]. Here Krüger et al. present a teleconference system which uses an attentive camera. They assume that the person who is talking creates more motion that the listeners. This assumption is used to turn a Pan-Tilt-Zoom-camera (PTZ-camera) in the direction of the area in the image which contains the most motion. Afterwards, it is verified that there is indeed a person in that direction, by analyzing the acoustic cues received through a

microphone mounted on top of the camera. Video examples from the system in use can be seen on *http://www.ks.informatik.uni-kiel.de/~vok/research/research.html*.

Commercial systems for automatic speaker attention do also exist. Examples of these are the SmartTrak Camera System made by VTel [41] and Polycom's ViewStation [30]. Both of these make use of voice tracking cameras – i.e. acoustic cues are received through a microphone mounted on the PTZ-camera and used to find the direction of the speaker. We have not been able to find any commercial systems that make use of visual cues to focus on the current speaker. This alone suggests that it is an area which needs more attention, which we hope to give it with this report.

1.3 System Overview

To be able to zoom in on a persons face, its position must be known. This can be achieved using face detection. However, face detection may not find a particular face in each image. Therefore, the system should have a way to keep track of where the faces are, when face detection does not find them. This can be done using a face tracker for each face that is found using face detection. The tracker does not need to rely on the face positions and sizes found by the face detection, but can use its own methods for finding the face, based on where the face was in the last image, and the assumption that the face will be close to this position. This way, its becomes acceptable that the face detection often does not find the face, as long as new faces will be detected within a reasonable amount of time after they appear in the image (e.g. a few seconds).

A face tracker is also desirable for another reason. When a person is raising his hand, he can be expected not to be moving his head very much, that is, it is spatially stable. This can be used to make the system more robust towards the different kinds of noise that can occur during a video conference, such as persons walking by in the background. If it is ensured that the camera only will zoom in on spatially stable objects, a face moving in the background could not accidentally be zoomed in on, as it is spatially unstable. Whether a face is spatially stable can only be determined from temporal information about its position and size, which a face tracker can provide.

The entire VICOWIJOY system illustrated in Figure 1.2 consists of a face detector, face trackers, hand-raise detectors, and a control module for the PTZ-camera. Separate face trackers are started for each of the faces found by the face detector, and each face tracker is associated with its own hand-raise detector that detects when the person whose face is being tracked raises his hand. The control module uses the information about the position and size of the face from the face tracker for controlling the PTZ-camera to zoom in on the person who has raised his hand.

When the face of a person is being tracked, it will be possible to restrict the search for hand-raise gestures to a region of interest (ROI) near the person, thus reducing the computational needs of the system. We will refer to this ROI as the *hand-raise ROI*. The hand-raise ROIs of two persons are illustrated in Figure 1.3. We impose the restriction that hand-raises must be done using the right hand. Therefore, the hand-raise ROI will be an area to the left of the person in the image.

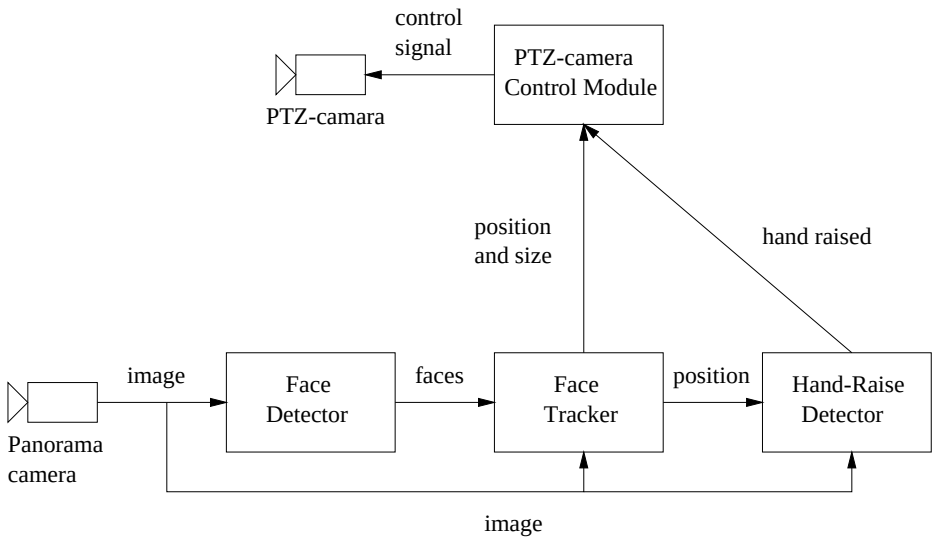


Figure 1.2: The VICOWIJOY Video Conferencing System.

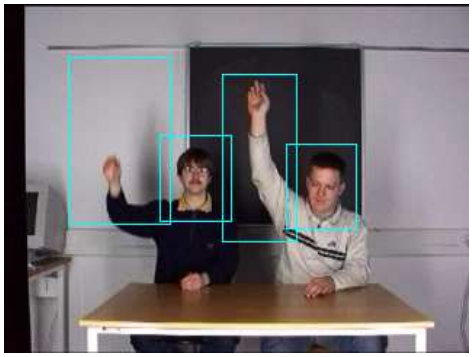


Figure 1.3: Hand-Raise ROIs. The two large rectangles are the hand-raise ROIs for the two persons being tracked by the system.

1.4 Delimitation

We restrict ourselves to the investigation of methods for detection and tracking of faces, and detection of hand-raise gestures. We shall not consider the control of the PTZ-camera, nor the communication, i.e. the exchange of images, between systems at different locations participating in a video conference.

We impose the following restrictions on the environment and participants in the video conference[1]:

- The background cannot contain large areas of skin coloured material/objects and cannot contain a large number of sharp edges.
- Participants are only allowed to raise their right hand, and the top of the fingers must be at least at the same height as the top of the head when the hand is raised. The hand must be within the image while it is being raised. The head of the hand-raising participant must be within the image as well.
- Participants are placed in a single row and the minimum inter-person distance is 0.6 m (center-to-center). They are not allowed to occlude each other.

[1]These restrictions are inspired by a set of requirements supplied by Devitech ApS.

- The participants are only tracked when they face (profile-to-profile) the camera. In order to simplify the face detection, the participants must look straight into the camera for the tracking to be initiated. Prior to this, the participants may not be tracked.
- The camera must be placed in approximately the chest/head height of the participants.
- The number of participants must be between 1 and 3.

1.5 Outline of Report

The structure of the report reflects the architecture of the system presented in Figure 1.2. In Chapter 2, methods for finding skin-coloured objects in the image are investigated. These methods are used by face detection, face tracking, and hand-raise detection. Face detection is described in Chapter 3, face tracking in Chapter 4, and hand-raise detection in Chapter 5. In Chapter 6, we present the design of the system that we have implemented and used for experiments. The experiments and their results are described in Chapter 7. This is followed by suggestions for future work in Chapter 8 and conclusions in Chapter 9.

Chapter 2

Focus of Attention

In this chapter, methods to use for the initial focus of attention in the VICOWIJOY system are described. Several ways of detecting skin-colours are described. Furthermore, we in particular describe how the system can adapt to illumination changes, e.g. due to a cloud covering the sun or artificial lights being switched on and off.

2.1 Introduction

When searching for humans the two major methods to use for focus of attention are skin-colour detection and motion detection. We have decided to focus only on detection of skin-colours, since the investigations we made in [5] showed that using motion detection in the focus of attention phase did not aid the search for human faces. In Chapter 5 motion detection is nevertheless used for detecting hand-raises. We do not include this as a part of the focus of attention phase, since it only applies to the detection of hand-raises and not detection of faces.

The first task of the VICOWIJOY system is to direct the attention to regions in the image which are likely to contain the objects of interest (also known as ROIs for regions of interest). In our case these are faces and hands. We will refer to this task as *Focus of Attention* (FoA). Using FoA makes the job of the following methods of face verification, face tracking, and hand-raise detection much easier, since they only have to operate in the areas found by the FoA methods. Because the face verification methods in most cases are more complex and thereby more computational demanding than the FoA methods, the use of FoA also makes the system able to perform faster (i.e. process more frames per second).

When searching for face and hands as we do in this project, detection of skin-colours in a chromatic colour space has been shown to be very effective to use as FoA [26, 20, 2, 39]. In these methods a likelihood image is created based on the likelihood of each pixel being skin-coloured. Further processing of this image is then used to detect the skin-coloured areas. The advantages of skin-colour detection methods are that they are invariant towards size and orientation and also light intensity if the right colour models are used (refer to Section 2.2). Furthermore, they are fast and therefore suitable for real-time tracking. To identify the skin-colour likelihood of a colour, several methods can be used. We have decided to investigate the use of *Lookup Tables* (LUTs) and *Gaussian models*. These are described in Sections 2.3 and 2.4.

Although the detection of skin-colours is invariant towards changes in light intensity,

changes in light colour have a great influence on where to search for skin-colours in the colour space used. Therefore we have also investigated methods which make it possible for the system to adapt to changes in lighting colour. These are described in Section 2.5

2.2 Colour Models

It has been shown that although not very obvious to the human eye, the human skin-colours lie in a small cluster when intensity is removed (i.e. skin-colours change in intensity but not in chromaticity) [33]. In order to make the skin-colour detection invariant towards changes in the intensity of lighting, we need to convert the RGB values of the input image to another representation, which has lighting as one of its parameters. Examples of such models are YCrCb, HSV, YUV, and Normalized RGB [44]. The results in [44] show that the choice of colour model is not that important – they more or less perform equivalently in the experiments described in the article. Therefore we have chosen to use the Normalized RGB (NRGB) model since this is the colour model we have read most about in the literature.

To transform an image from the RGB space to its representation in the NRGB space (also known as the *chromaticity plane*, where the chromatic values indicate the "pure" colours) Equations 2.1 through 2.3 are applied to every pixel.

$$r = \frac{R}{R+G+B} \tag{2.1}$$

$$g = \frac{G}{R+G+B} \tag{2.2}$$

$$b = \frac{B}{R+G+B} = 1 - r - g \tag{2.3}$$

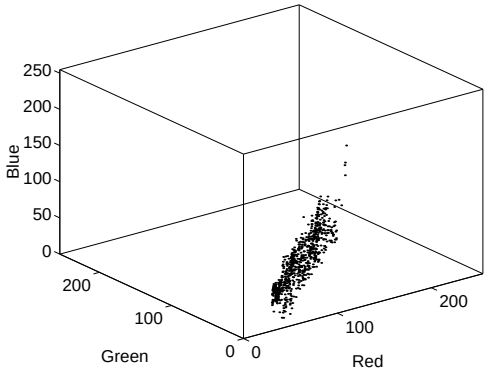
In Figure 2.1 it can be seen how a 3 dimensional distribution of skin-colours in the RGB model can be represented in only 2 dimensions in the chromaticity plane. This is because the last chromaticity b can be calculated when r and g are known, since intensity no longer is part of the colour space.

2.3 Lookup Tables

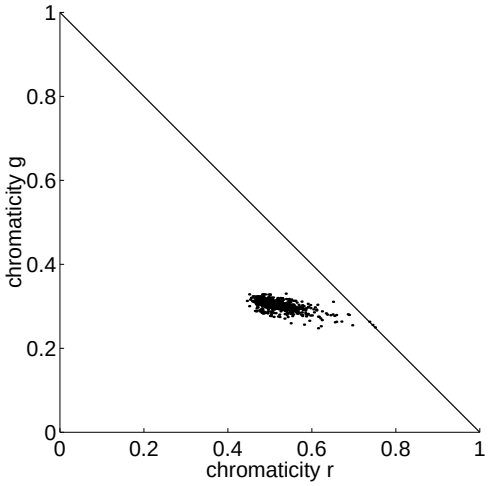
A simple way of verifying whether a colour is skin-coloured or not, is to define a set of thresholds holding the minimum and maximum allowable values of chromaticity r and g. Another way of describing this method is that a rectangular area for chromaticity r and g in the chromaticity plane (see Figure 2.2(a)) is defined. Colours inside this rectangle are defined as skin-coloured and colours outside the rectangle as non skin-coloured. This method is rather simple, however it has some significant drawbacks. First of all, the rectangle must be large enough to cover the colours of skin as they appear under all kinds of illumination. Therefore, many other colours than the actual skin-colours, will be identified as skin-coloured. These can be considered as *false positives* whereas *false negatives* are skin-colours which are identified as non skin-colours. Using the rectangle to define the skin-colours will in general give a high number of false positives and a low number of false negatives.

Another drawback of using the rectangle is that a colour is defined as either skin-coloured or non skin-coloured (i.e. its likelihood is either 1 or 0). Therefore, the likelihood image

(a) Selected Skin-Colour Region



(b) RGB Model



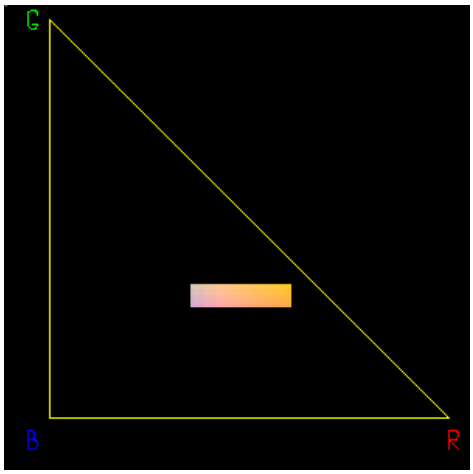
(c) Chromaticity Plane

Figure 2.1: Skin-Colour Representations: The selected skin-colour region (a) is represented by 3 dimensions in the RGB model (b) and 2 dimensions in the chromaticity plane (c).

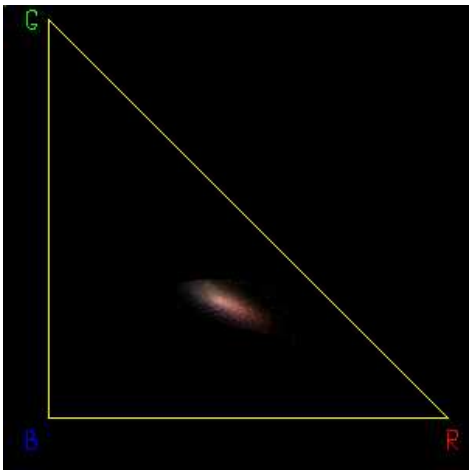
will be a simple binary image where white pixels indicate skin-colours and black pixels non skin-colours (see Figure 2.2(d)). This is a problem when the number of false positives are high, since areas of non skin-colours might be joined with skin-coloured areas. This can lead to a very inaccurate result of FoA and thereby an increasing computation time of the following methods of face verification and hand-raise detection.

Instead of using the rectangle or the thresholds, a *lookup table* (LUT) may be used. In this project this will be a two dimensional table which holds the values of chromaticity r and g in the chromaticity plane (r is the columns and g the rows). Each record in the LUT holds the likelihood of this particular combination of r and g being skin-coloured (see Figure 2.2(b), where high intensity indicates a high likelihood).

Compared to the use of the rectangle, the LUT makes it possible to be much more accurate when defining whether a colour is skin-coloured or not. First of all, the area in the LUT does not have to be as large as the rectangle. If the LUT e.g. is updated using the colours of the skin-coloured objects that are being tracked as input, it can adapt to the current skin-colours with good accuracy (see Section 2.5 for more about adaption). Moreover,

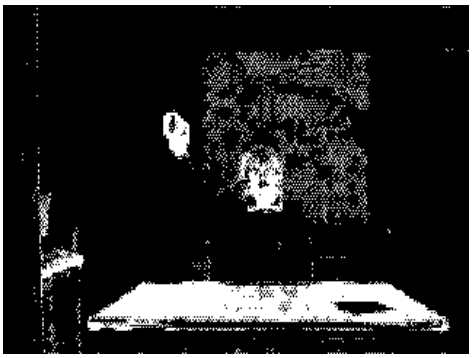


(a) The Skin-Colour Rectangle in Chromaticity Plane

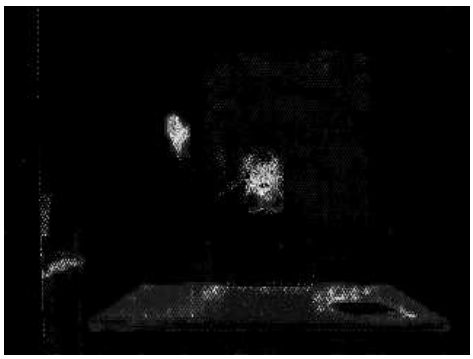


(b) LUT in Chromaticity Plane

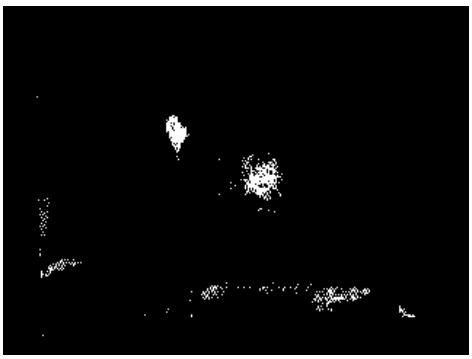
(c) Input Image



(d) Binary Image using the Rectangle



(e) Likelihood Image using the LUT



(f) Thresholded Likelihood Image using the LUT

Figure 2.2: Comparing the Rectangle with the LUT: Based on the skin-colour rectangle (a) a binary image (d) is created from the input image. Using a LUT (b) a skin-colour likelihood image (e) is made. Comparing the thresholded likelihood image for the LUT (f) with the binary image for the skin-colour rectangle (d), shows that the use of a LUT gives a more accurate and more noise free result.

it is possible to reflect how likely it is that a specific colour is skin-coloured – looking at Figure 2.2(e) (where high intensity indicates a likelihood close to 1 and low intensity a likelihood close to 0) it can be seen, that the faces and hands are much more likely to be skin-coloured than e.g. the table. Finally, thresholding the likelihood image makes it possible to ignore pixels that have a very low likelihood of being skin-coloured (can be considered as noise). Comparing the result of using the rectangle in Figure 2.2(d) with the result of using a LUT in Figure 2.2(f), it can be seen that using a LUT reduces the noise and increases the accuracy.

2.4 Gaussian Models

Skin-colours can be described using a Gaussian model if the colours are normalized and the skin is illuminated by a single colour [42, 2]. According to [39, 37] this even holds when different races are represented in the image.

2.4.1 Estimating the Model from a LUT

According to Feris *et al.* in [33] and Sun *et al.* in [39], a unimodal Gaussian density function can be denoted as $N(\mathbf{m}, \Sigma^2)$ where

$$\mathbf{m} = \begin{bmatrix} r_{avg} \\ g_{avg} \end{bmatrix}$$

or simply the center of mass, and Σ^2 the covariance matrix[1] where

$$\Sigma = \begin{bmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{bmatrix}$$

and σ_{**} the standard deviations. To calculate the mean of chromaticity r, r_{avg}, Equation 2.6 is used. In this, r_i represents chromaticity r of a vector $\mathbf{x}_i = [r_i \; g_i]^T$ in the LUT S and N the size of S. The function $S(\mathbf{x}_i)$ returns the normalized skin-colour likelihood of the colour represented by the vector $\mathbf{x}_i$. This is done according to Equation 2.4, where $\mathbf{x}_{max}$ represents the colour with the highest frequency in S. The reason why we use the the normalized skin-colour likelihood is that we want to use a fixed threshold value to segment the likelihood image into a binary image of skin-colours and non skin-colours. Alternatively, we could use a variable threshold value, which should be calculated based on the likelihood of the colour with the highest frequency in S. We have chosen the first strategy, since we think it is easier to relate to a fixed threshold than a changing threshold.

The function $lsum(S)$ defined in Equation 2.5 returns the sum of the likelihoods for all $\mathbf{x}_i$ in the dataset S.

The mean for chromaticity g is calculated in the same way, only now chromaticity g, g_i, is used from the LUT S. The equation for this can be seen in Equation 2.7.

$$S(\mathbf{x}_i) = \frac{freq(\mathbf{x}_i)}{freq(\mathbf{x}_{max})} \tag{2.4}$$

[1] We refer the reader to Appendix A for further description of probability theory.

$$lsum(S) = \sum_{i=1}^{N} S(\mathbf{x}_i) \tag{2.5}$$

$$r_{avg} = \frac{1}{lsum(S)} \sum_{i=1}^{N} r_i S(\mathbf{x}_i) \tag{2.6}$$

$$g_{avg} = \frac{1}{lsum(S)} \sum_{i=1}^{N} g_i S(\mathbf{x}_i) \tag{2.7}$$

The covariance matrix, Σ^2, of the Gaussian model, defines how concentrated the probability mass is around the center $\mathbf{m}$. A low covariance indicates that the probabilities are high close to $\mathbf{m}$, and a high covariance indicates that the probabilities are distributed over a larger area. To calculate the covariances, σ^2_{**}, in Σ^2, based on the LUT S consisting of vectors $[r_i \;\; g_i]^T$ and their likelihoods, Equations 2.8 to 2.11 are used.

$$\sigma^2_{rr} = \frac{1}{lsum(S)} \sum_{i=1}^{N} (r_i - r_{avg})^2 S(\mathbf{x}_i) \tag{2.8}$$

$$\sigma^2_{rg} = \frac{1}{lsum(S)} \sum_{i=1}^{N} (r_i - r_{avg})(g_i - g_{avg}) S(\mathbf{x}_i) \tag{2.9}$$

$$\sigma^2_{gr} = \frac{1}{lsum(S)} \sum_{i=1}^{N} (g_i - g_{avg})(r_i - r_{avg}) S(\mathbf{x}_i) \tag{2.10}$$

$$\sigma^2_{gg} = \frac{1}{lsum(S)} \sum_{i=1}^{N} (g_i - g_{avg})^2 S(\mathbf{x}_i) \tag{2.11}$$

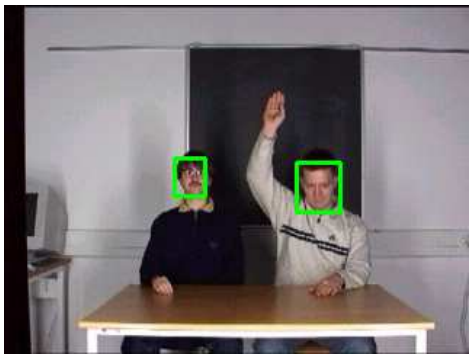
As it can be seen from Equation 2.9 and 2.10, the values of σ^2_{rg} and σ^2_{gr} will always be equal. The estimation of the model can therefore be computationally optimized, if we only calculate one of these values and then assign this value to the other. In Figure 2.3 the appearance of a Gaussian model estimated from the skin-colours of the faces in an image can be seen. The estimated center of mass, $\mathbf{m}$, and covariance matrix, Σ^2 are described below.

$$\mathbf{m} = \begin{bmatrix} 0.467 \\ 0.277 \end{bmatrix} \qquad \Sigma^2 = \begin{bmatrix} 0.435 & -0.104 \\ -0.104 & 0.087 \end{bmatrix}$$

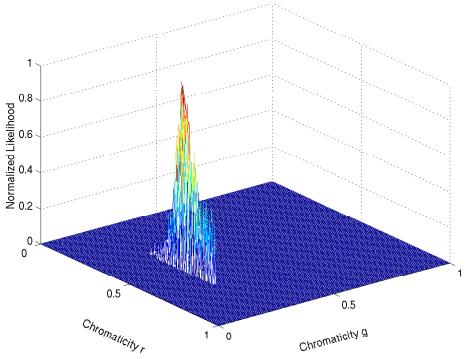
2.4.2 Calculating Likelihoods from the Model

The Gaussian model $N(\mathbf{m}, \Sigma^2)$ estimated in the previous section, can be used to calculate skin-colour likelihoods of the pixels in an input image. The image colours are first converted to chromaticity plane and afterwards the likelihood, $l_{skin}(\mathbf{x}_i)$, of each pixel, $\mathbf{x}_i$, being skin-coloured is calculated using Equation 2.12.

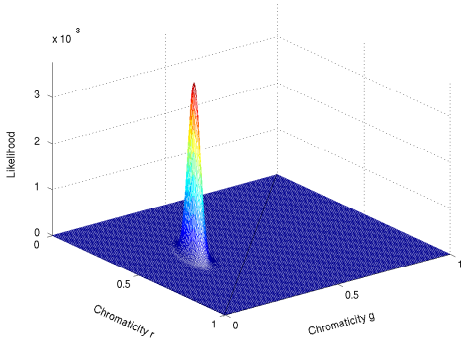
$$l_{skin}(\mathbf{x}_i) = \frac{exp[-\frac{1}{2}(\mathbf{x}_i - \mathbf{m})^T \; inv(\Sigma^2)(\mathbf{x}_i - \mathbf{m})]}{2\pi \; det(\Sigma^2)^{\frac{1}{2}}} \tag{2.12}$$



(a) Input Image



(b) Chromaticity Distribution in LUT



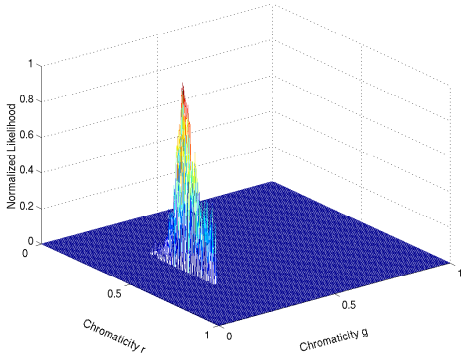
(c) Estimated Gaussian Model

Figure 2.3: Estimating a Gaussian Model from a LUT: Using the skin-colours in the faces in the input image (a) a LUT is created (b). The Gaussian model (c) is then estimated based on the chromaticity r and g values and the normalized skin-colour likelihoods of the colours represented by (r, g).

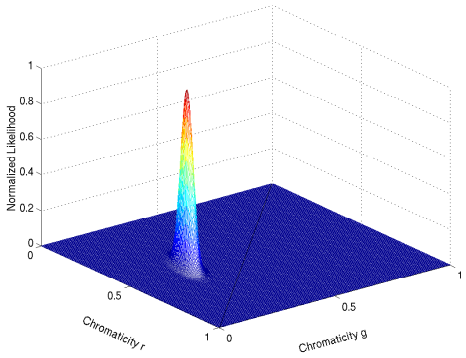
Since a Gaussian model is a probability model and we use a 2 dimensional model to describe skin-colours, its volume is 1. Therefore, the center does not usually have the likelihood 1, unless the model is estimated from a single value where $S(\mathbf{x}_i) = 1$. E.g. the likelihood of the center of the model estimated in last section is less than $3.5 \cdot 10^{-3} = 0.35\%$ (see Figure 2.3). To get the normalized likelihood, we need to adjust the result of Equation 2.12, such that a colour placed in the center of the Gaussian model gets a likelihood of 1. This is done as described by Equation 2.13 where every likelihood, $l_{skin}(\mathbf{x}_i)$, is divided by the likelihood of the center of mass in the Gaussian model, $l_{skin}(\mathbf{m})$.

$$norml_{skin}(\mathbf{x}_i) = \frac{l_{skin}(\mathbf{x}_i)}{l_{skin}(\mathbf{m})} \tag{2.13}$$

The image shown in Figure 2.4(e) illustrates the skin-colour likelihoods when using a Gaussian model. Comparing it to the result of using a LUT which is shown in Figure 2.4(d), no significant difference can be observed. According to [24] the use of LUTs should give slightly more accurate results compared to the use of Gaussian models. Furthermore, LUTs should be faster to calculate, which is desirable in real-time systems.

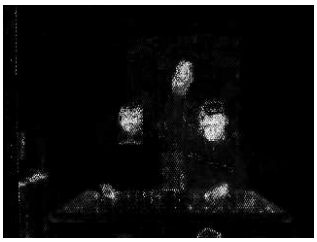


(a) LUT



(b) Gaussian Model

(c) Input Image



(d) LUT Likelihood Image

(e) Gaussian Likelihood Image

Figure 2.4: Comparing a Gaussian Model with a LUT: The likelihood images (d) and (e) of the input image (c) is created based on a LUT (a) and a Gaussian model (b). The results are close to identical.

2.5 Handling Changes in Light Colours

As long as the lighting colour does not change, a LUT or a Gaussian model estimated from a training set of images can be used to find skin-colours. However, in video conferences lighting colour may change, i.e. lights in the ceiling are turned on and off, light coming through a window changes when the sun comes and goes, curtains are pulled forth and back, etc. When the colour of the lighting changes, the area in the chromaticity plane which represents skin-colours (also known as the skin chromaticity distribution) moves [38, 20]. A system which should be able to handle changes in lighting colour must therefore be able to adjust itself to these changes based on observations from the input images.

2.5.1 The Skin Locus

When the illumination colour changes, the skin chromaticity distribution moves along a locus which is similar to the *Planckian locus* [26, 19]. The Planckian locus is the line in chromaticity plane along which colours of *Blackbody radiators* are placed[2]. A Blackbody radiator is a theoretical object, which is a perfect radiator that changes in colour when heated. The *correlated colour temperature* (CCT) of a light source may then be measured as the temperature (measured in Kelvin (K)) needed to make a Blackbody

[2]Refer to Appendix C for more information about the skin-colour model we have used in this project. This model was first described by Störring *et al.* in [27]

radiator the same colour (we refer to [18] for more theory about Blackbody radiators and colour temperatures). Not all light sources have a colour that is similar to the colour of a Blackbody radiator, but the everyday light sources such as sunlight, fluorescent lamps, light bulbs, etc. do. As long as these are used, the skin-colour model we have used in this project should be valid.

In the following we refer to the line along which the skin chromaticity distributions in the skin colour model are placed as the *skin locus*. What happens is that when the CCT decreases (i.e. when the CCT is low objects appear reddish and when the CCT is high they appear bluish), the skin chromaticity distribution moves to the right along the skin locus (see Figure 2.5). If the CCT increases, the skin chromaticity distribution moves to the left. Moreover, the size and shape of the skin chromaticity distribution changes, when it moves along the chromaticity r axis. Finally, the skin chromaticity distribution also rotates clockwise when moving right along the chromaticity r axis. In Figure 2.5 examples of skin chromaticity distributions along the skin locus are illustrated. In this figure a membership function defines the upper and lower boundary of the skin chromaticities (the two yellow lines). In [19] they estimated this membership function as two quadratic functions. To decide whether a pixel (r, g) is inside the upper and lower boundary, Equations 2.14 to 2.16 are used. The values of the parameters A_{up}, b_{up}, c_{up}, A_{down}, b_{down}, and c_{down} are estimated from a set of training images.

$$g_{up} = A_{up}r^2 + b_{up}r + c_{up} \tag{2.14}$$

$$g_{down} = A_{down}r^2 + b_{down}r + c_{down} \tag{2.15}$$

$$skin(r, g) = \begin{cases} 1, & (g < g_{up}) \text{ and } (g > g_{down}) \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

2.5.2 Lookup Table Update

To adapt to changes in CCT when using LUTs, the skin-coloured areas found in the images can be used to update the LUT. If e.g. the faces are detected and tracked, the LUT can be updated with pixels from these. In this way a change in CCT will eventually be reflected in the LUT. In the following we will explain two ways of updating a LUT.

Simple Update

The simplest way of updating a LUT, S, at time t based on a region of interest (ROI – could e.g. be a face) in the input image, is to create a LUT, M, for this ROI and then adjust the values in S according to Equation 2.17.

In this, $S(\mathbf{x}_i)$ and $M(\mathbf{x}_i)$ returns the normalized skin-colour likelihood of a colour $\mathbf{x}_i = [r\ g]^T$ in respectively S and M. $skin(\mathbf{x}_i)$ uses Equation 2.16 to detect whether the $[r\ g]^T$ values in $\mathbf{x}_i$ are inside the area of skin-colours in chromaticity plane or not.

$$S_t(\mathbf{x}_i) = ((1 - \alpha)S_{t-1}(\mathbf{x}_i) + \alpha M_t(\mathbf{x}_i))skin(\mathbf{x}_i) \tag{2.17}$$

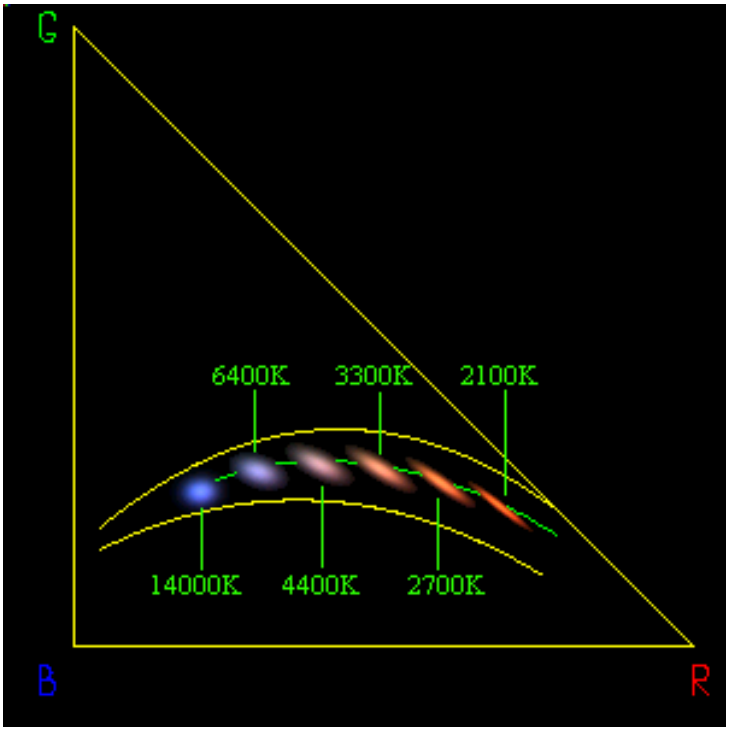


Figure 2.5: The Skin Locus: The skin chromaticity distribution moves along a locus similar to the Planckian locus of Blackbody radiators when the CCT changes. When going from the left to the right along the skin locus, the area of the skin chromaticity distribution also changes in size and shape and rotates clockwise. The figure illustrates 6 different skin chromaticity distributions along the skin locus. Upper and lower boundaries for skin chromaticities (the yellow lines) can be estimated from a set of training images.

The constant $0 \leq \alpha \leq 1$ is used to control how fast the adaption occurs – i.e. when α is close to 1 the system will adapt very quickly.

When S has been updated according to M, it must be normalized such that the highest likelihood of a pixel $S(\mathbf{x}_i)$ is 1. This is done by dividing every value in S with its maximum likelihood.

Ratio Update

Calculating LUTs as *ratio histograms* [19] is another way of finding the likelihoods of skin-colours. Not normalized LUTs are made for both the ROI, M, and for the whole image, I. The ratio LUT, R, at time t is then calculated as

$$R_t = \frac{M_t}{I_t}$$

and afterwards normalized likelihoods are computed by dividing R with its highest likelihood value. In this way, colours which fall inside the skin chromaticity area and are highly represented in the input image, will not have great effect on the likelihoods in the LUT. If e.g. a large, light brown cupboard is present in the input images, the use of ratio update ensures that skin-colours that are highly represented on this cupboard do not get high likelihoods. Therefore, the cupboard colours will not get high skin-colour likelihoods although they should happen to be of some of the same colours as the faces that are being tracked.

Equation 2.18 illustrates how the colour $\mathbf{x}_i$ in S is updated at the time t when using ratio update. The function $skin(\mathbf{x}_i)$ uses Equation 2.16 to determine if a colour $\mathbf{x}_i$ falls inside the skin chromaticity area. The constant $0 \leq \alpha \leq 1$ is used to control the adaption speed over time.

$$S_t(\mathbf{x}_i) = ((1-\alpha)S_{t-1}(\mathbf{x}_i) + \alpha R_t(\mathbf{x}_i))skin(\mathbf{x}_i) \tag{2.18}$$

2.5.3 Gaussian Model Update

When using a Gaussian model to represent skin-colours in chromaticity plane, the adaption to changes in CCT can occur in many ways. Amongst these are *Gaussian ratio update* and *weighted parameters of Gaussians* which will be described in the following sections.

Gaussian Ratio Update

This method makes heavy use of the LUT ratio method explained in the previous section. A Gaussian model $N(\mathbf{m}, \Sigma^2)$ is represented by a LUT. This is done by calculating the normalized likelihood for each of the possible positions, $\mathbf{x}_i$, in the LUT using Equation 2.13 on page 13. The LUT is then updated with skin-colours from ROIs in the images in the same way as explained in Section 2.5.2. When this is done, a new Gaussian model is estimated from the updated LUT as described in Section 2.4.1 on page 11.

Shortly said this method is the same as LUT ratio update, except that a Gaussian model is estimated from the LUT and afterwards used to update the LUT to hold likelihoods which fits to this model.

Weighted Parameters of Gaussians

In this method a Gaussian model $M(\mathbf{n}, \xi^2)$ is estimated from the ROIs in the input image. Based on the parameters from this model, the parameters of the Gaussian model for skin-colours, $N(\mathbf{m}, \Sigma^2)$, is updated at time t using Equation 2.19 and 2.20.

$$\mathbf{m}_t = (1-\alpha)\mathbf{m}_{t-1} + \alpha\mathbf{n}_t \tag{2.19}$$

$$\Sigma_t^2 = (1-\beta)\Sigma_{t-1}^2 + \beta\xi_t^2 \tag{2.20}$$

As for LUTs, the constants $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ can be used to control how fast the adaption to changes in CCT should occur. I.e. α controls the position of the center of mass, $\mathbf{m}$, of the Gaussian model and β the covariances in the matrix Σ^2.

2.5.4 Moment Constraints

If for some reason the adaption goes wrong, such as when a LUT or Gaussian model "over-adapts" to skin-colours in the face that is being tracked, it could happen that the updated version of the LUT or Gaussian model is distributed over a very small area in chromaticity plane. Over-adaption happens when – according to the current LUT or

Gaussian model – the ROI used to update the LUT or Gaussian model only contains very few, tightly clustered skin-colours. This may happen if the face is partly occluded, the illumination changes, or the face is being tracked inaccurately and parts of the background are present in the ROI. Having small skin chromaticity distributions in the LUT or Gaussian model lead to that only very few skin-colours will be detected in the future images and thereby too small objects will be found. In the worst case faces being tracked are lost.

The opposite may happen if the skin chromaticities in the LUT or Gaussian model distributes over a very large area. Then too many pixels will be identified as skin-coloured and too large objects will be found. Large distributions may occur if not only the face but also parts of a skin-coloured background are present in the ROI used to update the LT or Gaussian model.

Wrong adaption can also move the skin chromaticity distribution away from the skin locus. Again, this could happen due to occlusions, inaccurate tracking, or illumination changes. In the worst case, the skin chromaticity distribution might move to the edge of the boundaries defined in Equations 2.14 and 2.15 on page 15. This could again lead to no skin-coloured pixels being found and, as a consequence of that, loss of faces being tracked.

To avoid these situations, the moments of the LUT and the Gaussian model can be constrained. Examples of how to do that are explained in the following sections.

Center of Mass

To make sure that the skin chromaticity distribution in the LUT or the Gaussian model does not move in a wrong direction (i.e. out of the area of skin-colours), a constraint on the position of its center of mass can be made. How to calculate the center of mass in a Gaussian model was shown in Equations 2.5 to 2.7 on page 12. To calculate the center of mass for a LUT, the same equations are used. To constrain the position of the center of mass of either the Gaussian model or the LUT, we can demand that it lies within a certain distance of the skin locus. This can be done by using the chromaticity r value of the center of mass to set minimum and maximum borders for the chromaticity g value center of mass. I.e. if the chromaticity g center of mass is below the minimum border it is set to the minimum border and if it is above the maximum border it is set to the maximum border. Updating a Gaussian model in this way is simple – we just change the position of its center of mass. For a LUT we also have to move all its likelihoods the same distance as its center of mass is moved.

Variance and Covariance Size

Human skin-colours tend to be distributed across an area of a certain size in chromaticity plane. According to our investigations made in Appendix C, the shape of this area will be close to circular when the CCT is high and a flattened ellipse when the CCT is low. Moreover, the area gets smaller when moving in either direction along the chromaticity r axis from a chromaticity r value of 0.357. I.e. high and low CCTs give small areas, whereas CCTs in between give larger areas. Finally, the area will rotate clockwise when it moves to the right along the chromaticity r axis (refer to Figure 2.5 on page 16).

The verifications of the skin-colour model in Appendix C showed that using the amount of chromaticity r as an indication of how to constrain the variances and covariances in Σ^2

was not a good idea. Overall constraints on the values in Σ^2 should be used instead. To control the size and shape of the skin chromaticity distribution, we can make minimum and maximum borders for the sizes of the variances along the chromaticity r and g axes, σ_{rr}^2 and σ_{gg}^2. Furthermore, we can demand that σ_{rr}^2 always is larger than σ_{gg}^2. Finally, we can ensure clockwise rotation by demanding that the covariances, σ_{rg}^2 and σ_{gr}^2, always are negative (i.e. positive covariances in Σ^2 indicate counterclockwise rotation and negative covariances indicate clockwise rotation). LUTs can also be constrained if we calculate covariance matrices similar to Σ^2 and use constraints on these.

Even in the case where the variances and covariances are of the right sizes, problems may arise for LUTs. This is because their variances and covariances can be represented by a few, relatively large values (this is not the case for Gaussian models since they always spread normally). Therefore, a minimum number of likelihoods must be larger than the mean of the likelihoods in a LUT.

Constraining Gaussian Models

To constrain the shape of the Gaussian model we demand that the variance σ_{rr}^2 along the chromaticity r axis is always larger than or equal to the variance σ_{gg}^2 along the chromaticity g axis. This is done according to Equation 2.21.

$$\sigma_{gg1}^2 = \begin{cases} \sigma_{rr}^2, & \sigma_{gg}^2 > \sigma_{rr}^2 \\ \sigma_{gg}^2, & \text{otherwise} \end{cases} \tag{2.21}$$

To constrain the area size of the Gaussian model, minimum and maximum values for the variances along the chromaticity r and g axis are used. These can be estimated using the skin-colour model described in Appendix C. Equations 2.22 and 2.23 illustrate how the variances are constrained. α_{min}, α_{max}, β_{min}, and β_{max} are the minimum an maximum borders for the variances along the chromaticity r and g axis.

$$\sigma_{rr1}^2 = \begin{cases} \alpha_{min}, & area < \alpha_{min} \\ \alpha_{max}, & area > \alpha_{max} \\ \sigma_{rr}^2, & \text{otherwise} \end{cases} \tag{2.22}$$

$$\sigma_{gg2}^2 = \begin{cases} \beta_{min}, & area < \beta_{min} \\ \beta_{max}, & area > \beta_{max} \\ \sigma_{gg1}^2, & \text{otherwise} \end{cases} \tag{2.23}$$

To constrain the rotation of the Gaussian model we simply demand that the covariances σ_{rg}^2 and σ_{gr}^2 are negative or zero. This is done in Equation 2.24 and 2.25.

$$\sigma_{rg1}^2 = \begin{cases} 0, & \sigma_{rg}^2 > 0 \\ \sigma_{rg}^2, & \text{otherwise} \end{cases} \tag{2.24}$$

$$\sigma_{gr1}^2 = \begin{cases} 0, & \sigma_{gr}^2 > 0 \\ \sigma_{gr}^2, & \text{otherwise} \end{cases} \tag{2.25}$$

When all the variances and covariances have been constrained, we need to update σ_{gr1} and σ_{rg1} to reflect the changes which may have happened to σ_{rr} and σ_{gg}. This is done according to Equations 2.26 and 2.27.

$$\sigma_{rg2}^2 = \sigma_{rg1}^2 \sqrt{\frac{\sigma_{rr}}{\sigma_{rr1}}} \sqrt{\frac{\sigma_{gg}}{\sigma_{gg2}}} \tag{2.26}$$

$$\sigma_{gr2}^2 = \sigma_{gr1}^2 \sqrt{\frac{\sigma_{rr}}{\sigma_{rr1}}} \sqrt{\frac{\sigma_{gg}}{\sigma_{gg2}}} \tag{2.27}$$

Constraining LUTs

Handling too small or too large variances and covariances for LUTs is also possible. First of all, we need to calculate the variances for the chromaticity r and g values, and the covariance along the rg axis. This is done in the same way as for Gaussian models which was shown in Equations 2.8 to 2.11 on page 12.

Afterwards, two of the rules used to constrain the Gaussian model are verified. These are to ensure that σ_{gg}^2 is smaller than σ_{rr}^2 (shape) and that the variances along the chromaticity r and g axis are between minimum and maximum borders (area size). We have decided not to constrain the rotation of a LUT.

The actions taken when the rules of shape and area size are confirmed differ from the actions made on Gaussian model. This is because LUTs hold all the likelihood values, whereas Gaussian models calculate them based on their parameters. It is not possible to change a LUT simply by changing the values of it variances or covariances. Instead we use dilation and erosion to increase and decrease the sizes of the variances and the covariances until they are above or below a border value.

If $\sigma_{gg}^2 > \sigma_{rr}^2$ we erode the LUT along the chromaticity g axis until $\sigma_{gg}^2 \leq \sigma_{rr}^2$. Erosion along the chromaticity g axis is done by changing each likelihood's value to the minimum value of its neighbours and itself. Since the chromaticity g axis is vertical, we look at the two nearest vertical neighbours[3]. Equation 2.28 illustrates the actions taken to constrain σ_{gg}^2 to be smaller than σ_{rr}^2. S is the LUT and $erode_{gg}$ erodes S along the chromaticity g axis. After each erosion the LUT is normalized, and a new covariance matrix Σ^2 is calculated for the LUT, and the value of σ_{gg}^2 in this is used to find out if it is necessary to erode the LUT again.

$$lts(S) = \begin{cases} lts(erode_{gg}(S)), & \sigma_{gg}^2 > \sigma_{rr}^2 \\ S, & \text{otherwise} \end{cases} \tag{2.28}$$

To control the area size of the skin chromaticity distribution indicated by σ_{rr}^2 and σ_{gg}^2, we either dilate or erode the LUT along the appropriate axis until it gets above the minimum border or below the maximum border. Dilation is the opposite of erosion, which means that instead of setting a likelihood to the lowest value of itself and its neighbours, it is set to the highest value. Equation 2.29 shows the actions to be taken to ensure that σ_{rr}^2 and σ_{gg}^2 are between minimum and maximum borders. When eroding and dilating along the chromaticity g axis, the two nearest horizontal neighbours are used instead of the two nearest vertical neighbours. α_{min}, α_{max}, β_{min}, and β_{max} are the minimum an maximum borders for the variances along the chromaticity r and g axis. After each erosion the

[3] We also refer to [31] for more theory about dilation and erosion.

LUT is normalized and new variances calculated. The values of these are used to check whether another dilation or erosion of S is necessary.

$$lta(S) = \begin{cases} lta(erode_{rr}(S)), & \sigma_{rr}^2 > \alpha_{max} \\ lta(dilate_{rr}(S)), & \sigma_{rr}^2 < \alpha_{min} \\ lta(erode_{gg}(S)), & \sigma_{gg}^2 > \beta_{max} \\ lta(dilate_{gg}(S)), & \sigma_{gg}^2 < \beta_{min} \\ S, & \text{otherwise} \end{cases} \tag{2.29}$$

When the variances of the LUT have been made of appropriate sizes, we need to make sure that the skin-colours not are represented by a low number of likelihoods with large values. This is done by ensuring that a minimum percentage of likelihoods is above the mean of the likelihoods. The mean is calculated according to Equation 2.30 where M is the number of likelihoods that have a value above 0 and N is the total number of likelihoods. $S(\mathbf{x}_i)$ returns the likelihood value at position $\mathbf{x}_i = [r\ g]^T$ in the LUT S.

$$lmean(S) = \frac{1}{M} \sum_{i=1}^{N} S(\mathbf{x}_i) \tag{2.30}$$

Afterwards, we check if the percentage of likelihoods $B(S) > lmean(S)$ is higher than a minimum value β. If not, we raise the values of the likelihoods which are above 0 and below the mean with a constant value γ. In Equation 2.31 and 2.32 the actions to be taken to ensure a minimum percentage of likelihoods above the mean are summarized. $lmean_s$ is the original likelihood mean of the LUT S.

$$S_{add}(\mathbf{x}_i) = \begin{cases} S(\mathbf{x}_i) + \gamma, & 0 < S(\mathbf{x}_i) < lmean_s \\ S(\mathbf{x}_i), & \text{otherwise} \end{cases} \tag{2.31}$$

$$ltr(S) = \begin{cases} ltr(S_{add}), & B(S) > \beta \\ S, & \text{otherwise} \end{cases} \tag{2.32}$$

2.6 Focus of Attention Conclusions

In this chapter, methods for the initial FoA of a video conferencing system have been described. If the RGB colours of an image with a single illumination colour are normalized, human skin-colours are located in a very small area. This can be used to make an effective distinction between skin-colours and non skin-colours in the image. Two ways of representing skin chromaticity distributions have been investigated. Lookup tables (LUTs) and Gaussian models. LUTs consist of likelihoods of skin-colours in chromaticity plane and are calculated using values from skin-coloured areas, which could e.g. be faces that are being tracked. A likelihood image is made based on the LUT and an input image. This image can afterwards be thresholded to remove pixels with a very low likelihood of being skin-coloured.

The skin chromaticity distribution is close to normal according to [39, 37]. Therefore, a Gaussian model can be used to describe skin-colour likelihood. A Gaussian skin-colour model can be estimated from a set of training images or from images acquired at run-time from the objects being tracked. Comparing LUTs and Gaussian models, shows that they

produce more or less the same results. According to [24] LUTs should be slightly more accurate and computationally faster than Gaussian models.

If the LUT or Gaussian model is estimated off-line, problems will occur as soon as the correlated colour temperature (CCT) changes. This is because the skin chromaticity distribution moves when the CCT changes. This movement follows the skin locus, which is almost placed along the Planckian locus of Blackbody radiators. A system capable of coping with changes in CCTs must therefore be able to adjust itself to these changes. To update the LUT or Gaussian model to reflect the skin-colours under the current CCT, the skin-coloured areas of the objects being tracked can be used. The LUT can be updated using either *simple update* or *ratio update*. Both methods calculate a new LUT of skin-colour likelihoods of the objects being tracked. The simple method then updates the LUT used for skin-colour detection, using a constant between 0 and 1 to control how fast the adaption should occur. The ratio method divides the LUT made from the objects being tracked with a LUT of the whole image, and uses the result of this to update the LUT used for skin-colour detection. The advantage of doing this, is that colours which are highly represented in the input image will only have little effect on the likelihoods in the updated LUT.

Two ways to update a Gaussian model to reflect the position of the skin chromaticity distribution were also explained. These are *Gaussian ratio update* and *weighted parameters of Gaussians*. The ratio method makes heavy use of the ratio method for LUTs. It simply calculates a LUT holding the likelihoods in the Gaussian model divided by the likelihood of its center. Updates then occur as with a normal LUT but afterwards a new Gaussian model is estimated from the LUT. This new Gaussian model is then again used to update the LUT to hold its likelihoods divided by the likelihood of its center. The weighted parameter method first calculates a new Gaussian model using the chromaticities of the objects being tracked. Based on the parameters of this model, the model used for skin-colour detection is updated. This is done in the same way as for LUTs by using constants to weight how fast the adaption to new values should occur.

To avoid that regions of interest (ROIs) holding many non skin-colours make the calculated position and area of the skin chromaticity distribution reflect something else than reality, constraints can be made on the moments of the LUTs and Gaussian models. In Appendix C we have investigated and verified a skin-colour model invented by Störring *et al.* in [27]. These investigations showed that areas of skin chromaticity distributions are of certain sizes. Therefore, minimum and maximum values of the variances along the chromaticity r and g axes can be used to avoid situations where the areas get too small or too large. Furthermore, the variance along the chromaticity r axis is nearly always larger than the variance along the chromaticity g axis. This can be used to constrain the size of the chromaticity g variance to be less than or equal to the chromaticity r variance. The skin chromaticity distribution also rotates clockwise when moving to the right along the chromaticity r axis. Using a minimum constraint of always having clockwise rotation of the skin chromaticity distribution should therefore be a good idea. This can be ensured by always making the covariances 0 or negative. According to the investigations made in Appendix C, the position of the center of mass of the LUT or the Gaussian model can be also be constrained to be within a certain distance of the skin locus. If the center moves further away it can simply be relocated in the Gaussian model. In the LUT all the likelihoods also need to be moved by the same offset as the center of mass. This is because LUTs hold all the likelihood values, whereas the Gaussian models calculate them based on their parameters.

Chapter 3

Face Verification

In this chapter methods to use for face verification are described. The likelihood image generated by the focus of attention phase is preprocessed and segmented into objects of face candidates. Based on size, shape, solidity, template matching using an average nose-eye template, and ellipse matching each face candidate is classified as face or non-face.

3.1 Introduction

In the previous chapter it was explained how the detection of skin-colours can be used for directing the focus of attention (FoA) to areas in the images which are of further interest. In Section 3.2 we describe how a number of preprocessing operations followed by contour segmentation are used to group the skin-colour likelihood image into a list of *face candidates*. Afterwards, in Section 3.3, face candidates which are not face-like in size and shape are removed using a rectangular size and shape filter. The solidity of the face candidates which made it through this filter is then computed – i.e. the solidity of faces will be high since most parts of them are skin-coloured. Solidity is explained in Section 3.4. Following that, the remaining face candidates are controlled using nose-eye template matching. This is done in Section 3.5. Finally, a method for detecting ellipses in gradient images is applied to the face candidates which also survived the nose-eye template matching. This method is explained in Section 3.6.

Figure 3.1 illustrates the face verification process. As it can be seen, the face verification methods are organized serially and in order of increasing complexity[1]. This is because it supports the computational efficiency of the system, since the simple methods (the rectangle and solidity filters) remove the face candidates which are far from being face-like. In this way, only the face candidates which have a close resemblance with faces are used as input to the more complex methods of template and ellipse matching.

It is more important not to identify anything else than faces as faces, than to find all the faces in all the images. This is because the face candidates which are verified as faces are used to start new face trackers. These new trackers have influence on the areas used to search for hand-raises in the already active face trackers (we refer the reader to Chapter 5 on page 51 for more about hand-raise detection). Therefore, if something else than a face is classified as a face (also known as a *false positive*), it may cause that hand-raises are missed. The face verification methods must therefore be made strict enough to avoid that (preferably) any false positives are made. When doing this, the number of faces that are

[1] Although it can be argued whether the template matching or the ellipse matching is the most complex.

identified as non-faces (also known as *false negatives*) will inevitably raise. However, a high number of false negatives is not that big a problem, because we use videos of 12.5 Hz. I.e. if the number of false positives is e.g. 75% it would still be possible to detect a face in every fourth image or more than 3 times per second. The time used to find the face of a participant in a video conference would therefore be almost not noticeable.

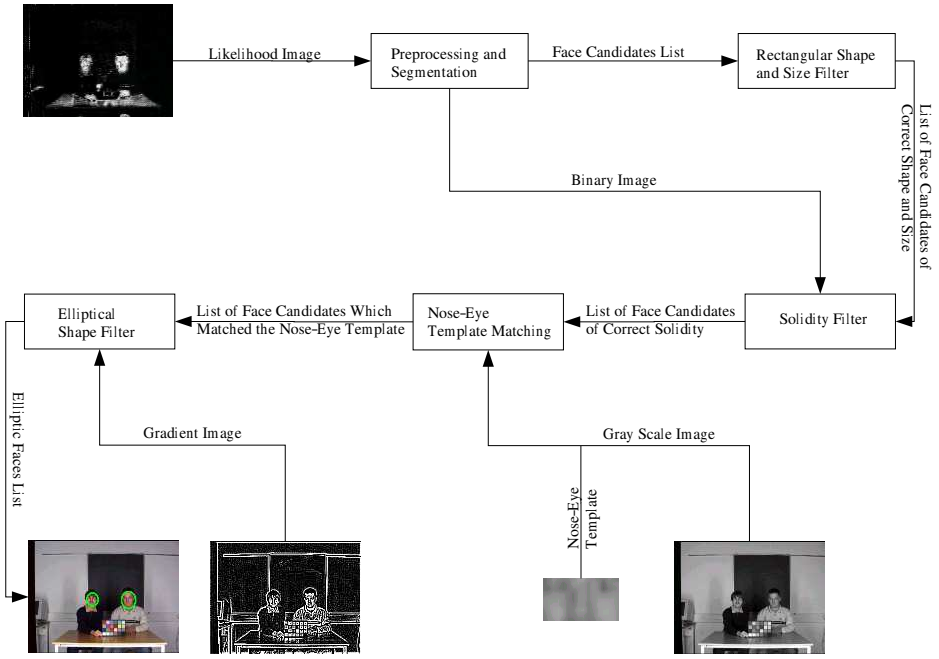


Figure 3.1: The Face Verification Process: Based on the skin-colour likelihood image made in the focus of attention phase, face candidates are found and verified. The resulting list of faces are illustrated as green ellipses in the original input image.

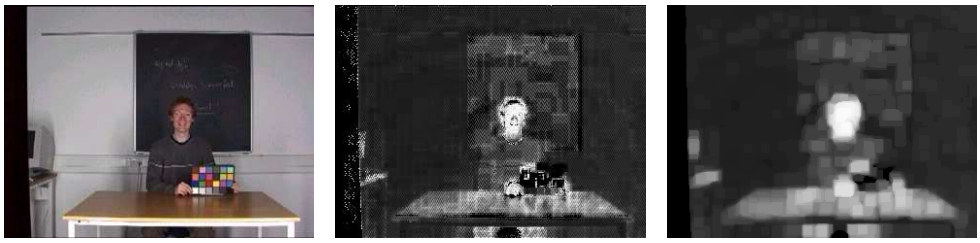
3.2 Preprocessing and Segmentation

To enhance the areas of high likelihoods in the skin-colour likelihood image, the morphological operations erosion and dilation can be used. First, the use of erosion can remove isolated likelihoods, and the low likelihoods which tend to be along the contour of faces. Afterwards, dilation can be used to enlarge the objects which are left after the erosion. Both methods also have the effect of quantizing groupings of similar likelihoods in the likelihood image into a lower number of gray levels (see Figure 3.2(b) and 3.2(c)). I.e. when erosion is used every pixel is set to the lowest gray level in its neighbourhood (in this case we have defined the neighbourhood as the 8 nearest neighbours, also known as 8-connectivity). Therefore, when applied enough times, the likelihood image will eventually consist of only one gray level – the lowest in the image. For dilation the opposite happens, i.e. a pixel is set to the highest gray level in its 8-connected neighbourhood. Therefore, if dilation is used enough times, the likelihood image will eventually consist of only the gray level of the highest likelihood in the image.

When using erosion and dilation in reasonable[2] combinations, the result will be a likelihood image where groupings of high likelihoods are raised and groupings of low likelihoods are lowered (see Figure 3.2(c)). Afterwards, a simple threshold method can be

[2]We have found a combination of $1\times$ erosion and $4-6\times$ dilation to be reasonable in this project. This is when images of 320×240 pixels are used and erosion and dilation is done using 8-connectivity

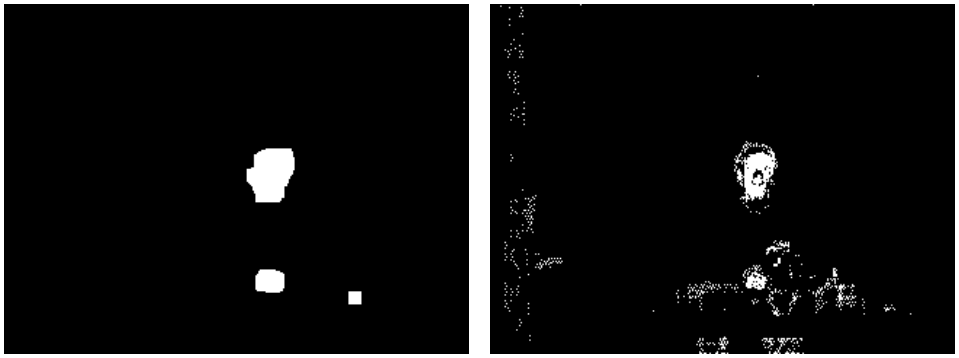
used to make a binary image where white pixels are skin and black pixels non-skin (see Figure 3.2(d)). This image will then be used as input to the segmentation method.



(a) Input Image

(b) Likelihood Image

(c) Dilated and Eroded Likelihood Image



(d) Thresholded Dilated and Eroded Image

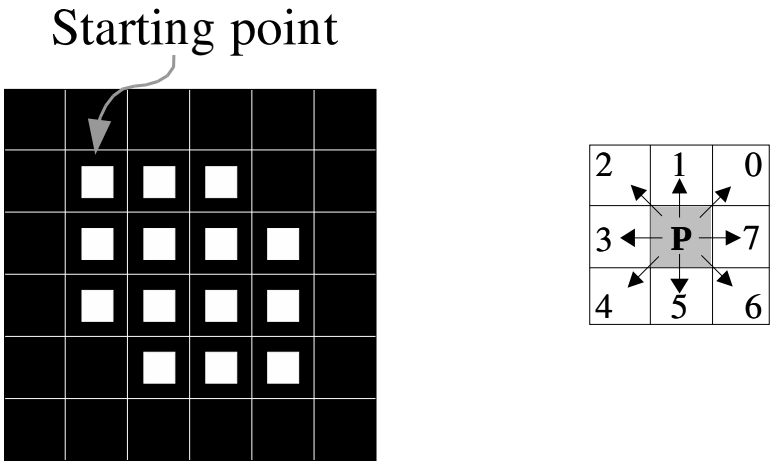
(e) Thresholded Likelihood Image

Figure 3.2: Preprocessing of Likelihood Image: The FoA phase generates the likelihood image (b) based on the input image (a). Using dilation and erosion (c) before thresholding the image makes it possible to remove noise and make clearer distinctions between the groupings of likelihoods. In this case 1 time of erosion and 4 times of dilation has been used. Comparing the result of thresholding (a threshold of 200 was used) the likelihood image with (d) and without (e) the use of dilation and erosion, should confirm the advantages of using erosion and dilation.

3.2.1 Segmentation

To segment the binary image generated by the preprocessing methods into objects of face candidates, we use a contour segmentation method [1]. The method is only used to find the bounding boxes of the face candidates, i.e. we do not use the information of the contours to anything. An alternative method to use for the same purpose, could be the connected components method described in [12].

The input image can be divided into foreground (white) pixels and background (black) pixels. Only white pixels are considered for the contours since those represent skin-colours. To describe the contours we use *chain codes* [12]. For any pixel we can enumerate all its neighbours with numbers from 0 to 7 (see Figure 3.3(b)). These numbers are used to indicate where the next pixel along the contour should be found. I.e. 2 for upper left, 4 for lower left, 7 for right, etc. (see description of the object in Figure 3.3). Looking at the 8 nearest neighbours when searching for the next pixel of an contour is also referred to as using *8-connectivity*. This is one of the two common sorts of connectivity. The other is *4-connectivity* where only the 4 nearest neighbours are being considered.



(a) Object (b) Chain Code Directions

Figure 3.3: Chain Codes: To describe the contour of an object (a) we use a chain code, where numbers (b) indicate where to find the next pixel along the contour. In this case the object (a) has the chain code: 7765533211.

To find the contours in the image we start in the upper left corner and work through all pixels from left to right, top to bottom (also known as raster scanning). Every time a white pixel is encountered, we mark it as unavailable and initiate a new chain code object and insert the position of the pixel (known as the *starting point* of the contour) into it. Thereafter, the chain code of the contour is generated by looking for white, available pixels using 8-connectivity. If any is found and it is 8-connected to at least one black pixel (i.e. it is at the border of an object), this new pixel is marked as unavailable and the direction of it is added to the chain code. Next, we do again look for available, white pixels using 8-connectivity and if any is found it is marked as unavailable and added to the chain code. This continues until no available, white pixels are in the neighbourhood of a pixel. If this pixel is 8-connected to the starting point of the contour, we have a valid description of a contour and add it to a list of contours. Otherwise, the contour is discarded.

Hereafter, we return to the raster scanning of the image and search for a new available, white pixel which can be used to initiate a new contour. During the raster scanning we make sure to mark white pixels as unavailable if they are 8-connected to a white, unavailable pixel. This is because such pixels are bound to be inside an already detected contour and therefore they cannot be used as initiators of new contours. In Figure 3.4 a pseudo code algorithm for contour segmentation can be seen.

When the raster scanning is finished, the contour list will hold all the valid contours in the image. Based on the starting point and the chain code of each contour, a rectangle which exactly spans the contour is found. These are then inserted into a list of rectangular face candidates (refer to Figure 3.1, where it can be seen that the outcome of the preprocessing and segmentation is a *rectangular faces list*). This list is then used as input to the next phase of face verification where the size and shape of the faces will be investigated.

```
For every pixel x in binary image
  if x = white and x = available
    x = unavailable
    if all 8-connected pixels to x are available
      create contour object
      starting point = position of x

      while any 8-connected pixel c to x is white and available
            and any 8-connected pixel to c is black
        chain code = chain code + direction of c
        x = unavailable
        x = c
      endwhile

      if x is 8-connected to starting point
        insert contour object into contour list
      else
        delete contour object
      endif
    endif
  endif
endfor
```

Figure 3.4: The Contour Segmentation Algorithm.

3.3 Rectangular Size and Shape

This method takes as input the face candidates found by the contour segmentation method explained in last section. A rectangle is spanned around each of these regions (see Figure 3.5(c)). Afterwards, rectangles that are larger than e.g. 100 pixels or smaller than e.g. 20 pixels in width or height are removed (these limits should suit the images of 320×240 pixels which we have used in this case, but will of course be different if the resolution is changed). Furthermore, rectangles where the relation between width and height are not between e.g. 0.6 and 2.0 (these values will have to be determined empirically) are removed. An example of the result of these limitations can be seen in Figure 3.5 (the Gaussian model explained in Section 2.4 on page 11 were used to detect the skin-colours in the input image).

3.4 Solidity

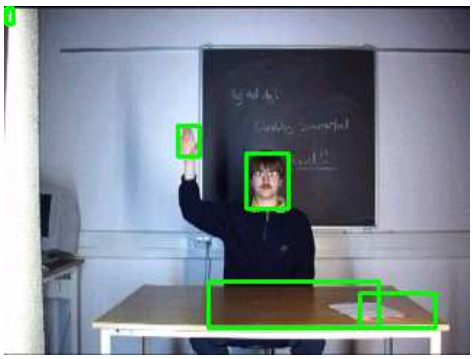
Characteristic for objects containing faces is that they will contain many skin-coloured pixels. Therefore, a face candidate received from the rectangular size and shape method can be verified by calculating the ratio of its area to the size of its bounding box. This can also be referred to as the *solidity* of the face candidate [2]. The area is calculated using the binary image made by the preprocessing methods and is simply the number of white pixels covered by the bounding box of the face candidate. The Equation for the solidity of a face candidate is therefore as illustrated in Equation 3.1, where A is the area and w and h the width and height of the bounding box.

$$solidity = \frac{A}{wh} \tag{3.1}$$

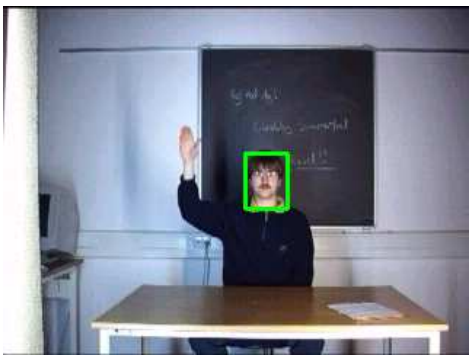
A face candidate is removed if its solidity is below a threshold α or above a threshold β. It is also possible to define a maximum solidity of faces because they normally are elliptic in shape. Henceforth, only a part of the bounding box should be covered by the face. The maximum solidity of a face should therefore never be able to reach 1.

(a) Input Image

(b) Binary Image Created using a Gaussian Model and the Preprocessing Methods



(c) Regions Found using Contour Segmentation



(d) Regions left after Size and Shape Verification

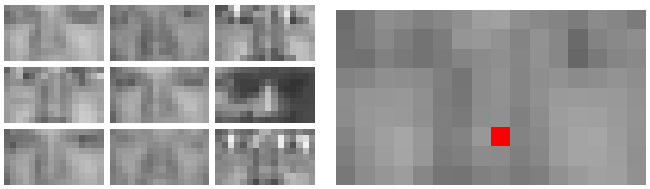
Figure 3.5: Removing Regions of "Wrong" Sizes and Shapes: skin-colours in the input image (a) are found using a Gaussian skin-colour model. The resulting likelihood image is eroded, dilated, and thresholded into a binary image (b). Afterwards, the contour segmentation method segments the image into the regions in (c). The result of using size and shape verification on these regions can be seen in (d). Notice how e.g. the regions containing the skin-coloured table have been removed due to their "wrong" relations between width and height. Furthermore, the hand and the object in the upper left corner are removed because they are too small to be faces.

3.5 Nose-Eye Template Matching

Nose-eye template matching is the process of classifying a face candidate as a face or non-face, based on how similar it is to an average template (the *reference template*) made from a number of nose-eye cut-outs (see Figure 3.6). The reference template is made out of cut-outs from gray-scale images. Therefore, we also convert the input image to a gray-scale image, and use this for the template match. Only the areas in the image which are holding a face candidate are used for the matching process. To match the reference template with a position inside one of the face candidates in the input image (a *candidate template*), it is placed with its center on top of this position, and a distance measure is calculated. If this measure is below or above a threshold[3] the reference template is said to match the candidate template.

Normally, template matching is done by calculating a distance measure for every pixel

[3]Whether it should be above or below a threshold depends on the method used for distance measuring.



(a) Examples of Nose-Eye Cut-Outs

(b) The Average Nose-Eye Template

Figure 3.6: Nose-Eye Template: The average nose-eye template (b) is created from a number of cut-outs of the nose-eye area of different persons (a). The red point indicates where the local maximum used as the center of the template is placed.

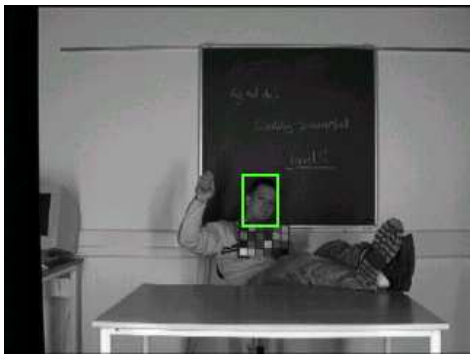
in the object (in our case a face candidate) to be matched with the reference template. According to [32], the use of a gray-scale nose-eye template can nevertheless speed up the matching process. This is because the tip of the human nose in most cases holds a local maximum – i.e. a pixel that is 8-connected with pixels of lower intensities. Setting this pixel as the center of the reference template, we only need to calculate distance measures for the local maxima in the part of the gray-scale input image covered by the face candidate. In general the number of local maxima in an image is many times lower than the number of pixels altogether (see Figure 3.7). Therefore, the time spend on calculating local maxima comes back many times, because most of the template matches can be skipped.

3.5.1 Image Pyramids

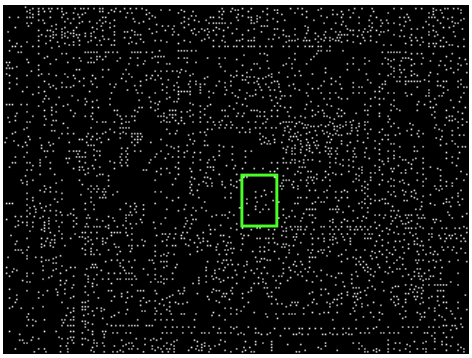
To be able to detect faces of different sizes we generate an image pyramid. This is simply a pyramid consisting of the gray-scale input image scaled at different sizes. I.e. the bottom of the pyramid holds an image of the same size as the input image and then the images get smaller and smaller the higher the pyramid is climbed. We have used a pyramid of 6 images, zooming out in steps of 10% of the original image. This means, that the image at e.g. the third level in the pyramid (the bottom of the pyramid is level 0) is scaled to 70% of the original image.

When doing the template match we search through all the levels in the pyramid to find the level with the best match. The scale of this level is afterwards used to calculate the size of the face in the original image. This is done by using simple reasoning about the position and size of the face when we know where the nose-eye area is. Using the pyramid we should be able to find faces with nose-eye areas as small as the reference template and up to four times the size of the reference template (i.e. twice the width and twice the height of the reference template).

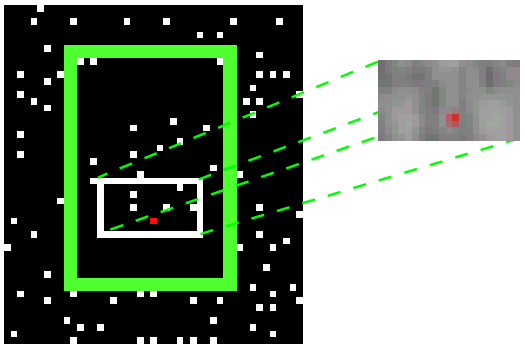
Since the reference template we use is an average of upright faces, we are not able to detect faces when they are rotated in either plane or depth. Since we demand that people must look straight into the camera for the tracking of them to be initiated (refer to the delimitation in Section 1.4 on page 4), this should nevertheless not be a problem. To handle faces rotated in plane we could use rotated versions of the reference template. This would though increase the computation time of the template match significantly.



(a) Gray-scale Input Image



(b) Local Maxima in the Image



(c) Matching with the Nose-Eye Template

Figure 3.7: Template Matching Using Local Maxima: Using nose-eye templates we only need to match on local maxima (white pixels in (b)) in the gray-scale input image (a). A match is made by placing the local maximum in the nose-eye template above a local maximum in the input image and calculate the distance between them (c). In this case the number of local maxima in the whole image were approximately 4.5% of the number of all pixels. The green rectangles indicate the face candidate area

3.5.2 Distance Measuring

To measure the distance between the reference template and a candidate template, a number of methods can be used. Among these are *Sum of Absolute Differences* (SAD), *Normalized Cross Correlation* (NCC), and *Zero Mean Normalized Cross Correlation* (ZM-NCC). These will be described in the following sections.

Sum of Absolute Differences

To calculate the SAD between a reference template T with center (x'_c, y'_x) and size $w \times h$ and a position (x, y) in the image I, Equation 3.2 is used. The result is a value that indicates the total distance in intensities between the reference template and the candidate template. I.e. the larger the value the worse the match. A value of 0 indicates a perfect match.

Although the method is incredible simple and computationally efficient, it has the disadvantage of being affected by changes in light intensity. I.e. a reference template would not be very similar to the exact same face as it was taken from, if e.g. the face is shadowed.

$$SAD(x,y) = \sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} | \, T(x',y') - I(x + (x' - x'_c), y + (y' - y'_c)) \, | \tag{3.2}$$

Normalized Cross Correlation

The NCC is far from being as simple as the SAD but has the advantage of including statistical measures in the final result of the match. The pixel intensities $T(x', y')$ in the reference template are multiplied by their corresponding intensities $I(x + (x' - x'_c), y + (y' - y'_c))$ in the candidate template. Afterwards, the result is normalized by the *absolute variances* of the reference- and candidate template. The absolute variance expresses the variances of the absolute pixel values regardless of their mean value.

Compared to the SAD, the NCC takes into account the absolute variances before the correlation result is delivered. This has the effect of lowering the result of templates, which otherwise would be given a high correlation. This is because the result are normalized by the absolute variances, such that the relative distance between a good match and a bad match is lowered.

The NCC measure at position (x, y) is given by Equation 3.3 and returns a value $0 \leq NCC(x,y) \leq 1$, where 1 indicates a perfect match.

$$NCC(x,y) = \frac{\sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} T(x',y') I(x + (x' - x'_c), y + (y' - y'_c))}{\sqrt{\sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} T(x',y')^2 \sum_{y'=0}^{h-1} \sum_{x'=0}^{w-1} I(x + (x' - x'_c), y + (y' - y'_c))^2}} \tag{3.3}$$

Zero Mean Normalized Cross Correlation

Using ZMNCC the *relative variance* instead of the absolute variance is used for normalization. The relative variance for a pixel in a template is found by squaring the difference between its intensity and the mean intensity of the template. Furthermore, the intensities in the denominator in Equation 3.6 are subtracted the mean intensity of either the reference template, $\bar{T}$, or the candidate template, $\bar{I}$. Equations 3.4 and 3.5 are used to define the value of intensities subtracted their mean values.

ZMNCC therefore has the advantage of being invariant towards changes in light intensity, as long as the intensity change is the same for all pixels in the candidate template. This is because it matches by looking at relative distances instead of absolute differences (see Figure 3.8).

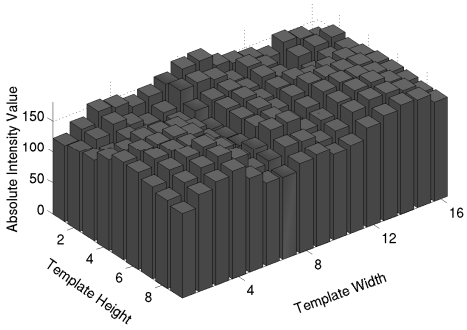
The value of the ZMNCC measure at position (x, y) is given by Equation 3.6 and returns a value $-1 \leq ZMNCC(x,y) \leq 1$, where 1 indicates a perfect match and -1 a perfect mismatch.

$$\tilde{T}(x,y) = T(x,y) - \bar{T} \tag{3.4}$$

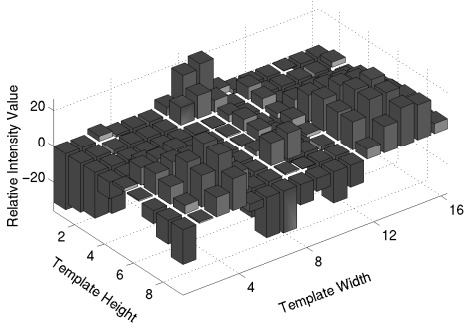
$$\tilde{I}(x,y) = I(x - x'_c, y - y'_c) - \bar{I} \tag{3.5}$$

$$ZMNCC(x,y) = \frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{T}(x',y')\tilde{I}(x+x',y+y')}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{T}(x',y')^2\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1}\tilde{I}(x+x',y+y')^2}} \tag{3.6}$$

(a) A Nose-Eye Template



(b) The Absolute Differences



(c) The Relative Differences

Figure 3.8: Absolute and Relative Distances: The absolute distances (b) of a template (a) are simply the value of its intensities. The relative distances (c) are found by subtracting the average intensity of the template from all its pixel intensities. An overall change in lighting intensity would change the absolute distances but not the relative distances. Using relative distances is therefore invariant towards changes in lighting intensity, which is not the case for absolute distances.

3.6 Elliptic Shapes

This method is based upon [4] and assumes that a face can be described as a vertical ellipse having a minor axis of σ (see Equation 3.7) and a fixed aspect ratio of 1.2. Using the face candidate list received from the nose-eye template match method and a gradient image of the input image, the best fitting ellipse to each of the regions can be found. Together with a threshold this can be used to find out whether the object inside the rectangle is a face or non-face. If so, it is added to the final list of elliptic faces which is used by the tracking methods described in Chapter 4. Describing the faces using ellipses instead of rectangles does also increase the accuracy, because faces in general are elliptic shaped.

3.6.1 Calculating the Fit of an Ellipse

To calculate the fit of an ellipse the method in [4] uses the normalized sum of the gradient around the perimeter of the ellipse:

$$\phi_g(s) = \frac{1}{N_\sigma} \sum_{\mathbf{i}=1}^{N_\sigma} \mid n_\sigma(\mathbf{i}) g_s(\mathbf{i}) \mid, \tag{3.7}$$

where $g_s(\mathbf{i})$ is the intensity gradient at perimeter pixel $\mathbf{i}$ of the ellipse at location s, and N_σ is the number of pixels on the perimeter of an ellipse with size σ.

To find the position of each pixel on the perimeter, Equation 3.8 is used. This is a modified version of the formula describing an ellipse [17], where a and b indicates the sizes of the major and minor axis.

$$y = \sqrt{(1 - \frac{x^2}{a^2})b^2} \tag{3.8}$$

To calculate the positions, the value of x is initiated at the center of the ellipse and increased towards $+a$ (see Figure 3.9). For each value of x, y is calculated using Equation 3.8. This gives the positions of $\frac{1}{4}$ of the pixels on the perimeter. The rest can be found using simple mirroring in the major and minor axis.

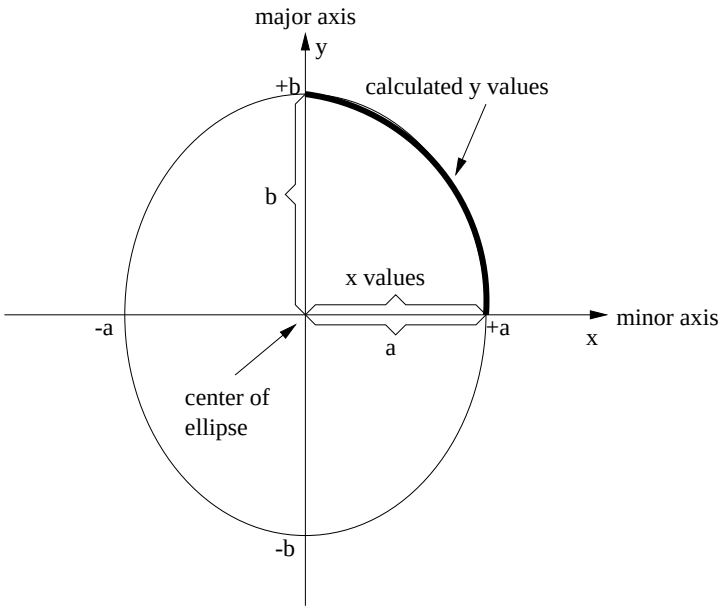


Figure 3.9: The Face Model: The face is described as a vertical ellipse. The positions of the pixels on the perimeter are calculated increasing the x position from 0 to +1. For each value of x the corresponding value of y is found. The rest of the positions are found by mirroring the x and y values in the major and minor axis.

3.6.2 Limiting the Search

The search for the best fitting ellipse is limited by a maximum, σ_{max}, and minimum, σ_{min}, size of the minor axis σ. Initially σ is set to half the width of the rectangle, i.e.

$$\sigma = \frac{width\ of\ rectangle}{2}$$

Afterwards, σ_{max} and σ_{min} are calculated according to Equations 3.9 and 3.10 where α is a constant used to control the size of the search area. Furthermore, α_{max} and α_{min} are used to ensure a maximum and minimum size of the ellipses that are matched.

$$\sigma_{max} = \begin{cases} \sigma\alpha, & \sigma\alpha \leq \alpha_{max} \\ \alpha_{max}, & \text{otherwise} \end{cases} \tag{3.9}$$

$$\sigma_{min} = \begin{cases} \frac{\sigma}{\alpha}, & \frac{\sigma}{\alpha} \geq \alpha_{min} \\ \alpha_{min}, & \text{otherwise} \end{cases} \tag{3.10}$$

3.6.3 Best Fit vs. First Fit

We have decided to soften the demand of finding the best fitting ellipse (referred to as *Best Fit*), such that we instead search for the first ellipse (referred to as *First Fit*) which have an average gradient sum above the threshold. This is done from $\sigma = \sigma_{max}$ and towards $\sigma = \sigma_{min}$. The reason for this softening is that many gradients often will be present in the center of a face. If the face is of a certain size, the best fitting ellipse will in most cases be too small – i.e. hold only the center of the face (see Figure 3.10(c)). When starting from the largest possible ellipse and searching inwards, we make sure that we get the largest ellipse, which have an average perimeter gradient above the threshold. Furthermore, the first fit method is computationally more efficient than the best fit method. This is because it stops searching as soon as the threshold is reached. The best fit method has to try out all possible ellipses before it can be sure, that it has found the best fitting ellipse.

(a) Input Image

(b) Gradient Image

(c) Ellipses Found using Best Fit

(d) Ellipses Found using First Fit

Figure 3.10: First Fit vs. Best Fit: Instead of finding the best fitting ellipse (c) in the gradient image (a), the first fitting ellipse (b) is found. This is done from max_σ towards min_σ such that the largest possible ellipse is found.

3.7 Face Verification Conclusions

In this chapter methods to be used for face verification have been investigated. The skin-colour likelihood image made by the FoA phase is preprocessed using the morphological operations erosion and dilation. Afterwards, the resulting image is thresholded and objects in the image are found using a contour segmentation method. The smallest possible rectangles spanning these objects are found and an initial list of face candidates is created.

Four methods are used to classify the face candidates as faces or non-faces. First, a rectangle filter is used to remove face candidates which are too large or too small or have wrong rectangular shapes (i.e. are long and thin). Afterwards, the solidity of the face candidates are found by calculating the ratio of their areas to the size of their bounding boxes. Because faces are elliptic in shape and have a high solidity, upper and lower solidity thresholds can be used to verify face candidates as faces or non-faces. Hereafter a nose-eye template matching method is applied to the remaining list of face candidates. This method speeds up the matching process by using the fact that there almost always is a local maximum on the tip of peoples nose. The nose-eye template is therefore only matched to the positions of local maxima. Only one template is used for the matching process. To handle faces of different scales, an image pyramid is made. Rotation in plane is not handled since one of the delimitations of the VICOWIJOY system is, that people must look straight into the camera without turning their faces too much, for the tracking of them to be initiated. Face candidates which look similar to the template are forwarded to the last method of verification. This is a ellipse detector which based on a gradient image finds either the best or first fitting ellipse to the face candidate. The detection is based upon counting the average intensity value along the ellipses perimeter. If this is above a threshold the ellipse is said to fit, and the face candidate is added to the final list of faces represented by the ellipses.

The face verification process is made as a serial combination of methods of increasing complexity. Combining them in this way should support the computational efficiency of the system, since most face candidates can be removed by the simple methods. The more complex methods are therefore only used on face candidates which have a close resemblance to faces.

Chapter 4

Face Tracking

In this chapter, we describe several ways of tracking faces. We then present the approach that we have chosen to use, which uses skin-colour likelihoods and intensity gradients to determine the position and size of a face, and a Kalman filter to handle the uncertainties related to these measurements. Finally, an algorithm for maintaining a set of trackers for the faces in the input images is described.

4.1 Introduction

We consider *tracking* of a face to be the maintenance over time of estimates of face parameters – such as position – given a number of measurements of the parameters at each time step. The parameters related to the face constitute a state vector. When tracking a face in an image sequence, the tracker must for each image:

- **Match** the predicted state vector of the face with a face in the current image and **measure** the parameters of that face from the image.
- **Estimate** the state vector from the measurements. This results in an *a posteriori* state vector estimate.
- **Predict** the state vector in the next image. This prediction, referred to as the *a priori* estimate, is usually necessary because the image may contain several other objects that – when only considering the parameters included in the state vector – look like the face being tracked (i.e. *clutter*). Furthermore, it can be used to reduce the search space necessary when matching.

Below, we will describe each of these steps further.

4.2 Matching

How the matching is done is of course highly dependent on the type of information that is included in the state vector. Obviously, the position and size would be relevant parameters to include in our case. The matching could then be done using template matching as described in Section 3.5. The face area from the previous image could be used as the reference template. The reference template is then matched to different parts of the

current image by computing a similarity measure at each position inside a search window placed around the predicted position of the object. The position with the highest similarity measure can then be considered the new position of the object.

Another approach would be to use a *snake* placed around the face. A snake is a deformable contour that is governed by interior and exterior forces. The interior forces ensure smoothness of the contour, while the exterior forces attract the contour to features in the image. Sobottka and Pitas [36] use snakes to track the contours of the skin-coloured regions caused by faces. A snake is placed on the image at the location where the face was detected, and the best fit of the snake is computed by minimizing a sum of internal and external energy terms. The matching is done by placing this snake on the next image in the sequence and repeating energy minimization to find the new shape and position of the snake.

Menser and Wien [2] track faces in colour image sequences by converting each image to a skin-colour likelihood image, which are analyzed at different threshold levels (i.e. the skin-colour likelihood image is thresholded with different thresholds, resulting in a set of binary images). When a face-like region has been found, matching is done by projecting the region into the next skin-colour likelihood image and, at all threshold levels, eliminating all connected components that are covered by less than a predefined rate of the projected region. Then connected components that differ more than a predefined degree from the projected region with regard to size of bounding box and center of mass are discarded. If more than one region is left, the region with the largest degree of compactness is chosen.

Schwerdt and Crowley [16] also use skin-colour likelihood images for the tracking of faces. They weight the likelihood images by placing a Gaussian function at the location where the face is expected and compute the center of mass of the likelihood image, which approximately – because of the weighting – is the center of mass of the face. The covariance of the Gaussian is estimated from the previous image. Initially, the covariance is the size of the expected face.

The Mean Shift algorithm [6, 1] can be used for matching based on skin-colour likelihood images (or other types of distributions). It considers a part of the image limited by a search window, which is centered at the predicted center of the face. Within this search window, the center of mass of the distribution is computed. The search window is then centered at this location and the new center of mass computed. This last step is repeated until the center of mass converges (or moves less than a predefined threshold). The center of mass found this way will be close to the center of the face (provided that only the face has high skin-colour likelihoods).

The CAMShift algorithm [6, 1] is an extension of Mean Shift. CAMShift is short for Continuously Adaptive Mean Shift, and it continuously adapts the size of the search window based on the zeroth moment[1] of the part of the likelihood image that is contained in the search window. Thus the object to be tracked does not need to have a constant size.

Birchfield [4] uses colour histograms for face and hair and intensity gradients to track faces. The head is modeled as position and size of an ellipse with a fixed aspect ratio. Matching is done – in a search space within a range of the predicted values – by maximizing the sum of a colour-based score and a gradient-based score. The colour-based scored is determined by comparing the colour histogram for the pixels inside the ellipse with a colour histogram for the subject produced off-line. The gradient-based score is determined by computing a normalized sum of the dot products of the gradient and the

[1] The zeroth moment of a region is the sum of all likelihoods or pixel intensities in that region.

unit vector normal to the ellipse for each pixel around the perimeter of the ellipse.

In a previous project [5], we used skin-colour and motion detection followed by connected components segmentation of a thresholded likelihood image to find and track faces and hands in a video conference situation. The matching was done by comparing the positions of the connected components with the predicted positions of the objects.

Our preliminary experiments with CAMShift indicated, as it could be expected, that using the zeroth moment to determine the search window size would not work very well when the skin-colour detection generated false positives for background pixels – the search window would suddenly grow and become too large, even though the face had much larger likelihoods than most non-face pixels. However, Mean Shift seemed to produce a very stable estimate of the center of the face, much more stable than the positions we found using connected components segmentation in our previous project. The size could then be found using ellipse fitting in a gradient image, as described in Section 3.6 on page 32, and the search window adjusted using this size.

We suspect that using Menser and Wien's connected components-based approach would produce the same amount of jitter as the somewhat simpler method we used in our previous project, and thus provide less stable estimates than Mean Shift.

Schwerdt and Crowley's method – Gaussian weighting of the likelihood image and computation of mean – will probably not perform much different from Mean Shift. Although it has the desirable property that skin-coloured objects near the face are weighted less than the face if the face is at the predicted position, it might be necessary, if the distance between predicted and actual position is large, to increase the variances of the Gaussian function to a level where it will perform equal to or worse than Mean Shift because a too large area in the image is used. Mean Shift does not need the entire face to be within the search window because it will iterate until the center of mass converges.

Using high-resolution snakes as Sobottka and Pitas do would be overkill, since information about the shape of the faces is irrelevant for our purposes. Besides, the shape of a face is quite constant. An ellipse is essentially a parameterized snake, and fitting this to the face will probably be more computationally efficient. Furthermore, allowing the snake to deviate much from the elliptic shape could be problematic when e.g. a hand is temporarily occluding the face during a hand raise. For this reason, the energy functions for the snake should be such that the snake approximates an elliptical shape, but then similar results could probably be obtained using an ellipse.

Doing ellipse fitting using both colours and gradients like Birchfield is rather expensive, computationally, and our preliminary experiments have indicated that using just the gradients is sufficient if the center of the skin-coloured area is found using Mean Shift.

Using templates will also be problematic in the case of occlusion, as it may be difficult to tell whether the poor fit is due to occlusion or e.g. the person turning his head.

Based on these preliminary experiments and thoughts, we have chosen to base the face tracking on the Mean Shift algorithm combined with ellipse fitting.

4.3 Estimation

Two estimators commonly used for tracking are the Kalman filter and the CONDENSATION algorithm (both are described in Appendix B on page 129).

The Kalman filter can be used for integrating several measurements into a single estimate

based on the measurements and the previous *a posteriori* state estimate, and for predicting the next state using a process model. When computing the estimate, it incorporates measures of the uncertainties of the measurements (the *measurement noise*), and the state estimates are accompanied by uncertainty measures as well (the *error covariance*). It also incorporates information about the accuracy of the process model (the *process noise*).

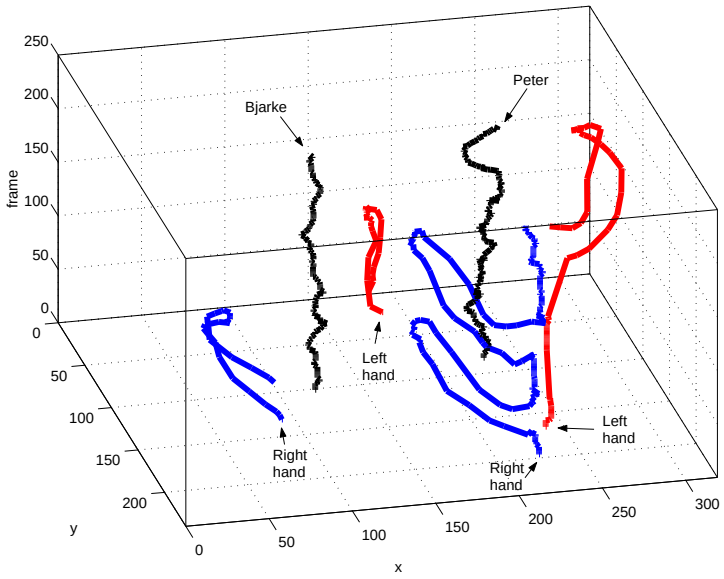
The Kalman filter is only able to represent unimodal state vector probability distributions as it only maintains estimates of the mean (i.e. the state estimate) and error covariance. Due to clutter and occlusion, the probability distribution may be multimodal [22]. If a wrong state at some point in time seems more likely than the true state, the tracker may loose the target, even though the true state soon would have become the most likely again. Therefore, it may be necessary to maintain several hypotheses of the target state to reliably track an object using a Kalman filter. This can be done by maintaining a bank of Kalman filters, where each filter is used to track a separate hypothesis. Cham and Rehg [40] track multiple hypotheses by maintaining a probability density using a piecewise Gaussian representation (i.e. the probability density at some point is determined by the Gaussian component that provides the largest contribution). At each time step, *a priori* estimates of the modes of the Gaussians are computed and used to obtain the *a posteriori* estimates of the modes through a state-space search.

The CONDENSATION algorithm has been designed specifically for multimodal probability distributions, and does not attempt to produce a single state estimate. Instead, it maintains a set of samples from the state vector probability distribution, from which e.g. the mean or the dominant mode can be used as the state vector estimate. If an uncertainty measure is desired, it can be derived from the sample set as well (the variance would be a good choice, as it will be low in the absence of clutter and occlusion).

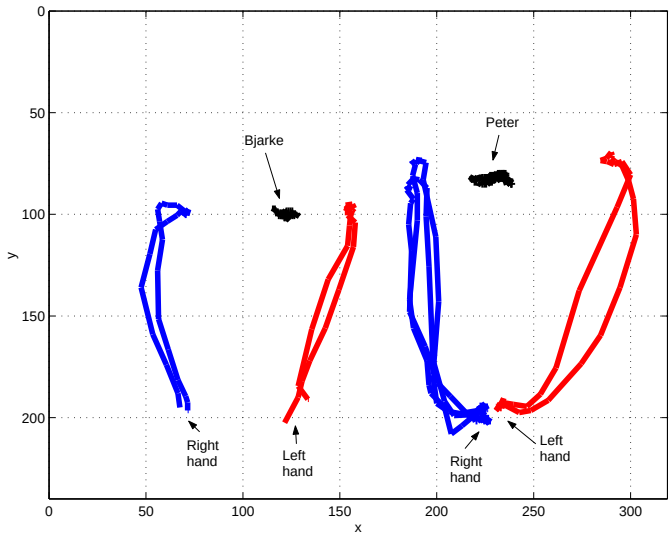
When tracking several faces, the probability distribution is most likely multimodal, because faces look very much alike. Hands also have some degree of similarity to faces, and are one of the main problems when tracking faces because they often appear near the faces, sometimes occluding the face. For this reason, it would be desirable with a tracker that is able to keep track of multiple hypothesis such as CONDENSATION. However, since the face will usually not move very much, an approach that we have found to be promising in our preliminary experiments is to use a single Kalman filter and let the amount of change in skin-colour likelihoods near the face determine how high the level of measurement noise should be when updating the Kalman filter. Thus, in the absence of clutter, the measurement noise will be low, and the Kalman filter will trust the measurements, and, when clutter appears, the measurement noise will be increased. We have chosen to use this approach for the face tracker, as it is likely to be more computationally efficient than using CONDENSATION or multiple Kalman filters.

4.4 Prediction

In a previous project [5], we found that the faces move very little during a video conference, as it can be seen by the trajectories in Figure 4.1, and that trackers predicting that the face had not moved performed better than trackers making predictions based on derivatives of the face position. Therefore, we will use a zeroth order Kalman filter, i.e. predict that the parameters have not changed This approach has also been used successfully by several others for tracking of faces and facial features [2, 36, 33].



(a) Trajectories of Faces and Hands



(b) Projection of Trajectories onto XY-plane

(c) Frame 0

(d) Frame 40

(e) Frame 150

Figure 4.1: Face and Hand Trajectories. (a) shows the positions of the faces and hands of the authors in an image sequence. The black trajectories are the positions of the faces. Three images from this sequence are shown in (c), (d), and (e).

4.5 Face Tracker

We model the face as an ellipse with a vertical major axis as the one shown in Figure 3.9 on page 33. The height/width-ratio is fixed to 1.2. Therefore, the state vector only needs to contain the center coordinates and one of the radii or diameters. We use a state vector containing the center coordinates and the width:

$$\mathbf{x} = \begin{bmatrix} x_c \\ y_c \\ w^{face} \end{bmatrix} \tag{4.1}$$

As mentioned in Section 4.2, two types of measurements are used to track the faces:

- Center of face found using the Mean Shift algorithm.
- Size of face found using ellipse fitting.

Furthermore, since this information is available at least some of the time, we will use:

- Face position and size from face detection module.

The center and size of the face found using Mean Shift and ellipse fitting are combined into a single measurement, which is passed on to the Kalman filter. If the face detection has found a face that has a certain degree of overlap with the tracker's estimate, the position and size of the detected face are incorporated in the tracker's estimate as well. This is described in detail below.

The update of the tracker, which is done for each image, is illustrated in Figure 4.2. First the skin-colour likelihood image is computed. This is done as described in Chapter 2, but with a skin-colour lookup table specific for the face being tracked.

The Mean Shift algorithm (grey box in Figure 4.2) is then used to find the center of mass of the face in the skin-colour likelihood image as described in Section 4.2. The search window is initially centered at the previous center of the face. The size $w_{t+1} \times h_{t+1}$ of the search window is determined from the Kalman filter's *a priori* estimate of the width of the face, w_t^{face}:

$$w_{t+1} = w_t^{face} + \beta z \tag{4.2}$$

$$h_{t+1} = 1.2 \cdot w_{t+1} \tag{4.3}$$

β is chosen such that the search window always is large enough to contain the entire face. z depends on the distance to the person[2], and is small when the person is far away and large when the person is close. Thus, when the person is close, the number added to the width of the face will be large, reflecting that movement of the head will produce a larger shift of skin-colour pixels in the image than when the head is far away. This is done because, preferably, the entire face should be inside the search window even when the window is centered at the previous position of the face, as it is likely that the Mean Shift algorithm otherwise will require more iterations. The multiplication with 1.2 when computing the height is due to the fixed height/width-ratio.

[2] z is a user-defined constant in our implementation, but could, in principle, be estimated from the images.

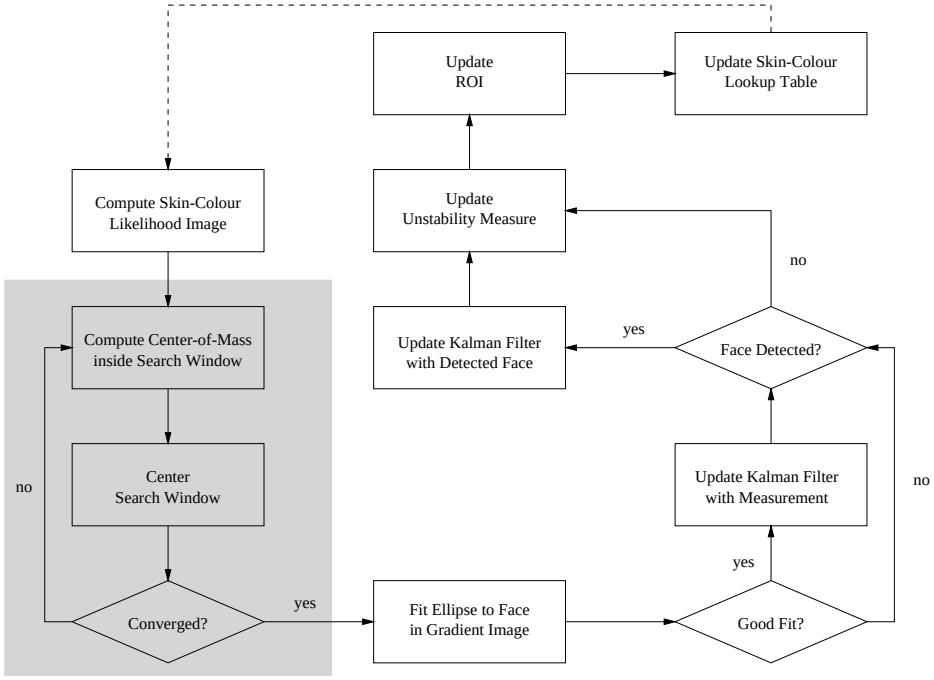
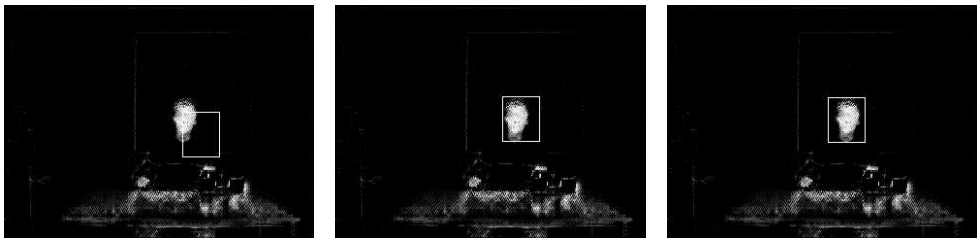


Figure 4.2: Tracker Update Cycle: The update of a tracker begins with the computation of the skin-colour likelihood image for the new input image and ends with the update of the skin-colour lookup table that will be used to compute the next likelihood image.

The Mean Shift loop in Figure 4.2 terminates when the search window moves less than a predefined distance or when the maximum number of iterations allowed have been done. The sequence of images in Figure 4.3 illustrates how the Mean Shift algorithm finds the center of mass in a likelihood image.



(a) Initial position (b) After 1 iteration (c) After 2 iterations

Figure 4.3: Mean Shift Example: The Mean Shift algorithm iteratively finds the center of mass of the face by computing the center of mass and centering the search window at this position.

Using ellipse fitting as described in Section 3.6 on page 32, it is attempted to fit an ellipse to the face. The position found by the Mean Shift algorithm is used as center of the ellipse. If an ellipse can be successfully fitted to the face, the center found by Mean Shift and the size found by ellipse fitting are used to update the estimate of face position and size using a Kalman filter.

Since the faces move very little most of the time during a video conference, and since loosing the face if the person moves significantly (e.g. leaves) is acceptable, a constant, low process noise that only allows the face to move a few pixels is used for the Kalman filter. The measurement noise can then be made to vary from values that are smaller than

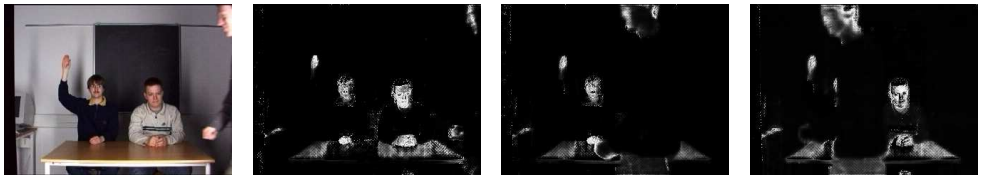
the process noise – when the conditions for measuring position and size are ideal – to values that are many times larger when the measurements are believed not to be reliable. To achieve this, we use a set for formulas for computing the measurement noise that we have determined empirically. In Section 7.4, we verify that they perform as desired.

When the face of a person is being tracked, an accumulated difference picture[3] (ADP) is computed for the hand-raise ROI of the person. This ADP is based on the skin-colour likelihood images, and will reflect the amount of skin-colour movement in the hand-raise ROI. While its primary purpose is detection of hand-raise gestures, as described in Chapter 5, it is also used when determining the measurement noise for the Kalman filter.

The measurement noise associated with the position and size measurements is based on several parameters:

- The zeroth moment M_t^{ADP} of the ADP for the hand-raise ROI of the person and the zeroth moment $M_t^{ADP,N}$ of the hand-raise ROI ADP for the neighbour to the right in the image.
- The difference between the zeroth moment of the Mean Shift search window in the skin-colour likelihood image and an average value for this moment. This difference will be referred to as ΔM_t.
- The difference between the variance of the Mean Shift search window in the skin-colour likelihood image and an average value for this moment. This difference will be referred to as $\Delta \sigma_t^2$.

The moments of the hand-raise ROI ADPs are used to increase the measurement noise when one of the persons raises his hand, when something skin-coloured is moving in the background, and when the skin-colour detection generates false positives (e.g. because of changing illumination). The moments of the Mean Shift search window in the skin-colour likelihood image are used to increase the measurement noise when the face gets occluded by a non-skin-coloured object and when the skin-colour detection does not generate a stable skin-colour likelihood image, e.g. because it is adapting to a new illumination colour. How the skin-colour likelihood images look when these events occur can be seen in the image sequences in Figures 4.4–4.5.



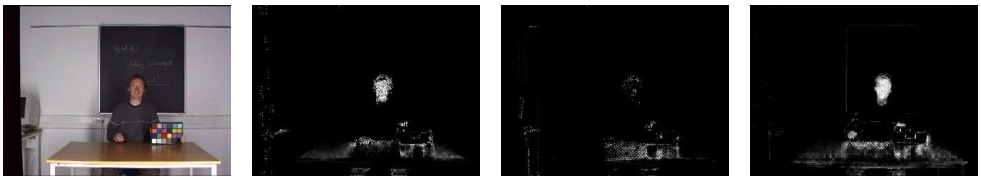
(a) Input (b) Before (c) During (d) After

Figure 4.4: Skin-Colour Likelihoods during Occlusion: The images illustrate how the skin-colour likelihoods for the face drop as the face becomes occluded.

The measurement noise

$$R = \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_3 \end{bmatrix} \tag{4.4}$$

[3] How the ADP is computed is described in Chapter 5 on page 51.



(a) Input (b) Before (c) During (d) After

Figure 4.5: Skin-Colour Likelihoods during Illumination Change: The images illustrate how the skin-colour likelihoods for the face drop temporarily when the illumination changes.

is computed as follows:

$$r_i = s_i(k_H H + F^2 + k_i z) \tag{4.5}$$

r_i contains a constant term $k_i z$, which has been included to ensure a minimum measurement noise level. Since this level should depend on the distance to the face, k_i is multiplied by the distance constant z described on page 42. Another constant is s_i, which is used to scale the rest of the expression to a suitable level. H and F are computed using the moments mentioned above.

As indicated by the i subscripts, each r_i may be associated with its own values of k_i and s_i. In preliminary experiments, we have observed that the ellipse fitting tends to fit the ellipse to two different outlines, see Figure 4.6. Sometimes, it will fit it to the edge between the top of the head and the background, and sometimes to the edge between hair and face. This varies from image to image for each person, and is probably related to movement or small changes in the skin-colour likelihood images. Since the output from the Mean Shift algorithm tends to be quite stable, it is desirable to associate a higher measurement noise with the size, found using ellipse fitting, than with the position. This can be done by using different values of k_i and k_i for the different entries in the measurement noise matrix.

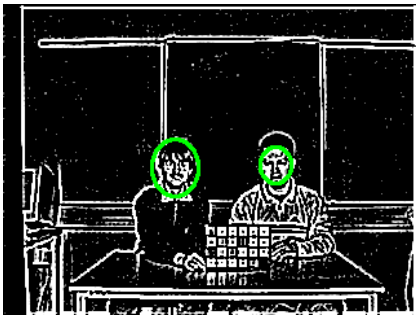


Figure 4.6: Result of Ellipse Fitting in Gradient Image

H is computed using the zeroth moments of the hand-raise ROI ADPs:

$$H = (\widehat{ln}(\frac{c_1}{N_1}\sum_{\tau=0}^{N_1}\frac{M_{t-\tau}^{ADP}}{A}))^2 + (\widehat{ln}(\frac{c_1}{N_1}\sum_{\tau=0}^{N_1}\frac{M_{t-\tau}^{ADP,N}}{A}))^2 \tag{4.6}$$

where c_1 is constant for scaling the result and A is the maximum area of the hand-raise ROI, i.e. if the hand-raise ROI have this size, the zeroth moment divided with A is the

average pixel intensity in the ADP. M_t^{ADP} is the zeroth moment of the hand-raise ROI ADP of the person being tracked, while $M_t^{ADP,N}$ is the zeroth moment of the hand-raise ROI ADP of the neighbour to the right in the image. The t subscripts indicate that these values depend on the time t. $\widehat{ln}(x)$ is a non-negative version of the natural logarithm:

$$\widehat{ln}(x) = \begin{cases} ln(x), & x > 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

The value of H is smoothed over time by using the average of the ADP moments over a period of N_1 images, as preliminary experiments suggested that this be beneficial. The purpose of the logarithm is to limit the value of H, such that there is a "saturation point" where a further increase in the sum of the moments does not significantly affect the value of the measurement noise. Using the constants c_1 in Equation 4.6 and k_H in Equation 4.5, it can be controlled when this saturation point will be reached, and what the contribution of H to the measurement noise should be at this point. Thus, a change in the zeroth moments will have a larger effect on the measurement noise when the value of the moments are low than when the values are high. This reflects an assumption that after some point has been passed, the magnitude of skin-colour likelihood change will not affect the probability that something that can confuse the tracker is happening.

F is computed from the zeroth moment and the variance of the Mean Shift search window after the final iteration of Mean Shift for the current image:

$$F = \frac{c_2}{N_2} \frac{\sum_{\tau=0}^{N_2} \omega_{M,t-\tau} \Delta M_{t-\tau}}{z^2} + \frac{c_3}{N_3} \frac{\sum_{\tau=0}^{N_3} \omega_{\sigma,t-\tau} \Delta \sigma_{t-\tau}^2}{z} \tag{4.8}$$

where c_2 and c_3 are scaling constants and

$$\Delta M_t = M_t - m_t^M \tag{4.9}$$

$$\Delta \sigma_t^2 = \sigma_t^2 - m_t^\sigma \tag{4.10}$$

where M is the zeroth moment of the skin-colour likelihood image inside the face search window, and $\sigma^2 = \sigma_x^2 + \sigma_y^2$ is sum of the variances of this area in the vertical and horizontal directions. m_t^M and m_t^σ are the current "average" levels for ΔM_t and $\Delta \sigma_t^2$, respectively, and are computed as follows:

$$m_{t+1}^M = (1 - \alpha) m_t^M + \alpha \Delta M_t \tag{4.11}$$

$$m_{t+1}^\sigma = (1 - \alpha) m_t^\sigma + \alpha \Delta \sigma_t^2 \tag{4.12}$$

Preliminary experiments indicated that using these averages instead of simply using the previous values or the unweighted averages for the entire image sequence produced a better measurement noise level during and after illumination changes.

The weighting functions $\omega_{M,t}$ and $\omega_{\sigma,t}$ in Equation 4.8 are computed based on the standard deviations of ΔM_t and $\Delta \sigma_t^2$, which are computed/adjusted dynamically. These standard deviations will be referred to as σ_M and σ_σ, respectively. The weights are computed using a zero-mean Gaussian function $G(x, \sigma)$:

$$\omega_{M,t} = 1 - \frac{G(\Delta M_t, \sigma_M)}{G(0, \sigma_M)} \tag{4.13}$$

$$\omega_{\sigma,t} = 1 - \frac{G(\Delta \sigma_t^2, \sigma_\sigma)}{G(0, \sigma_\sigma)} \tag{4.14}$$

The Gaussian function is:

$$G(x, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-x^2}{2\sigma^2}} \tag{4.15}$$

If the value of ΔM_t is low compared to the standard deviation σ_M, the weight $\omega_{M,t}$ will be low, and as the value of ΔM_t grows, the weight will become larger. The same holds for $\omega_{\sigma,t}$. This weighting scheme has been introduced to allow some amount of head movement without a significant increase in the measurement noise level. Note that it only affect the computation of F when ΔM_t and $\Delta \sigma_t^2$ are small, as the standard deviations are adapted to the average amount of movement, which is usually low.

If a face has been detected by the face detection module with approximately the same position and size as the face a tracker has been tracking in the past frames, it is assumed to be the same face. The position and size of the detected face is then used to update the Kalman filter's estimate, using the measurement noise given above. This way, the results of the face detection are used to guide the tracker if it is unable to find the face itself using Mean Shift and ellipse fitting. This could, for instance, happen because the illumination has changed too fast, or because clutter or occlusion has confused the tracker.

Each time the Kalman filter's estimate has been updated, a counter u is incremented. Moreover, a timer t_{upd} is reset to make it possible to measure how long time that has passed since the last update of the estimate. This information is used by the Tracker Manager described below.

To determine how much a person is moving, an *unstability measure* ξ_t is computed based on the results of Mean Shift and ellipse fitting:

$$\xi_{t+1} = (1 - a) \cdot \xi_t + a \cdot (|x_f - x_e| + |y_f - y_e| + |w_f - w_e| + |h_f - h_e|)/z \tag{4.16}$$

where x_i, y_i, w_i, and h_i are center coordinates, width, and height. x_f, y_f, w_f, and h_f are the values from the trackers *a posteriori* estimate, and x_e, y_e, w_e, and h_e the values found using Mean Shift and ellipse fitting. Note that when ellipse fitting did not succeed, the latter values have not been used to update the tracker; in this case it is likely that the distance between the *a posteriori* estimate and the values found using ellipse fitting will be even larger. $a \in]0, 1]$ is a constant that is used to regulate the sensitivity to changes in the distance between the measurements and the trackers *a posteriori* estimate.

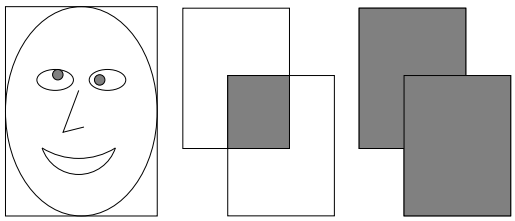
The unstability measure is used to determine whether it should be possible for the person that is being tracked to get the attention of the system by raising his hand. If ξ is large, it is likely that the level of noise is high or that the face is moving a lot, which is usually not the case when the person is seated and ready to speak. ξ is also used by the tracker manager described below to determine whether the tracker should be deleted. If a tracker is tracking a non-face object, e.g. a hand, it may have a large ξ compared to the usual values for faces. Thus, ξ may be used to discriminate between trackers tracking faces

and trackers tracking non-face objects. ξ is initially set to a large value to prevent the system from allowing the (hypothetical) person to speak. This way, the object that is being tracked must be spatially stable for some time, before the system accepts the object as a face. This helps eliminate trackers started because of false positives from the face detection.

4.6 Tracker Manager

The tracker manager maintains a set T of trackers for the faces in the input images. For each new image, it goes through the series of steps described below.

Let B_p denote set of pixels within the bounding box of the face tracked by the tracker $p \in T$, i.e. the set of pixels enclosed by the smallest rectangle containing the *a posteriori* estimate of the face ellipse. This is illustrated in Figure 4.7(a). Below, we will use $B_1 \cap B_2$ to denote the intersection of two such bounding boxes, as illustrated by the grey area in Figure 4.7(b). $B_1 \cap B_2$ is used to denote the union, which is the grey area in Figure 4.7(c).



(a) (b) (c)

Figure 4.7: Bounding Box Intersection and Union: (a) is the bounding box of a face, the grey area in (b) is the intersection of two bounding boxes, and the grey area in (c) is the union of two bounding boxes.

$|B|$, the size of B, is used below to denote the number of elements in the set B. If B is a bounding box, $|B|$ is the number of pixels inside this bounding box.

1. **Eliminate unstable and dead trackers.**

   A tracker p is considered unstable if its unstability measure ξ_p is larger than ξ_{max}. It is considered dead if its estimate has not been updated for θ_{time} images in a row. Thus, trackers fulfilling the condition

   $$(\xi_p > \xi_{max} \vee t_{upd} > \theta_{time}) \wedge u_p \geq u_{min}$$

   are deleted. The condition $u_p \geq u_{min}$ is included to allow trackers to have a large ξ in a short duration of time after they have been created. This is done, as the face of a person may be detected before the person is seated, and therefore initially may move relatively much.

2. **Eliminate overlapping trackers.**

   The estimates of a pair of trackers are not allowed to overlap more than a predefined degree. If it happens, the tracker with the largest unstability measure is deleted.

Let θ_{del} be the maximal allowable ratio between the size of the intersection and the size of the union of the bounding box estimates of two different trackers. Each pair of trackers $(p, q) \in T \times T$ is examined. If, for a pair of trackers (p, q),

$$\frac{|B_p \cap B_q|}{|B_p \cup B_q|} > \theta_{del}$$

then the unstability measures of the two trackers are compared. If $\xi_p > \xi_q$, p is deleted, otherwise q is deleted.

3. **Distribute detected faces to trackers.**

   If the face detection finds a face that overlaps more than a predefined degree with a tracker's estimate, the tracker is informed about the position and size of the detected face, which is used in the update cycle as described in the previous section.

   Let F be the set of faces found by face detection and let θ_{upd} be the maximal allowable ratio between the size of the intersection and the size of the union of the bounding box of the detected face and a tracker's bounding box estimate.

   For each $f \in F$ the pair $(f, p) \in F \times T$ that maximizes

   $$z = \frac{|B_p \cap B_f|}{|B_p \cup B_f|}$$

   is found, and if $z > \theta_{upd}$, the detected face f is passed on to the tracker p.

4. **Create new trackers.**

   If a face has been detected that has no overlap with a tracker's estimate, a new tracker is created for that face. That is, for each face $f \in F$, if there does not exist a $p \in T$ such that $|B_p \cap B_f| > 0$, then a new tracker is added to T for f.

5. **Update trackers with the positions of their neighbours.**

   Each tracker is associated with a hand-raise detector, which detects hand-raises for the person whose face is being tracked. This hand-raise detector considers an area to the left of the person – the hand-raise ROI – the width of which depends on the position of the face to the left of the face being tracked.

   To make it possible for each tracker to adjust the size of the hand-raise ROI of the associated hand-raise detector, it is informed about which tracker that is its left neighbour in the image. Moreover, as each tracker needs to know the zeroth moment of the hand-raise ROI to the right to compute the measurement noise, it is also informed about its right neighbour.

   Let x_i denote the estimated x position of the tracker i. The tracker $p \in T$ is the left neighbour of $q \in T$ if

   $$x_p \leq x_q \wedge (\neg \exists i \in T \ (x_p < x_i < x_q)) \wedge u_p \geq u_{min}$$

   Likewise, $p \in T$ is the right neighbour of $q \in T$ if

   $$x_p \geq x_q \wedge (\neg \exists i \in T \ (x_p > x_i > x_q)) \wedge u_p \geq u_{min}$$

   The condition $u_p \geq u_{min}$ helps to ensure that trackers that have been created for non-face objects – due to false positives from the face detection – do not disturb the other trackers.

If several choices for a left or right neighbour are possible – this can happen if two trackers have the same x estimate – one is chosen by the tracker manager. If no neighbour exists, the tracker is informed that this is the case. (If no left neighbour exists, the hand-raise ROI can assume its maximal width.)

6. **Update trackers with the current image.**

   Each tracker is updated with the current image. This causes each tracker to go through the update cycle described in the previous section.

7. **Determine if a hand-raise has occurred.**

   The tracker manager must inform the supervisor about hand-raises to make it possible for the supervisor to control the PTZ-camera that zooms in on the speaker.

   To determine whether one of the participants has raised his hand, all trackers $p \in T$ for which

$$\xi_p \leq \xi_{max} \wedge u_p \geq u_{min}$$

   are queried. The condition ensures that only stable persons that have been tracked for a while can get the attention of the system. This is done to reduce the risk that the system detects hand-raises for non-face objects and that the hand-raise detection generates false positives while a person is coming or leaving.

4.7 Face Tracking Conclusions

In this chapter, we have described the approach we use for face tracking. A tracker manager maintains a set of trackers for objects in the input images that are believed to be faces, and it continuously attempts to eliminate trackers for objects that are not faces from this set. Each face tracker tracks a face using a zeroth order Kalman filter. The measurement noise for this Kalman filter is determined based on changes in skin-colour likelihoods inside the face area and in the hand-raise ROIs to the left and right of the face. The measurements of the position and size of the face are done using the Mean Shift algorithm and ellipse fitting. The Mean Shift algorithm determines the center of mass of the face in the skin-colour likelihood image. Using this center, it is attempted to fit an ellipse to the outline of the face in a gradient image. If this can be done, the width of the ellipse is used together with the center of mass found using Mean Shift to update the Kalman filter. If the face detection (described in Chapter 3) finds a face that has a sufficient degree of overlap with the trackers *a priori* estimate of the face position, this detected face is also used to update the Kalman filter.

Chapter 5

Hand-Raise Detection

In this chapter, we describe how hand-raise gestures may be recognized in sequences of skin-colour likelihood images using a Naive Bayesian Classifier, and we estimate probability density functions for the attributes of this classifier.

5.1 Introduction

One way to recognize gestures is to track the hand, elbow, and shoulder. A gesture can then be defined as sequence of vectors containing e.g. the relative positions of these body parts, and gesture recognition can be seen as matching in the space of possible gestures. The matching of gestures can be done using e.g. Dynamic Time Warping, Hidden Markov Models, or CONDENSATION [23].

Azoz *et al.* [43] propose a combined 3D arm localization scheme and tracking framework that use colour, motion, and shape to localize the hand, elbow, and shoulder of a person. Colour segmentation is used to find the face and the hand, and using the positions of these together with the time-varying edges that movement of the arm produces in the image, the elbow location is determined. These locations are tracked using a Kalman filter (described in Appendix B). To improve the estimates, 3D distance constraints limiting the distances between the locations being tracked are incorporated in the estimate.

Moeslund and Granum [25] use a similar approach for localization, using kinematic and collision constraints to limit the search space. Using colour segmentation, the face and the hand are found. The elbow is then located in a silhouette image of the arm, which is produced by subtracting a background image.

Bernardo *et al.* [9] track the human arm in 3D using gray-scale images. The arm is modeled as shoulder and elbow joints and two truncated (i.e. the top is missing) cones representing the upper and lower arms. Using a Kalman filter, the pose of the arm is predicted. Matching is done by – for sets of parameters within a search window around the predicted values – projecting the arm onto the image plane and computing the distance to a thresholded and blurred version of the actual input image. When the best match has been found, it is used to update the Kalman filter, using the distance as prediction error.

Black and Jepson [23] use the CONDENSATION algorithm (described in Appendix B) for recognizing gestures performed holding a distinctively coloured object in hand. A gesture is modeled as a sequence of vectors containing the velocities of the object. Each state vector in the sample set maintained by the CONDENSATION algorithm contains a

number indicating which gesture model that is being matched, the position of the vector within the model that aligns the model and the input gesture at the current time step, and two parameters used to scale the model in time and space. From the sample set, a probability distribution indicating which gesture that has been performed can be derived.

Since we only need to be able to recognize one type of gesture, and do not need information about the exact position and pose of the arm, we have decided to try a somewhat different approach to gesture recognition based on changes in skin-colour likelihood images over time. In an accumulated difference picture based on the likelihood images, a hand-raise gesture will leave a vertical track. Using morphological operations and thresholding, this track is emphasized and turned into a single connected component, which then is classified using a Bayesian network. This approach is described in detail below.

5.2 Preprocessing

When the tracker has determined the position of the face of a person in the image, it is possible to set a region of interest (ROI) in the image to which the search for hand-raise gestures by that person can be limited. These hand-raise ROIs are shown with boxes in Figure 5.1.

The height and width of the hand-raise ROI are scaled according to the distance to the person[1]. As mentioned in Section 4.6 on page 48, information about the position of the neighbour is used to limit the width of the ROI such that the neighbour's face is not contained in the ROI. When we refer to the "maximum ROI" size below, we mean the size that the ROI will have if it is not limited by the presence of a neighbour.

Within the ROI of each person, the changes over time in the skin-colour likelihood images generated with that person's skin-colour LUT are determined using an accumulated difference picture (ADP). To achieve a less noisy ADP, the likelihood images are preprocessed using erosion with a 3×3 neighbourhood[2]. That is, for each pixel (x, y) in the likelihood image, the pixel itself and its eight neighbour pixels are considered. The value of the pixel with the smallest value is chosen as the value of the pixel (x, y) in the eroded image. To neutralize the effect of the erosion on the hand in the likelihood image, the erosion is followed by dilation using the same neighbourhood. Dilation is done in the same way as erosion, except that the maximum value is chosen [1]. To get a more connected track in the ADP, the dilation is repeated two times. The resulting image, that is used as input when computing the ADP, can be seen in Figure 5.1(c).

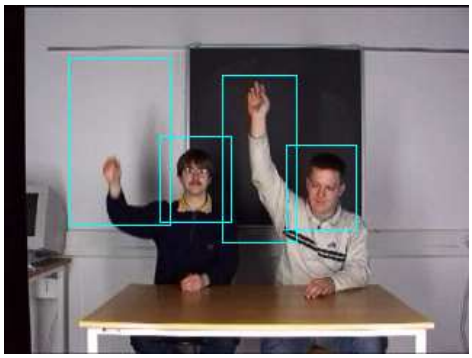
The ADP is computed using a spatio-temporal filter that compares each of the previous k images to a reference image, in our case the first image in the sequence, and increments the score/intensity of a pixel in the output image each time the difference between its value in the reference image and the other image exceeds a predefined threshold τ. This can be expressed recursively as follows [31]:

$$ADP_0(x, y) = 0 \tag{5.1}$$

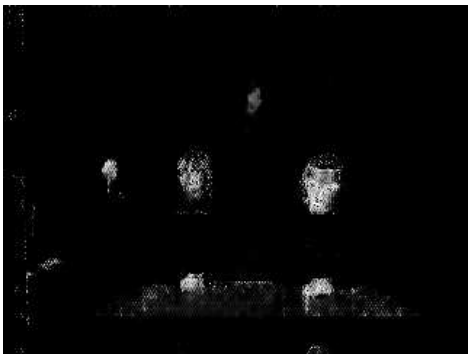
$$ADP_k(x, y) = ADP_{k-1}(x, y) + DP_{1k}(x, y) \tag{5.2}$$

[1]For this purpose, we use the predefined scaling constant z which was introduced in Section 4.5 on page 42, and do not attempt to determine the distance from the images.

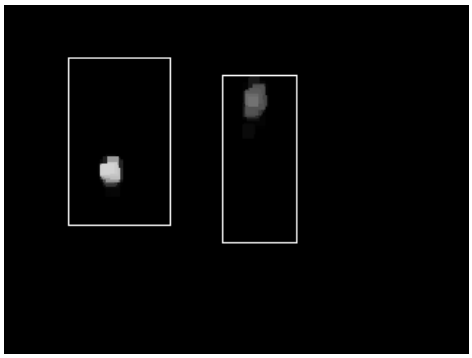
[2]The neighbourhood sizes presented here are designed for a resolution of 320×240 pixels. Their sizes should be adapted if the image resolution is changed significantly.



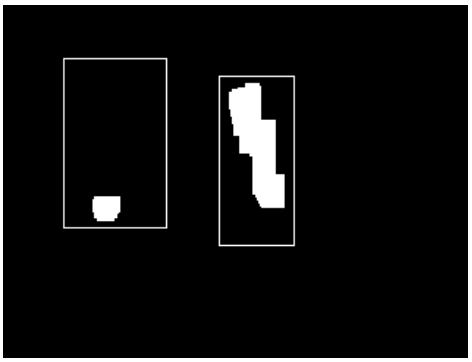
(a) Input Image



(b) Skin-Colour Likelihood Image



(c) ADP Input Image



(d) Final Image

Figure 5.1: Hand-Raise Detection Preprocessing. The large boxes are the hand-raise ROIs, the small boxes the search windows used by the trackers. The input image (a) is converted to a skin-colour likelihood image (b), which is processed with erosion and dilation to produce an input image for the ADP computation (c). The output for the ADP is processed further and thresholded to produce the image in (d), the contents of which is classified as being either a hand-raise or noise.

where DP is a binary image indicating for which pixels the difference in intensity between two images is more than τ. Let $I(x, y, i)$ be the intensity of the pixel at (x, y) in the i'th image in the sequence. Then DP can be computed as follows:

$$DP_{jk}(x, y) = \begin{cases} 1 & \text{if } |I(x, y, j) - I(x, y, k)| > \tau, \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

In our preliminary experiments, we have found $k = 9$ to be a suitable number of images to use for the ADP, and we will use this value when estimating the probability density functions for the Bayesian network. With a frame rate of 12.5 Hz, this number of images corresponds to approximately 0.7 seconds.

The ADP is thresholded, and to remove noise, it is eroded using a three pixels wide horizontal neighbourhood that consists a pixel and its immediate neighbours to the left and right. Using this neighbourhood appears to be sufficient to eliminate much noise, and it leaves the heights of blobs caused by a hand mostly unaffected.

This is followed by dilation using the same neighbourhood. The dilation is then repeated, which have the effect of making the blobs in the image wider than they originally were.

This is done to ensure that blobs due to hand movement have shared x positions when the hand has been moved horizontally as well as vertically. These blobs are now connected using the *vline* operation described in Figure 5.2, resulting in an image like the one presented in Figure 5.1(d). This image is segmented using contour segmentation as described in Section 3.2 on page 24, and the contour classified as described in the next section. If the hand-raise ROI contains more than one contour, only the contour with the largest bounding box is considered.

```
procedure vline(image, min_height, min_blobcount)
  for x := 0 to image.width-1
    blobheight := 0
    min_y := -1
    blobcount := 0

    /* count the number of times the vertical line through x intersects
       a blob which is at least min_height pixels high at this x position */
    for y := 0 to image.height-1
      if pixel at (x,y) is white
        flag := true
        blobheight := blobheight + 1
        if blobheight >= min_height
          if min_y = -1 then min_y := y
          max_y := y
        endif
      elseif flag = true and pixel is black
        if blobheight >= min_height then blobcount := blobcount + 1
        flag := false
        blobheight := 0
      endif
    endfor

    /* handle the case where the last pixel is white */
    if flag = true and blobheight >= min_height then blobcount = blobcount + 1

    if blobcount >= min_blobcount then draw line from (x,min_y) to (x,max_y)
  endfor
endprocedure
```

Figure 5.2: VLine Algorithm: This algorithm connects blobs in a binary image which have shared x positions. To reduce the risk that noise becomes connected, *min_height* is set to the minimum height of a blob caused by a hand, and *min_blobcount* is set to the minimum number of blobs with shared x positions that a hand-raise will leave in the ADP.

5.3 Classification

The contour in the hand-raise ROI is classified as being either the result of a hand-raise gesture or something else, henceforth referred to as noise. Noise could for instance be a person walking behind the video conference participants, hand movement that is not a hand-raise gesture, or noise due to changes in the illumination.

5.3.1 Attributes

Whether a person has raised his hand or not will influence several attributes of the contour found in the the hand-raise ROI, including:

- The **position** of the contour. When a person is raising his hand, it will to some degree follow a mean trajectory, while noise, e.g. due to a person moving in the background, will tend to appear other places as well in the hand-raise ROI. The

position used could be the center of mass of the area within the contour, or as we will do, the center of the bounding box of the contour, which will often be close to the center of mass.

- The **height and width** of the contour. The height of the contour will generally be larger than the width, and the area will be large compared to e.g. the head of the neighbour, which may occasionally be included in the hand-raise ROI because it has not been found yet by face detection or because the tracker fails. Area and height/width-ratio provide the same information as height and width, but by representing this information as area and height/width-ratio, information about size and shape is separated.

- The **number of skin-coloured pixels** in the upper part of the area delimited by the contour. When a person is raising his hand, the hand will be in the upper part of this area, and thus the number of skin-coloured pixels will be larger than when the hand is being taken down. By counting the skin-coloured pixels in an area larger than that of the hand, it can be determined whether the skin-colour detection is producing false positives for the background pixels. If this is the case, the number of skin-coloured pixels is likely to be larger than it usually is during a hand-raise gesture, and the contour may by due to noise rather than a hand-raise gesture.

- The **size of the intersection** of the pixel sets contained in the current contour and the contour found in the previous image, compared to the size of the current contour. While some types of noise may have intersection areas similar to hand-raise gestures, a small intersection area will be a good indication that the contour is not caused by a hand-raise gesture.

- The **direction of the movement** of the skin-colour center of mass in the previous N likelihood images within the bounding box of the contour. During a hand-raise gesture, this center of mass will move upwards.

The values of these attributes also depend on whether the person's arm is covered when the hand is being raised, or he has bare arms.

5.3.2 Two Naive Bayesian Classifiers

The classification is done using a Naive Bayesian Classifier (NBC). NBCs assume that the attributes are independent given the hypothesis, but even when this assumption does not hold, NBCs are competitive with state-of-the-art classifiers [28].

A NBC for hand-raise detection is shown in Figure 5.3. The direction of skin-colour movement is not used in the NBC, as it requires significantly more computation than the other attributes. Instead, it is used to verify the result of the NBC when it classifies something as a hand-raise gesture.

The hypothesis variable "Hand Raised?" (henceforth, H) can be in the states $h_1, ..., h_n$. If the other variables, henceforth referred to as information variables, are labeled $A_1, ..., A_k$, the probability that H is in the state h given the observation $A_1 = a_1 \wedge ... \wedge A_k = a_k$ is, according to Bayes rule [8]:

$$P(H = h \mid \bigwedge_{i=1}^{k} A_i = a_i) = \frac{P(\bigwedge_{i=1}^{k} A_i = a_i \mid H = h)P(H = h)}{P(\bigwedge_{i=1}^{k} A_i = a_i)} \tag{5.4}$$

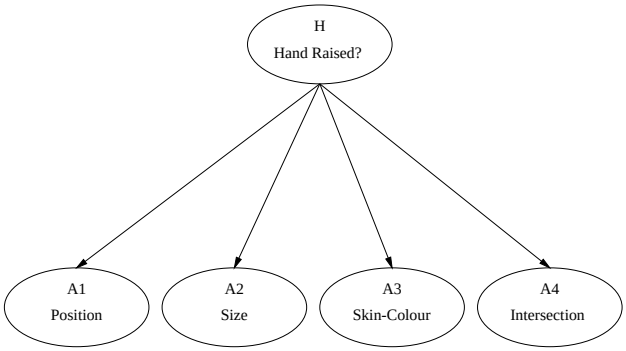


Figure 5.3: NBC1: A Naive Bayesian Classifier for Hand-Raise Detection.

$P(H = h)$ is the background probability of the hypothesis h. $P(\bigwedge_{i=1}^{k} A_i = a_i \mid H = h)$ is the probability of a particular observation given that $H = h$, and can, in principle, be estimated from data, but this is impractical because of the large state space.

If the events $A_1 = a_1, ..., A_k = a_k$ are assumed to be independent given the state of H, Equation 5.4 can be rewritten to:

$$P(H = h \mid \bigwedge_{i=1}^{k} A_i = a_i) = \frac{(\prod_{i=1}^{k} P(A_i = a_i \mid H = h))P(H = h)}{P(\bigwedge_{i=1}^{k} A_i = a_i)} \tag{5.5}$$

Since the probability $P(\bigwedge_{i=1}^{k} A_i = a_i)$ is the same for all h_i, it can be substituted with a normalizing constant z:

$$P(H = h \mid \bigwedge_{i=1}^{k} A_i = a_i) = \frac{(\prod_{i=1}^{k} P(A_i = a_i \mid H = h))P(H = h)}{z} \tag{5.6}$$

The probabilities $P(A_i = a_i \mid H = h)$ can be estimated from data.

The NBC in Figure 5.3 represents the x and y coordinates using a single information variable. Alternatively, this information could be represented using two variables, as in Figure 5.4. This has also been done for the size variable that contains the information about height and width of the contour. This variable can be split into a variable for the area and a variable for the height/width-ratio. By splitting these variables, it is assumed that the x and y positions are independent given the hypothesis, and that area and height/width-ratio are independent given the hypothesis. We will refer to the NBC in Figure 5.3 as NBC1, and the NBC in Figure 5.4 as NBC2.

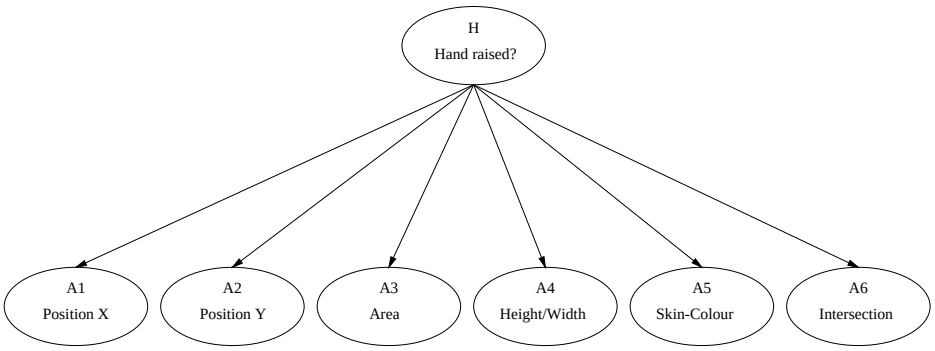


Figure 5.4: NBC2: Another Naive Bayesian Classifier for Hand-Raise Detection.

5.3.3 Estimation of Probability Density Functions

We estimate probability density functions (pdfs) for both of the classifiers presented in Figures 5.3–5.4. Since the values of the attributes depend on whether the person's arm is covered or not, the hypothesis variable can be in three different states:

- Hand-raise with **Arms Covered**
- Hand-raise with **Arms Bare**
- **Noise**

The histograms in Figures 5.5–5.10 show the frequencies of the various states of the information variables given the state of the hypothesis variable. They are based on a few hundred examples of contours resulting from hand-raise gestures and a few thousand examples of contours resulting from noise[3]. The data were generated automatically using, for some image sequences, a simple NBC with binary information variables, and for other sequences, a set of rules producing nearly identical classification. The number of false positives (i.e. noise classified as hand-raise gesture) is less than 0.5%. The small contours appearing during the first and last part of a hand-raise gesture were classified as noise.

By approximating the histograms with appropriate functions, e.g. Gaussian or piecewise Gaussian, and normalizing such that the area or volume beneath each function equals 1, pdfs providing reasonable probabilities for $P(A_i = a_i \mid H = h)$ can be obtained. It would not be appropriate to use the histograms directly, as only three persons participated in the video sequences, and each person only raised his hand a few times. Furthermore, many kinds of noise are not represented in the sequences. Thus, the data used does not provide sufficient information to make the histograms representative, and a functional approximation will probably result in a better mean performance.

Together with the histograms in Figures 5.5–5.10, Gaussian or piecewise Gaussian approximations are shown. The parameters of these functions are presented in Tables 5.1–5.2.

Attribute	**Hand-Raise**				**Noise**	
	Arms Covered		Arms Bare			
	μ	σ	μ	σ	μ	σ
Center of Mass X	-0.61	0.17	-0.80	0.21	-0.67	0.24
Center of Mass Y	-0.08	0.08	-0.16	0.10	-0.16	0.22
Area	0.15	0.08	0.33	0.16	0.00	0.12
Height/Width-Ratio	2.48	0.84	2.24	0.50	0.00	2.39
Intersection Area	1.00	0.16	1.00	0.13	0.00	0.03
					1.00	0.30
Skin Pixel Count	0.19	0.06	0.46	0.15	0.30	0.22

Table 5.1: Hand-Raise Attributes Mean and Variance. Two sets of numbers are given for the intersection area for noise, as a piecewise Gaussian approximation is used.

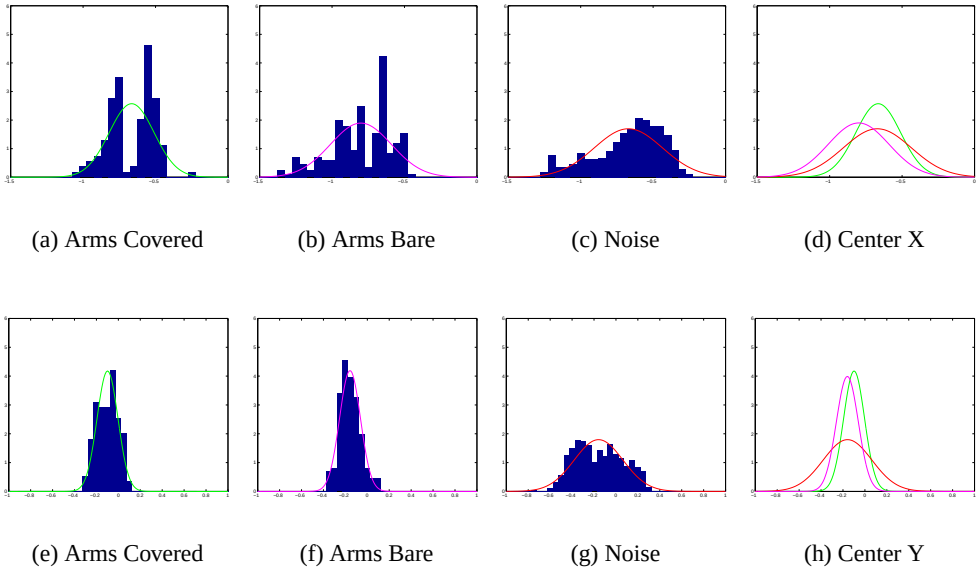
[3]The exact numbers are 112 examples of contours caused by hand-raises with covered arms, 175 examples of contours caused by hand-raises with bare arms, and 6092 examples of noise. The contours were primarily taken from V1–V5, V7, and V11. Contours from parts of V8–V10 and V19–V20 were used as examples of noise due to illumination change.

Attribute	Parameters	
	μ	Σ
X-Y Position, Arms Covered	$\begin{bmatrix} -0.0986 \\ -0.6647 \end{bmatrix}$	$\begin{bmatrix} 0.0091 & 0.0014 \\ 0.0014 & 0.0240 \end{bmatrix}$
X-Y Position, Arms Bare	$\begin{bmatrix} -0.1474 \\ -0.7248 \end{bmatrix}$	$\begin{bmatrix} 0.0086 & 0.0083 \\ 0.0083 & 0.0417 \end{bmatrix}$
X-Y Position, Noise	$\begin{bmatrix} -0.1584 \\ -0.6708 \end{bmatrix}$	$\begin{bmatrix} 0.0493 & 0.0025 \\ 0.0025 & 0.0553 \end{bmatrix}$
Width and Height, Arms Covered	$\begin{bmatrix} 0.5566 \\ 0.3842 \end{bmatrix}$	$\begin{bmatrix} 0.0101 & 0.0012 \\ 0.0012 & 0.0052 \end{bmatrix}$
Width and Height, Arms Bare	$\begin{bmatrix} 0.6086 \\ 0.4393 \end{bmatrix}$	$\begin{bmatrix} 0.0537 & 0.0263 \\ 0.0263 & 0.0171 \end{bmatrix}$
Width and Height, Noise	$\begin{bmatrix} 0.1909 \\ 0.2184 \end{bmatrix}$	$\begin{bmatrix} 0.0381 & 0.0197 \\ 0.0197 & 0.0206 \end{bmatrix}$

Table 5.2: Hand-Raise Attributes Mean and Variance for NBC1.

Position

The position is measured relative to the center of the face in a coordinate system with $(0, 0)$ in the upper left corner of the image, and the x and y coordinates are normalized by dividing with the maximum hand-raise ROI width and height, respectively. Thus, the x coordinates are always negative, and the y coordinates are negative when above the center of the face, and otherwise positive.



(a) Arms Covered (b) Arms Bare (c) Noise (d) Center X

(e) Arms Covered (f) Arms Bare (g) Noise (h) Center Y

Figure 5.5: Center Coordinates for Hand-Raise Gestures for NBC2.

The center y coordinates for hand-raise gestures in Figure 5.5 seem to be approximately normally distributed, and the histograms can be approximated with a Gaussian pdf by computing mean and standard deviation for the data. The y coordinates for noise can also be approximated with a Gaussian pdf.

For the x coordinates for hand-raise gestures in the case Arms Covered in Figure 5.5(a), the values do not seem to be normally distributed. This is because the hand-raises have

been done by two persons sitting next to each other. This constrains the x position of the hand for one of the persons, while the other person can move the hand freely. The result is a multimodal probability distribution. The persons were sitting quite close with an approximate center-to-center distance of 0.6 m. If the distance between the persons had been larger in some of the sequences, and more sequences had been used, the distribution might have become unimodal. For this reason, we approximate the distribution in Figure 5.5(a) with a single Gaussian pdf.

The x coordinates in the Arms Bare case in Figure 5.5(b) have a larger variance, but the histogram is otherwise quite similar and is also approximated with a single Gaussian pdf.

The pdfs for the center x coordinates are plotted in Figure 5.5(d), including the noise pdf. As can be seen from the figure, the x coordinate does not provide much information for classification, as the pdfs are very similar, with approximately same mean and standard deviation. The y coordinate provides more information, as it can be seen in Figure 5.5(h), where the probability at the means of the hand-raise pdfs are more than twice the probability of noise, due to the large standard deviation of noise compared to hand-raises.

In Figure 5.6, histograms and Gaussian approximations are shown for the three hypotheses in the case where the position is represented using a single variable. As it can be seen from the figure, the Gaussian functions are rotated because of nonzero covariances. This suggests that the position should indeed be represented as a single variable, as a similar effect could not be obtained by multiplying the x and y pdfs in Figure 5.5. Also note the large standard deviation for noise compared to hand-raises. As discussed above, this means that the position is useful for classification.

Height and Width

The height and width of the contour are normalized by dividing with, respectively, the maximum ROI height and width, and the area is normalized by dividing with the maximum ROI area. The area is computed as the area of the bounding box of the contour, i.e. by multiplying width and height.

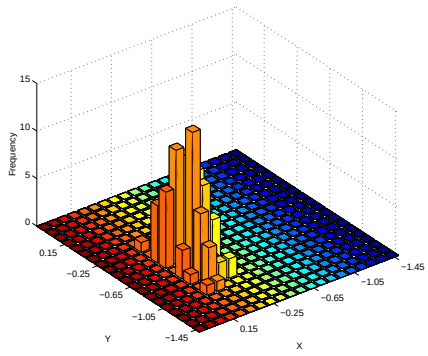
As can be seen from Figure 5.7, reasonable Gaussian approximation can be made for the hand-raise hypotheses for both area and height/width-ratio. The area when making hand-raises with bare arms tends to be somewhat larger, but also with a larger standard deviation. This is because of the larger skin-colour area that moves in the image.

The noise tends to have a small area and a small height/width-ratio compared to the hand-raise gestures. Simply making a Gaussian approximation by computing mean and standard deviation would not produce a good pdf, as the the smallest values of the attributes would not get the largest probabilities. Instead, a Gaussian approximation has been produced by mirroring[4] the data around 0 and then computing the mean and standard deviation. The resulting pdfs are not perfect fits, but seem acceptable.

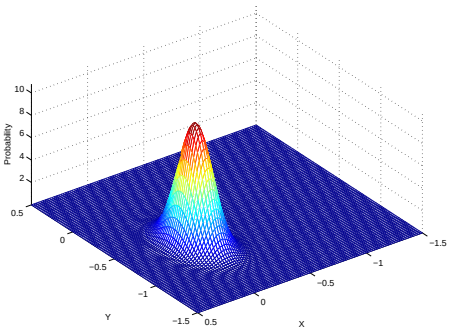
One might speculate that increasing the probabilities of noise for large areas and height/width-ratios could improve classification. However, due to the size and height/width-ratio of the maximum ROI[5], a contour with a large height/width-ratio will have a small area and vice versa.

[4]Let $h(x)$ be a function describing the histogram. This function is defined for $x \in [0; \infty[$. By mirroring around 0, we produce a function $h_m(x) = h(|x|)$, which is defined for $x \in]-\infty; \infty[$.

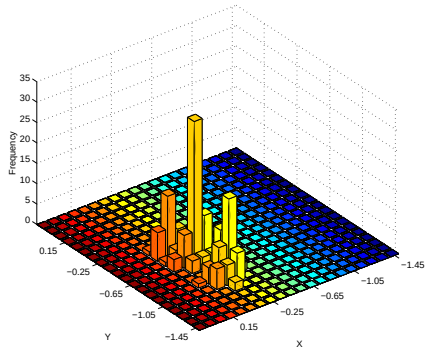
[5]We use a height/width-ratio of the ROI of 1.6 and an area of $8050z$, where z is a scaling constant depending on the distance to the person.



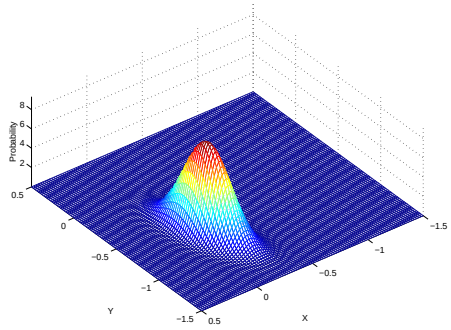
(a) Arms Covered



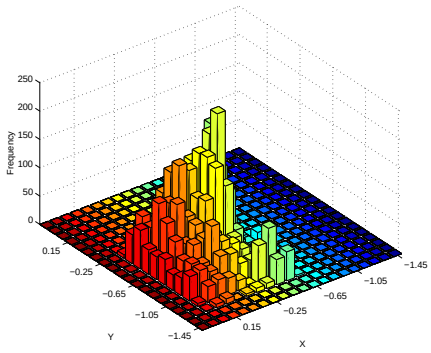
(b) Arms Covered



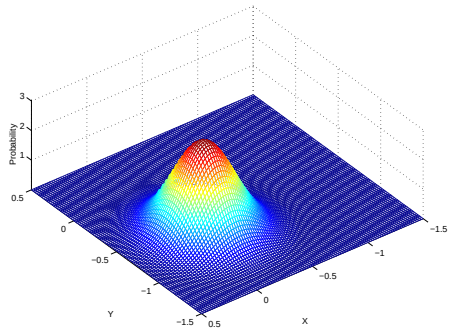
(c) Arms Bare



(d) Arms Bare



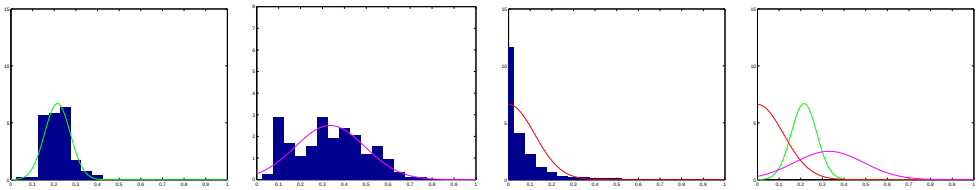
(e) Noise



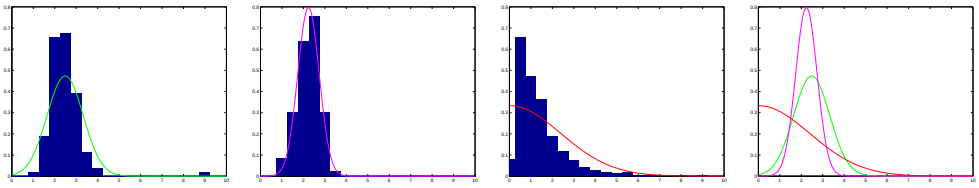
(f) Noise

Figure 5.6: Center Coordinates for Hand-Raise Gestures for NBC1.

When the size and shape is represented using a single variable, the Gaussian approximations to the height-width histograms in Figure 5.8 can be used as pdfs. Note that the shape of 5.8(d) and to some extent 5.8(b) indicates that contours with a particular height/width-ratio should have large probabilities. This confirms that it is sensible to use area and height/width-ratio, when the size and shape are modeled using two variables, instead of height and width.



(a) Arms Covered (b) Arms Bare (c) Noise (d) Area



(e) Arms Covered (f) Arms Bare (g) Noise (h) Height/Width-Ratio

Figure 5.7: Area and Height/Width-Ratio of Hand-Raise Gestures.

Intersection Area

The intersection area is determined as the percentage of the bounding box of the current contour that is covered by the bounding box of the previous contour. If no contour has been found for a few frames, it is assumed that the current contour is the first in a sequence of contours that will be caused by a hand-raise, and the intersection area is defined to be 1.

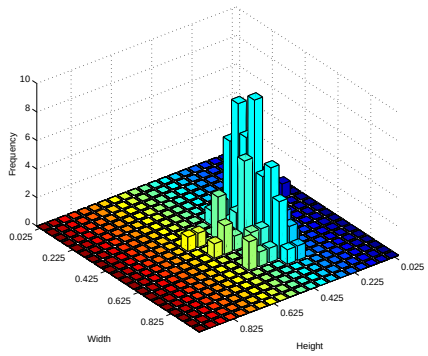
Histograms and Gaussian approximations for the intersection area are shown in Figure 5.9. The approximations for the histogram in Figures 5.9(a)–5.9(b) were produced by mirroring the data around 1 before computing mean and standard deviation. The noise in Figure 5.9(c) has been approximated using a piecewise Gaussian pdf. The data for the first piece of the pdf was mirrored around 0 and the data for the second piece was mirrored around 1.

Number of Skin-Coloured Pixels

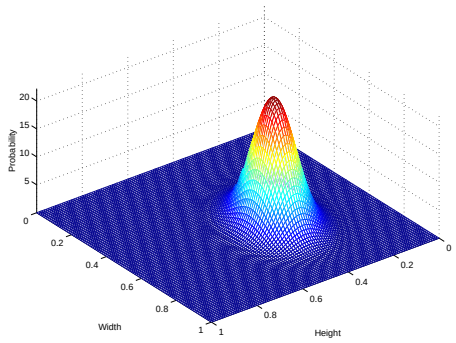
The number of skin-coloured pixels within a box placed at the upper part of the contour is determined, and normalized by dividing with the size of this box (which depends on the distance to the person). The histograms and Gaussian approximations for the values of this attribute are shown in Figure 5.10.

Verification using Skin-Colour Movement

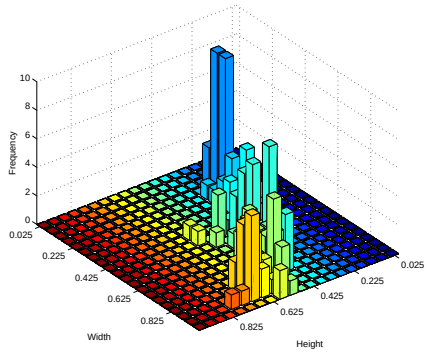
When a contour has been classified as a hand-raise by the NBC, a score based on the previous N skin-colour likelihood images is computed. This score is computed by determining, for each likelihood image, the center of mass inside the bounding box of the contour. These center of masses are considered in chronological order. Each time the center of mass moves more than ΔY pixels upwards, a counter u is incremented, and each time the center moves more than ΔY pixels downwards, another counter d is in-



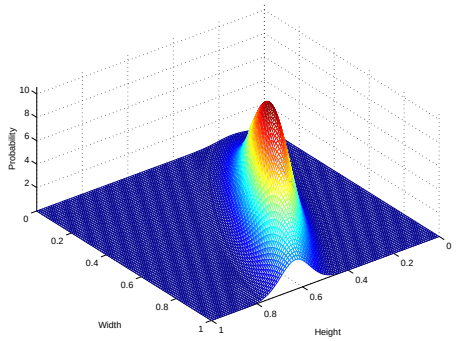
(a) Arms Covered



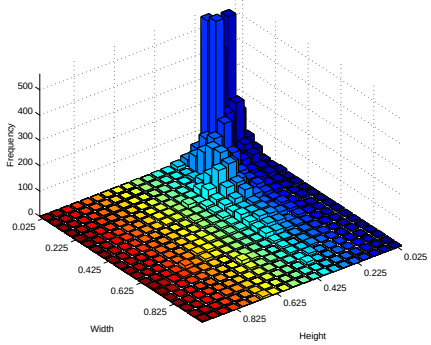
(b) Arms Covered



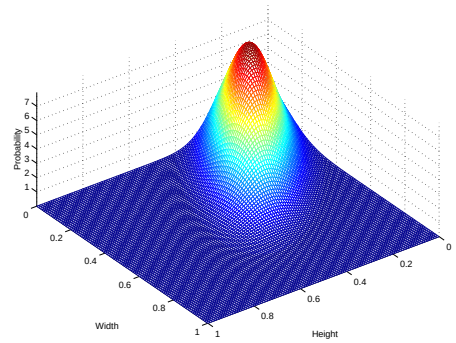
(c) Arms Bare



(d) Arms Bare



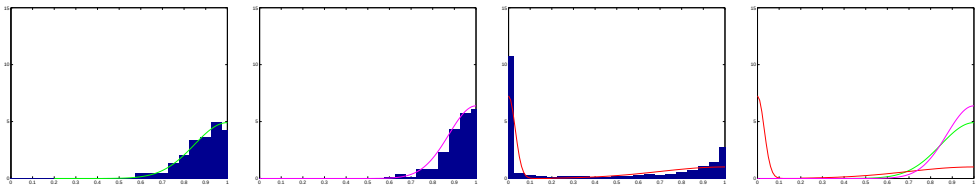
(e) Noise



(f) Noise

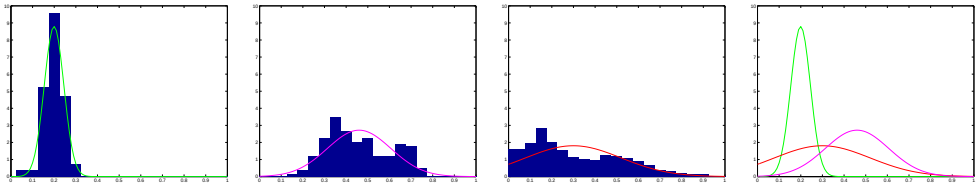
Figure 5.8: Height and Width of Hand-Raise Gestures.

cremented. The score is then computed as $(u - d)/N$, which will be a value between -1 and 1. If this score exceeds a threshold, the contour is accepted as being caused by a hand-raise.



(a) Arms Covered (b) Arms Bare (c) Noise (d) Intersection Area

Figure 5.9: Intersection Areas for Hand-Raise Gestures.



(a) Arms Covered (b) Arms Bare (c) Noise (d) Skin Pixel Count

Figure 5.10: Skin Pixel Count for Hand-Raise Gestures.

Background Probabilities

To compute the probability distribution for the hypothesis variable, the background probabilities for observing a hand-raise gesture must be known.

Based on the training data, the probabilities shown in Table 5.3 have been found. However, since contours caused by hand-raises appear in time clusters, the background probability of observing a hand-raise will depend on the number of hand-raise contour that have been observed recently. This could be modeled by a chain of NBCs as those presented, where the hypothesis variables are connected in a sequence. Then only the first NBC in the chain would use the background probabilities in Table 5.3.

$P(H = ArmsCovered)$	$P(H = ArmsBare)$	$P(H = Noise)$
0.0176	0.0274	0.9550

Table 5.3: Background Probabilities for Hand-Raise Gesture.

Instead of doing this, we have chosen a slightly simpler approach that achieves a similar result. The background probabilities are ignored. Each time a NBC classifies a contour as being caused by a hand-raise, and the skin-colour movement verification agrees, a counter is incremented and a timer is set to 0. If the value of the counter exceeds a predefined threshold, the system assumes that the person has raised his hand. For each frame that passes without a hand-raise being detected by the NBC, the timer is incremented. If the timer exceeds a predefined threshold, the counter is set to 0.

5.4 Hand-Raise Detection Conclusions

In this chapter, we have presented the method we use for hand-raise detection. The search for hand-raises for a person is restricted to a ROI to the left of the person in the image. Using the person's skin-colour LUT, a skin-colour likelihood image is produced for the ROI, which is preprocessed and then used for generating an ADP for the hand-raise ROI. In this ADP, a hand-raise will leave a vertical trace. We process the ADP to turn the trace into a single connected component, which is extracted using contour segmentation and then classified using a NBC. If the NBC classifies a contour as being caused by a hand-raise, the previous N skin-colour likelihood images are examined to determine if there has been a sufficient degree of upwards movement in the center of mass for this contour being caused by a hand-raise. If this is the case, and it happens sufficiently many times in a row, it is reported to the rest of the system that a hand-raise has occurred.

Chapter 6

System Design

In this chapter the design of the VICOWIJOY system will be described. The overall architecture will be described, followed by a more detailed description of each phase of the system. Finally, the implementation platform will be described.

6.1 Introduction

To be able to experiment on the methods described in Chapters 2–5 we have designed and implemented a system which is based on a supervisor process that controls and distributes data among a number of other processes. In a previous project of ours [5] we investigated several ways of integrating the methods in a video conferencing system with automatic speaker attention and found this model to be the most suitable. First, the overall architecture of the system is described in Section 6.2. Afterwards, the processes in the system are described in Sections 6.3 to 6.9. Finally, the implementation platform is described in Section 6.10.

6.2 VICOWIJOY Architecture

The overall architecture of the VICOWIJOY system is illustrated in Figure 6.1. The system consist of 6 different processes which are shortly described in the following list. In the following sections we will describe each of the processes in more detail.

- **Supervisor:** The supervisor is the main process in the VICOWIJOY system. It is responsible of distributing the data it receives from processes below it to other processes. Furthermore, it maintains a lookup table (LUT) which is updated based on the faces found by the face detection process. This LUT is used by the skin-colour detection process to generate a skin-colour likelihood image and by the tracker manager to give new face trackers an initial LUT to use.
- **Skin-Colour Detection:** This process takes as input a LUT from the supervisor and an image from the panorama camera. It converts input image to NRGB and uses the LUT to generate a skin-colour likelihood image. This is then returned to the supervisor.
- **Face Detection:** This process detects the faces in the images taken from the panorama camera. It preprocesses and segments the likelihood image received from the su-

pervisor into face candidates. Afterwards it uses the face verification methods described in Chapter 3 to remove the non-faces. The final list of faces is sent to the supervisor.

- **Tracker Manager:** The tracker manager maintains a set of trackers for the faces in the image using the face list produced by the face detection. It attempts to eliminate trackers for non-face objects that was started because of false positives from the face detection, as well as trackers that have lost the faces they were supposed to track, from this set.

- **Face Tracker and Hand-Raise Detector:** Each time a new face appears in the image, a face tracker and a hand-raise detector are started for this face by the tracker manager. The face tracker attempts to track the face, while the hand-raise detector, based on the face position determined by the face tracker, monitors the hand-raise ROI to detect hand-raises.

- **PTZ-Camera Control:** The purpose of this process is to control a PTZ-camera (Pan-Tilt-Zoom camera) based on the current speaker's size and position in the panorama camera. We have not implemented this part of the system, which is why it is shaded in Figure 6.1. Instead we use a simple digital zooming on the images from the panorama camera to emulate the PTZ-camera's function.

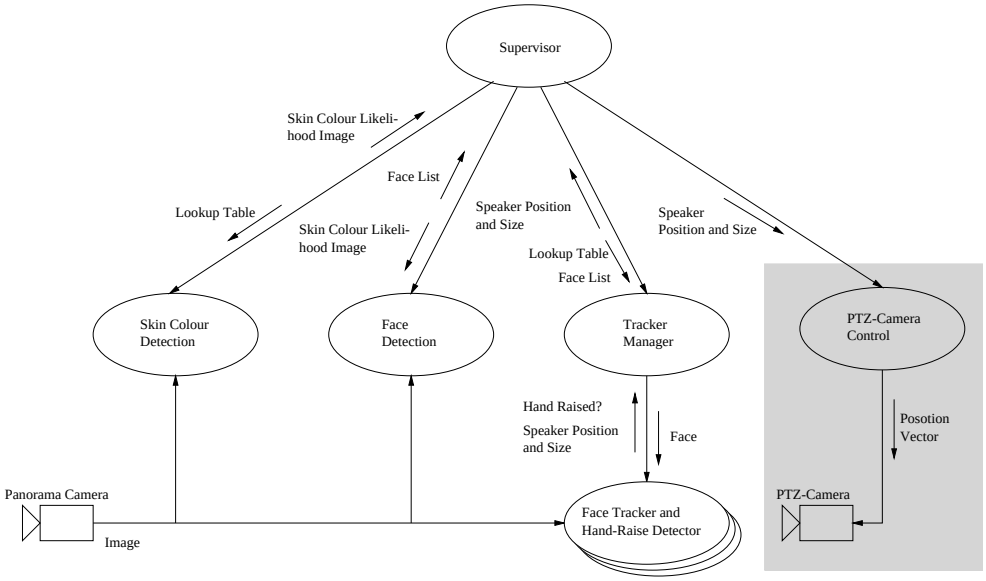


Figure 6.1: The VICOWIJOY Architecture. The system consists of 6 different processes. The PTZ-camera control process is shaded because it has not been implemented. Instead it is emulated using digital zooming on the images from the panorama camera.

6.3 Skin-Colour Detection

The skin-colour detection process is a rather simple process. It first calculates a NRGB image based on the input image which it receives from the supervisor. This is done as explained in Section 2.2 on page 8. Based on the LUT, which is also received from the supervisor, and the chromaticity image, a skin-colour likelihood image is calculated. This image is then sent back to the supervisor. The skin-colour detection process is illustrated in Figure 6.2.

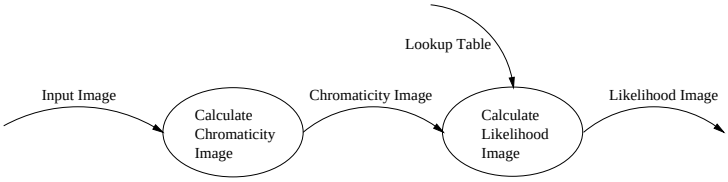


Figure 6.2: The Skin-Colour Detection Process.

6.4 Face Detection

The face detection process first erodes and dilates the likelihood image which is received from the supervisor (the whole face detection process is illustrated in Figure 6.3). The result of this is used by the threshold method to create a binary image where white pixels are skin and black pixel non skin. Based on the binary image the segmentation method explained in Section 3.2.1 on page 25 creates the initial list of face candidates and sends it to the first verification method. This is the method which verifies the rectangular size and shape of the face candidates (explained in Section 3.3 on page 27). Face candidates which do not apply to the size and shape of faces are removed from the list before the list is forwarded to the solidity verification. This method verifies whether solidity of face candidates is within lower and upper thresholds (the method is explained in Section 3.4 on page 27). Face candidates which are not within these thresholds are removed before the face candidate list is forwarded to the nose-eye template matching. Here the areas covered by the face candidates in a grey-scaled version of the input image are matched against an average nose-eye template. The face candidates which do not look like the template (defined by using a threshold as explained in Section 3.5 on page 28) are removed from the face candidate list. Finally, the face candidate list is forwarded to the ellipse matching process, which uses a gradient image to verify whether the face candidates are elliptic in shape or not (this method is explained in Section 3.6 on page 32). Face candidates which are below a threshold are removed from the list, before the final list of faces is sent to the supervisor.

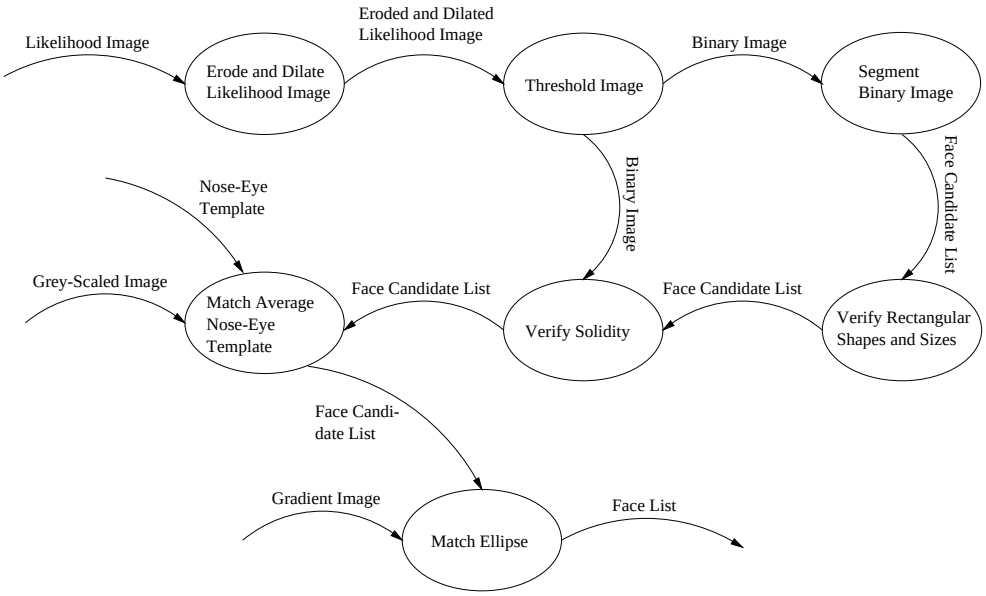


Figure 6.3: The Face Detection Process.

6.5 Tracker Manager

The tracker manager is an implementation of the algorithm described in Section 4.6 on page 48. For each new image, the supervisor provides a list of detected faces (which can be empty), as well as the LUT that was used for this image. When this happens, the tracker manager first go through the list of trackers to eliminate trackers that are unstable (i.e. moving to much) or dead (i.e. have not been updated for some time, see Section 4.6). Then it goes through the new list, that only contains "valid" trackers, and compares each tracker with each other tracker to determine if any of the trackers overlap, i.e. the size of the intersection of their estimated face bounding boxes is larger than a threshold. If this happens, the least stable tracker is deleted. Then the detected faces are paired with current trackers, and if a detected face does not overlap with a face currently being tracker, a new tracker is created for this face. The tracker manager then goes through the tracker list again to determine the neighbours of each person being tracked, and updates each tracker with the information about its neighbours, which allows the trackers to adjust the sizes of the hand-raise ROIs for the hand-raise detectors. Each tracker is then updated with the current image, which cause it to go through its update cycle, at the end of which the hand-raise detector will be invoked to determine if the person being tracked has raised his hand. If this is the case, the position of the person is given to the supervisor, that use it to zoom in on the person.

6.6 Face Tracker

The face tracker is an implementation of the face tracking algorithm described in Section 4.5 on page 42. First a NRGB image and a skin-colour likelihood image is produced as described in Section 6.3, using the tracker's own LUT. This LUT is initially provided by the supervisor (through the tracker manager). Using this likelihood image, the center of mass of the face is found using the Mean Shift algorithm. Then, the size is determined using ellipse fitting in a gradient image. This position and size, and, if available, the position and size of the detected face supplied by the tracker manager, is used to update the trackers estimate of the position and size of the face. Using the new estimate of the position and size, the skin-colour LUT is updated with pixels from the face area in the NRGB image.

6.7 Hand-Raise Detector

The hand-raise detector is responsible for classifying the contents of the hand-raise ROI. As input, it receives the skin-colour likelihood images produced by the face tracker. It performs preprocessing on these images as described in Section 5.2 on page 52. The preprocessing steps results in a (possibly empty) set of contours for the connected components in the ADP for the hand-raise ROI. The contour with the bounding box with largest area is classified using a Naive Bayesian Classifier (NBC), which is implemented as a set of functions corresponding to the attributes of the NBC. Each of these functions take the bounding box of the counter as input and returns the probabilities for the three possible states of the hypothesis variable, *hand raised with covered arms*, *hand raised with bare arms*, and *noise*. Using these probabilities the probability of the event *hand raised* is computed, and if it exceeds 50%, the hand-raise verification is initiated. If it is successful, the hand-raise counter is incremented as described in Section 5.3.3 on page 63.

If this counter exceeds a predefined threshold, the hand-raise detector informs the tracker manager that a hand-raise has occurred.

6.8 PTZ-Camera Control

The PTZ-camera control process is not implemented as a real controlling system for a camera. Instead we use digital zooming in the input image to show a close-up of the person who has got the attention by raising his hand. Based on the position and size delivered by the supervisor, the process copies the face from the input image and enlarges it to fit 320×240 pixels. This is then shown as the image of the current speaker.

6.9 The Supervisor

The main purpose of the supervisor is to distribute information among the other processes in the system. Besides that it has the task of maintaining a LUT to be used by the skin-colour detection process and the tracker manager process. Based on the skin-colour model described in Appendix C we have made 150 different Gaussian skin-colour models. These are represented as likelihoods in LUTs calculated as explained in Section 2.4.2 on page 12. The skin chromaticity distributions in the LUTs have chromaticity r centers of mass ranging from 0.2 to 0.8. Furthermore, we have doubled the standard deviations of the variances along the chromaticity r and g axes. This is because the skin-colour model is based upon that the centers of mass of the skin chromaticity distributions lie along the skin locus. This will not always be the case in the different environments of video conferences. However, using larger variances makes it possible to find faces that have skin-colours which are a within reasonable distance of the skin locus.

The 150 LUTs are divided into 6 groups of 25 LUTs. This is also illustrated in Figure 6.4, where we have misplaced the 5 groups along the chromaticity g axis to make the figure easier to read. I.e. the white group at the top lies along the real skin locus and the rest are misplaced. When the supervisor starts it gets the first LUT from the first group and uses this for skin-colour detection. Afterwards, face detection is made and if no faces are found, the next LUT from the current group will be used for the next image. When the end of the group is reached the first from the next group is used and when the end of the last group is reached the whole starts over with the first LUT from the first group. By using these rules the system covers the whole skin chromaticity area in 1 second (25 images at 25 Hz) when no faces are found. If faces are found in the image, the supervisor uses the skin chromaticities of these to update the current LUT and this LUT is then used to detect skin-colours in the next image. In this way, new people entering the scene should be found faster, when the illumination conditions are stable and other people already have been found in the scene. This is because the supervisor does not get the next LUT from one of the 6 groups as long as faces are detected.

The whole supervisor process is illustrated in Figure 6.5. As already explained it initiates with the first LUT from the first group of LUTs. It then goes into its main loop where it first gets an image from the panorama camera and sends this together with the LUT to the skin detection process. The skin detection process returns a skin-colour likelihood image, which the supervisor sends together with the input image to the face detection process. If the list of faces returned from the face detection process is empty, the supervisor gets the next LUT based on the rules explained above. If the list of faces is not empty the current

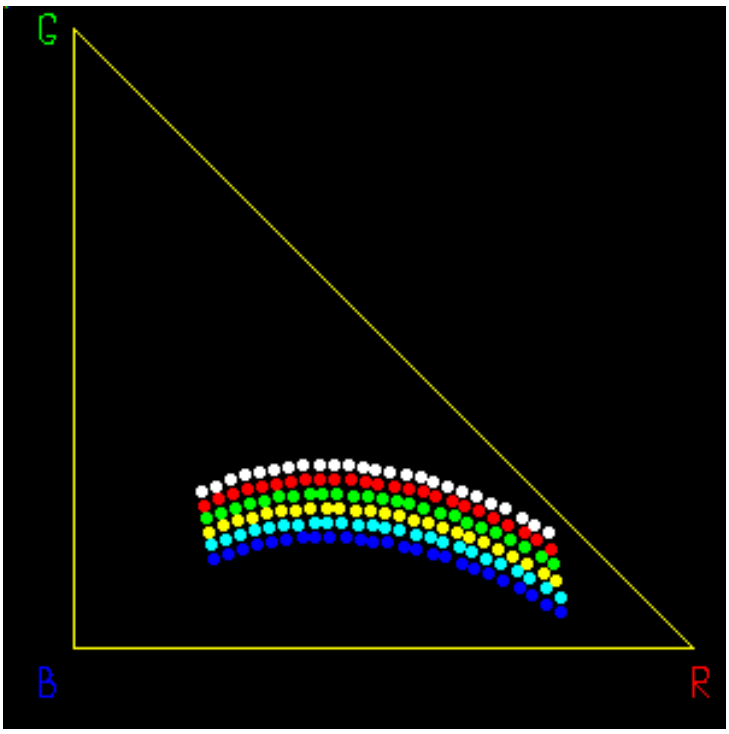


Figure 6.4: The LUTs Centers of Mass: The supervisor uses 6 groups of 25 LUTs to find skin-colours in the image.

LUT is updated based on the skin chromaticities of the faces in the list. This is done using one of the methods explained in Sections 2.5.2 on page 15 and 2.5.3 on page 17. Afterwards, the active trackers are updated and new trackers are initiated. This is done by calling the tracker manager with the input image and the detected face list. Finally, the supervisor calls the tracker manager to find out whether any hand-raises have taken place. If yes, the face of the hand raiser is zoomed in on before the supervisor loop starts over by getting the next image from the panorama camera.

6.10 Implementation Platform

We have implemented a Windows version of the system described in the previous sections. This has been done in C++ using the Microsoft Platform SDK and the Borland C++Builder compiler v5.5. Furthermore, we have used the OpenCV 3.4a and IPL v2.5 libraries from Intel to do many of the image manipulation functions. To emulate the panorama camera we have recorded a number of video sequences which are described in Appendix D. These are saved as AVI-files and read by the program at run-time.

6.11 System Design Conclusions

In this chapter we have described the design and implementation of a system called VICOWIJOY, which we will use to do experiments on the methods investigated in Chapters 2–5. The system is implemented in C++ and makes use of a number of libraries to do image manipulation. Input from real cameras are emulated using AVI-files and digital zooming. The system consists of a supervisor process which is used to distribute data

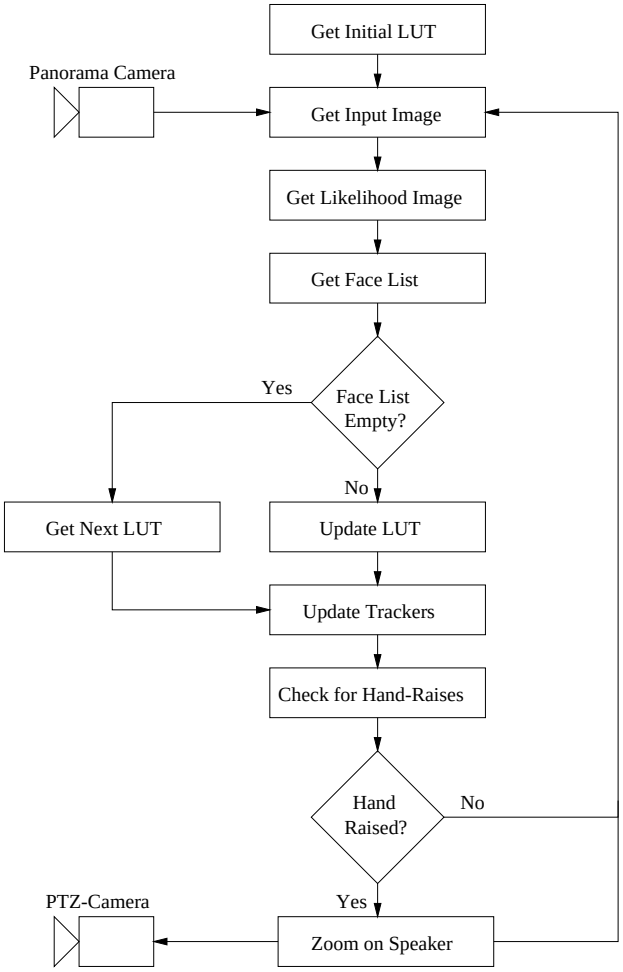


Figure 6.5: The Supervisor Process.

amongst the five other processes in the system. Furthermore, the supervisor must maintain a LUT which is used as input to the face detection process and the tracker manager. The skin-colour detection process is the initial process and is used to find the skin-colours in the images. It makes a skin-colour likelihood image and sends it to the face detection process via the Supervisor. The face detection process first segments the likelihood image into a list of face candidates and then classifies each candidate as face or non-face using rectangular shape and size, solidity, similarity to a nose-eye template, and ellipse matching. Based on the resulting list of detected faces from the face detection process, the tracker manager maintains a set of trackers for the faces in the image. If a detected face does not overlap with a face that is already being tracked, a new tracker for that face is created. Each face tracker tracks a face using skin-colour likelihood images and intensity gradient images. The likelihood images are produced by the tracker's own skin-colour LUT, and are also used for hand-raise detection for the person being tracked.

Chapter 7

Experiments

In this chapter we will describe the results of experiments made upon the methods described in Chapters 2, 3, 4, and 5. To be able to do these experiments we have recorded a number of video sequences of people doing hand-raises. These are described in Appendix D

7.1 Introduction

We have organized the experiments into four sections, each corresponding to one of the Chapters 2, 3, 4, and 5. First, experiments are made on the focus of attention methods in Section 7.2. In these we measure the performance of the methods by computing the distance between the average skin colour likelihood in the face and in the rest of the image. The best combination of methods found is used in the face verification experiments in Section 7.3. In these we first do experiments using the preprocessing methods dilation and erosion, and afterwards on each of the face verification methods alone and combined serially. Preprocessing is used to enhance the likelihood image before it is thresholded and segmented into a list of face candidates. It can therefore not be said to do actual face verification. On the other hand, the preprocessing cannot either be said to be part of the focus of attention phase. Since it in combination with the threshold and segmentation methods makes the initial list of face candidates, we have decided to experiment on it together with the face verification methods.

The performance of the face verification methods is identified by calculating the amount of false negatives (faces which are not identified as faces) and false positives (non-faces which are identified as faces).

The best combination of methods for focus of attention and face verification is used in Sections 7.4–7.5, where experiments are made with face tracking and hand-raise detection. We measure the performance of the face tracker by making a subjective evaluation of how well they follow the centers of the faces in the presence of clutter, when the faces become occluded, and when the illumination changes. The performance of the tracker manager is evaluated by examining its ability to eliminate trackers for non-face objects. Finally, experiments are made on the hand-raise detection in Section 7.5. Its performance is measured by how often it miss a hand-raise (false negative) and how often if detects a hand-raise although none has occurred (false positive). This is done for different combinations of parameters to produce different trade-offs between the two performance measures.

7.2 Focus of Attention Experiments

In this section we will do experiments on the four different methods, which can be used for focus of attention (FoA). These are:

- LUTs using the simple update method (simple LUT)
- LUTs using the ratio update method (ratio LUT)
- Gaussian models based on the ratio LUT method (ratio Gaussian)
- Gaussian models based on weighted parameters of Gaussians (weighted Gaussian)

The methods were explained in Section 2.5.2 on page 15 and Section 2.5.3 on page 17. Furthermore, experiments will be made on the use of moment constraints which were explained in Section 2.5.4 on page 17.

The methods are meant to make it possible to detect skin-colours under changing correlated colour temperatures (CCTs – we refer the reader to the beginning of Section 2.5.1 on page 14 for a description of CCTs). Therefore we have decided to do experiments using three videos, which all have been recorded under controlled, changing CCTs. These are the V8-V10 and are described in more details in Appendix D. In the videos, CCTs of 2600K, 3680K, 4700K, and 6200K are used. The experiments do therefore show both how well the methods detect skin-colours and how well they adapt to changes in the CCT. We have used a constant value $\alpha = 0.9$ (this constant is described in Section 2.5.2 on page 15) to indicate how fast the methods should adapt to changes in CCT. I.e. we trust the LUT or Gaussian model estimated so far 90% and the LUT or Gaussian model found from the current image 10%. Together their values are used to create the new LUT or Gaussian model which will be used in the next image.

7.2.1 Experiments Description

In the experiments we measure four values:

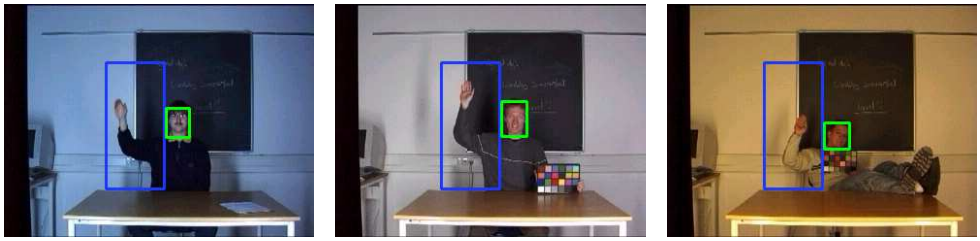
- The average skin-colour likelihood inside the face area
- The average skin-colour likelihood inside the background area
- The distance between the average skin-colour likelihood in the face and background areas
- The average computation time

The *face area* is the area in a video which contains the face. To avoid being dependent on a face tracker during these experiments, we have defined the face area at a fixed position and with a fixed size for each of the videos (see the green boxes in Figure 7.1). This can be done because the persons in the videos do not move their faces very much – i.e. they more or less stay inside the face area. Calculating the average skin-colour likelihood (from now just average likelihood) for the face area is done using Equations 7.1 and 7.2, where fw and fh are the width and height of the face area, and fx, fy its upper left corner in the image. The function $l(xx, yy)$ returns the likelihood of the pixel at position (xx, yy).

$$face_{sum} = \sum_{xx=fx}^{fx+fw} \sum_{yy=fy}^{fy+fh} l(xx, yy) \tag{7.1}$$

$$face_{avg} = \frac{face_{sum}}{fw \cdot fh} \tag{7.2}$$

The *background area* is the whole image excluding the face area and the *hand area*. The hand area is the area in the image where the hand-raises take place. It is also of a fixed size and position for each video (see the blue boxes in Figure 7.1). The hand areas have been defined to avoid, that the skin-colours of a hand influence on the average likelihood of an image area. Measuring the distance between the average likelihood of the background area and the average likelihood of the face area, tells us about the accuracy of the method used for skin-colour detection. I.e. to be able to make an easy distinction between the background area and the face area we want to have the largest possible distance between their likelihood averages.



(a) V8 (b) V9 (c) V10

Figure 7.1: The Face, Hand, and Image Areas: The images in the three videos are divided into areas of faces (green boxes) and hands (blue boxes). The background area is defined as the whole image minus the face area and the hand area.

To calculate the average likelihood of the background area, we first sum all the likelihoods in the image. Afterwards, the sum of the likelihoods of the face and hand areas is subtracted. This is done in Equation 7.4 where bw and bh are the width and height of the image. The sum of likelihoods in the hand area is calculated using Equation 7.3.

$$hand_{sum} = \sum_{xx=hx}^{hx+hw} \sum_{yy=hy}^{hy+hh} l(xx, yy) \tag{7.3}$$

$$background_{sum} = \left(\sum_{xx=0}^{bw} \sum_{yy=0}^{bh} l(xx, yy) \right) - face_{sum} - hand_{sum} \tag{7.4}$$

Thereafter, the average likelihood in the background area is found by dividing $background_{sum}$ with the number of likelihoods greater than 0 in the background area. Likelihoods above 0 are used because the result only should express the average of the potential skin-coloured pixels in the background image (i.e. the colours which are inside the upper and lower border of the skin locus illustrated in Figure 2.5 on page 16. Equation 7.5 calculates the average likelihood for the background area. M is the number of pixels in the background area with a likelihood greater than 0.

$$background_{avg} = \frac{background_{sum}}{M} \tag{7.5}$$

Finally, we also want to compare the average computation time for each of the methods. This can be used to find out whether increased accuracy is at the expense of a longer computation time.

7.2.2 Face Area

In Figure 7.2 the results of using each of the four methods on V8 are illustrated. Using the weighted Gaussian method (the blue line) gave the highest likelihood average under all kinds of CCT. Below the graphs in the figure we have indicated the CCT used. The changing CCTs happened because one of the arrangements of fluorescent lamps by incident was turned in the wrong direction. The other arrangement was at the same time positioned at 3680K, so the CCT should probably be found near 3680K. As it can be seen, all the methods degrade in performance for a period when the CCT changes from 2600K to 4700K and from 6200K to changing CCTs. This is because the methods need a small period of time to adapt to the new CCT, which in both cases is a long distance away in chromaticity plane. However, it is not more than 10-20 images and thereby less than a two seconds (the frame rate in the videos is 12.5 Hz). Going from 3680K to 2600K and from 4700K to 6200K does actually not make the methods perform worse for a short period. This must be because the distance moved in chromaticity plane is short enough to make the methods adapt fast enough from image to image. The reason for the instability at the end of V8 is, as explained above, that one of the arrangements of fluorescent lamps was rotated while the other was not. The result of this can be seen as the two abrupt breaks in the graphs in the area of changing CCTs.

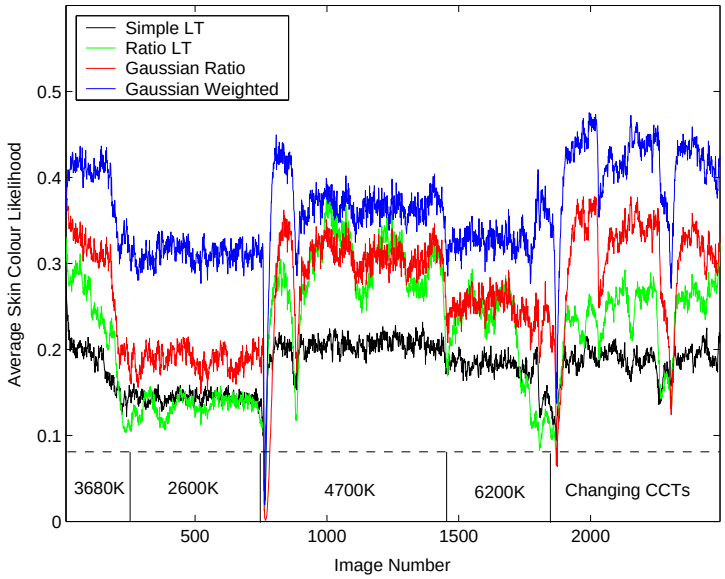


Figure 7.2: Average Likelihood in the Face Area: Using the weighted Gaussian method gives the highest average likelihood under all kinds CCTs.

The use of a CCT of 2600K and 6200K is a real problem for all the methods except for the weighted Gaussian. Investigations of the skin chromaticity distributions made by each of the methods, showed that the variances of the weighted Gaussian method always was the largest. This meant that the likelihood of the skin-colours in the face

area got higher and thereby increased the average likelihood. The Gaussian method also has the highest average likelihood in the face area when using other CCTs. In Table 7.1 we have illustrated the average likelihoods for V8, V9, and V10. As it can be seen, the use of the weighted Gaussian method gives the highest average likelihood in all the videos. Therefore, the weighted Gaussian method is the one to prefer, when looking at the likelihood average in the face area.

Video No.	Simple LUT	Ratio LUT	Ratio Gaussian	Weighted Gaussian
V8	0.155	0.164	0.220	0.336
V9	0.19	0.28	0.288	0.376
V10	0.162	0.234	0.258	0.331
Average	**0.172**	**0.229**	**0.255**	**0.348**

Table 7.1: Average Likelihood in the Face Area: The table illustrates the average likelihood in the face area in each video. The bottom row shows the average likelihood of all the videos. Using the weighted Gaussian method clearly gives the highest average likelihood.

7.2.3 Background Area

The use of the weighted Gaussian method gives the highest average likelihood in the face area. To verify that this is not at the expense of a high average likelihood in the background area, we calculated this for each of the images in V8. The results can be seen in Figure 7.3. Here the ratio LUT method performs best most of the time and the ratio Gaussian method worst most of the time. The weighted Gaussian method is close to the ratio LUT method, except when the CCT is 2600K. The better results made by the weighted Gaussian method and a CCT of 2600K is therefore at the expense of a higher average likelihood in the background area. In Table 7.2 we have illustrated the average image likelihoods in V8, V9, and V10 for each of the methods. As it can be seen, the average likelihood of the ratio LUT method is about half the size of the weighted Gaussian. In [24] they claimed that using a LUT should be more accurate than using a Gaussian model. The results in Figure 7.3 and Table 7.2 should verify this, if we define accuracy as finding only skin-coloured pixels inside the face areas. The increased accuracy nevertheless seems to be at the expense of a lower likelihood of the pixels, which actually are skin-coloured. In Table 7.3 we have illustrated the distances between the the average likelihood in the face and image areas. We want this distance to be as large as possible to be able to make a clear distinction between the face and the rest of the image. As it can be seen, the distance is by far the largest when using the weighted Gaussian method. So although this method finds higher likelihoods in the background area, the likelihoods in its face area are at the same time raised even more. Therefore, we will still say that the best method to use is the weighted Gaussian.

Video No.	Simple LUT	Ratio LUT	Ratio Gaussian	Weighted Gaussian
V8	0.052	0.040	0.075	0.063
V9	0.059	0.030	0.084	0.059
V10	0.069	0.030	0.125	0.062
Average	**0.060**	**0.033**	**0.095**	**0.061**

Table 7.2: Average Likelihood in the Background Area: The table illustrates the average likelihood in the background area in each video. The bottom row shows the average likelihood of all the videos. Using the ratio LUT method gives the lowest average likelihood in the background area.

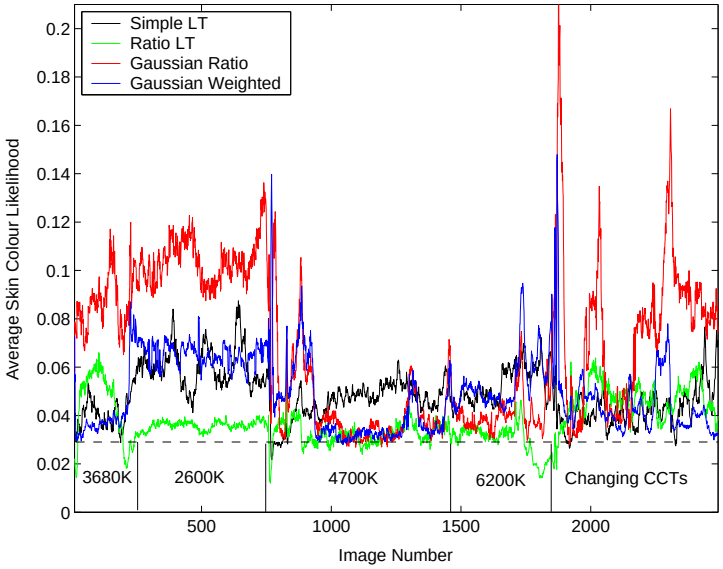


Figure 7.3: Average Likelihood in the Background Area: Using the ratio LUT gives the lowest average likelihood in the background area.

Video No.	Simple LUT	Ratio LUT	Ratio Gaussian	Weighted Gaussian
V8	0.103	0.124	0.145	0.273
V9	0.139	0.259	0.204	0.317
V10	0.093	0.204	0.133	0.269
Average	**0.112**	**0.196**	**0.161**	**0.286**

Table 7.3: Average Likelihood Distance: The table illustrates the distances between the average likelihoods in the face and background areas. When looking at this distance there is no doubt that the method to use is the weighted Gaussian.

7.2.4 Computation Time

Finally, we have calculated the average computation time for each method. The time is measured as the number of milliseconds used by a method on a single image. The results were achieved on a laptop with a 1 GHz PIII and 256 MB of memory. In Table 7.4 the average times for V8, V9, and V10 can be seen. What is interesting is whether the weighted Gaussian method's better distinction between the face and background area compared to the ratio LUT method is at the expense of a much longer computation time. As it can be seen this is not the case, since it only needs about 10% more computation time on the average. The simple LUT method is the fastest method to use but unfortunately its average face area likelihood also much lower than those of the other methods. Using the weighted Gaussian method seems to be a reasonable compromise between accuracy and speed.

7.2.5 Moment Constraints

In Section 2.5.4 on page 17 we wrote about how the performance of the methods used for FoA probably could be increased by constraining the moments of the skin chromaticity distributions. In this section we will do experiments with real image data to find out if the use of moment constraints actually can increase performance. Since the weighted Gaussian method clearly performs best without moment constraints, we have decided to only experiment on this method.

Video No.	Simple LUT	Ratio LUT	Ratio Gaussian	Weighted Gaussian
V8	16.2ms	29.2ms	46.6ms	33.1ms
V9	21.1ms	32.4ms	47.0ms	37.4ms
V10	16.8ms	33.3ms	51.7ms	33.6ms
Average	**18.0ms**	**31.6ms**	**48.4ms**	**34.7ms**

Table 7.4: Relative Average Computation Time: The fastest method to use is the simple LUT and the slowest the ratio Gaussian. Unfortunately the average face area likelihood of the simple LUT method is also much lower than those of the other methods. Using the weighted Gaussian method seems to be a reasonable compromise between accuracy and speed.

The constraints we want to make are the following:

- Minimum and maximum chromaticity g distances away from the center of masses defined by the skin-colour model described in Appendix C.
- Minimum and maximum sizes of the variance along the chromaticity r axis.
- Minimum and maximum sizes of the variance along the chromaticity g axis.
- Minimum size of the rotation angle (i.e. the covariances of rg and gr).

Having observed the position, sizes, and rotation angles of a number of Gaussian models made from V8-V10, we decided to use the moment constraints defined in Table 7.5. These values have therefore been determined empirically. The values are based on chromaticity r and g values going from 0 to 1. I.e. the maximum width, w, along the chromaticity r axis of a Gaussian model would be

$$w = 2\sqrt{0.0039} \approx 0.125$$

The minimum angle of 0° indicates that the covariances always must be 0 or negative (i.e. clockwise rotation of the skin chromaticity distribution).

Moment Constraint	Value
min g distance	-0.0118
max g distance	0.0118
min r variance	0.0010
max r variance	0.0039
min g variance	0.0010
max g variance	0.0015
min angle	0°

Table 7.5: Moment Constraints Values: The table illustrates the values we used to constrain the moments in the experiments.

In Figure 7.4 the average likelihood in the face area with and without the use of moment constraints can be seen. The results were achieved using V8 and the increase in computation time for one image was less than 1ms and thereby not noticeable. As it can be seen, the use of moment constraints does indeed increase the average likelihood under all CCTs. In Table 7.6 we have illustrated the average likelihoods achieved on V8-V10 with and without the use of moment constraints. In all the videos the average likelihood gets higher when using moment constraints. In average the use of moment constraints raises

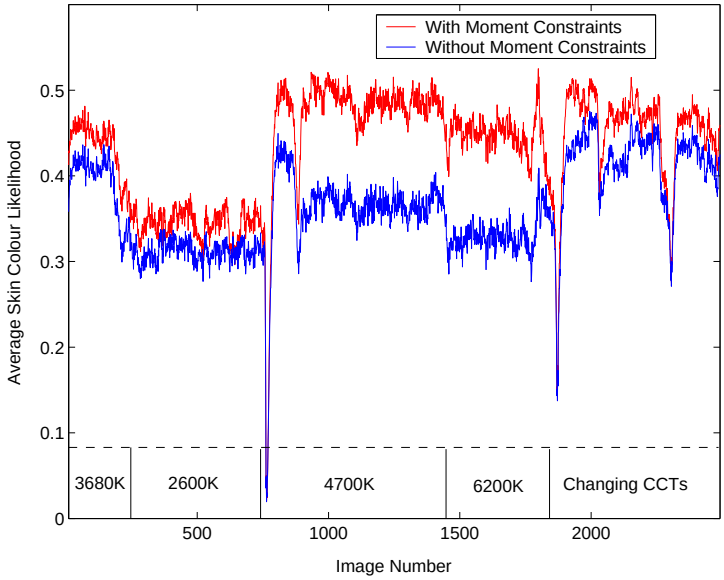


Figure 7.4: Average Likelihood in Face Area using Moment Constraints: By constraining the position, size, and angle of the Gaussian model it is possible to raise the average likelihood in the face area.

Video No.	Without Moment Constraints	With Moment Constraints
V8	0.336	0.372
V9	0.376	0.410
V10	0.331	0.429
Average	**0.348**	**0.404**

Table 7.6: Average Likelihood in the Face Area with and without Moment Constraints: Using moment constraints it is possible to increase the average likelihood in the face area.

the average likelihood in the face area by 16%. This indicates that moment constraints are worth using.

We need to ensure that the increased average likelihood in the face area is not at the expense of an even higher increase in the background area. In Figure 7.5 the average likelihood in the background area with and without the use of moment constraints are illustrated. As it can be seen, the average likelihood is increased. The increase is nevertheless much lower than the increase of the average likelihood in the face area. In Table 7.7 we have compared the distances between the average likelihoods in the face and background areas. The use of moment constraints increases the average distance for V8-V10 from 0.286 to 0.341. This is more than 19% and should verify that using moment constraints definitely is a good idea.

Video No.	Without Moment Constraints	With Moment Constraints
V8	0.273	0.299
V9	0.317	0.350
V10	0.269	0.375
Average	**0.286**	**0.341**

Table 7.7: Distance Between Face and Background Area using Moment Constraints: Using moment constraints increases the distance between the average likelihoods in the face and background area by more than 19%.

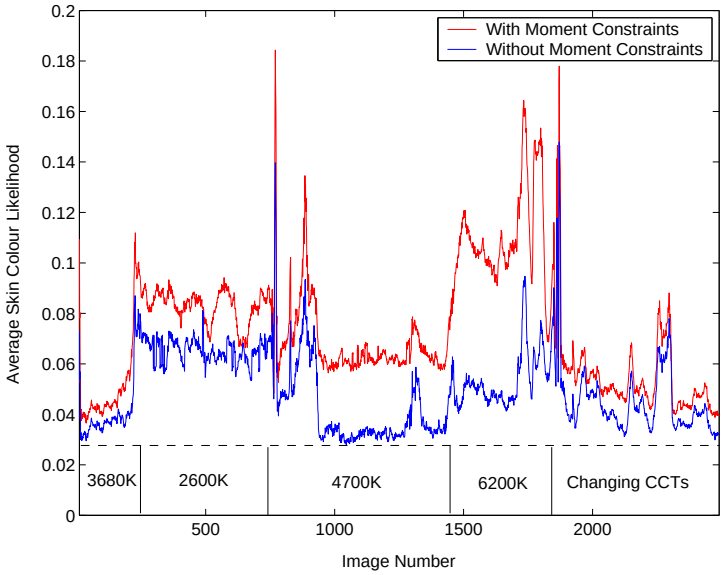


Figure 7.5: Average Likelihood in the Background Area using Moment Constraints: Constraining the moments causes a small increase in the average likelihood in the background area. This increase is nevertheless much lower than the increase of the average likelihood in the face area.

7.3 Face Verification Experiments

In this section we will describe the experiments we have made on the methods used for verification. These methods were all explained in Chapter 3 and are the following:

- Preprocessing of the skin-colour likelihood image received from the previous phase of focus of attention.
- Verification of the rectangular size and shape of the face candidates found by the segmentation.
- Verification of the solidity of the face candidates.
- Nose-eye template matching of the face candidates.
- Fitting of an ellipse around the face candidates.

First, experiments are made on each of the face verification methods to find the optimal threshold values to use. Furthermore, we will identify how many false negatives and false positives that are made using these thresholds. Afterwards, we will do experiments where we combine the methods to see if they together can make better results than alone.

To compare the computation time of the methods we have measured the average computation time of each. This has been done on the same computer as was used for the focus of attention experiments in Section 7.2.

7.3.1 Experiments Description

To experiment on the preprocessing methods we use the same videos (V8-V10) as we used for the focus of attention experiments. By using the same videos we want to find out, whether the use of preprocessing (i.e. erosion and dilation) can increase the distance

between the average skin-colour likelihoods in the face and image areas. Afterwards, we will use the result of the experiment on the preprocessing methods to choose a threshold value. This value will be used when the preprocessed likelihood image is converted into a binary image.

In the rest of the face verification experiments, we use the videos V3, V8, and V14 (all described in Appendix D). V3 has a fixed CCT of 3680K and people walking around in the background, V8 has changing CCTs of 2600K, 3680K, 4700K, and 6200K, and V14 has a fixed CCT of 3680K and little movement. Using these videos we hope that the experiments will be able to tell how the methods perform in different environments.

In each of the videos we have defined the face area as rectangle of fixed position and size. We have made these areas large enough to endure small movements of the faces (see Figure 7.6). To be able to measure the efficiency of the methods, we count the number of *false positives* and *false negatives*. False positives are face candidates which are classified as faces but really are something else. False negatives are face candidates which are classified as non faces but really are faces. Face candidates covering 50% or more (this value has been chosen based on empirical investigations) of a face area should be classified as faces. A good result is achieved when the number of false positives and negatives are close to 0.

(a) V3 (b) V8 (c) V14

Figure 7.6: The Face Areas: For each of the videos V3, V8, and V14 we have defined the area where the face should be found. These are indicated by the green rectangles in Figures (a)-(c).

To measure the percentage of false negatives and false positives, we use the outcome of the segmentation method as reference. This means that we have chosen to rely on the segmentation and not ground truth. For each segmentation of an image, we count the number of face candidates which cover the face areas. Adding the numbers together for all the images in a video, gives a sum which is equal to 100% true positives (i.e. faces that are actually verified as faces). To find the percentage of false negatives for a method, we use Equation 7.6. In this STP is the sum of true positives found by the segmentation method, and MTP the sum of true positives found by a face verification method.

$$FN = 1 - \frac{MTP}{STP} \tag{7.6}$$

The number of false positives found by the segmentation method is equal to 100% false positives. To calculate the number of false positives made by a face verification method, we use Equation 7.7. In this SFP is the number of false positives found by the segmentation method, and MFP the number of false positives found by a face verification method.

$$FP = \frac{MFP}{SFP} \tag{7.7}$$

7.3.2 Preprocessing and Segmentation

We want to find out whether the use of erosion and dilation can increase the distance between the average likelihood in the face and background area. The best result made in Section 7.2 is used as reference. This was the use of the weighted Gaussian method with moment constraints. In Figure 7.7 the results achieved when adding preprocessing (the 8-connected neighbourhood of each pixel was used for erosion and dilation) to the likelihood images of the weighted Gaussian method can be seen. The figure illustrates the average likelihood in the face area in V8. We have tried to use 3 different combinations of erosion and dilation. It should be clear that using these methods increases the average likelihood in the face area. At the same time, they do more or less not raise the average likelihood in the background area. In Table 7.8 the distances between the average likelihood in the face and background area can be seen. Using erosion and dilation clearly increases this distance for both V8, V9, and V10. The computation time used on the preprocessing methods was in average about 5ms per image. The average computation time for the weighted Gaussian method was measured to 34.7ms in Table 7.4. We therefore think that the increase in distance between the average likelihoods in the face and background areas outweighs the increase in computation time.

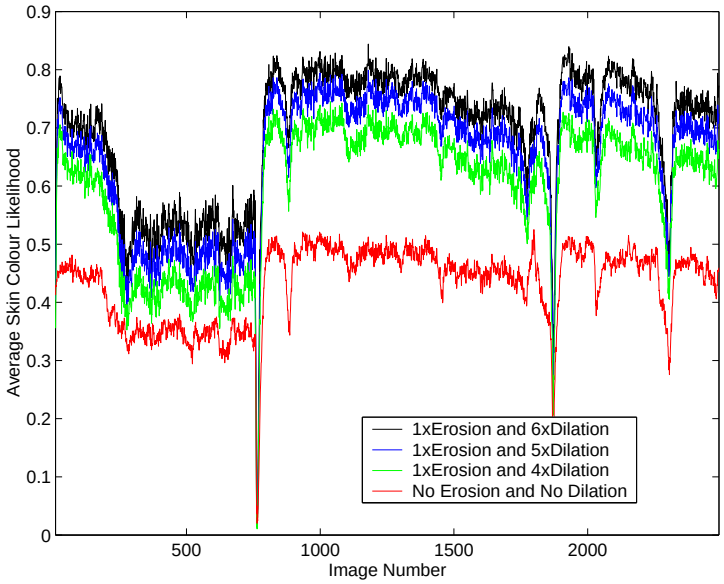


Figure 7.7: Average Likelihood when using Preprocessing: Preprocessing the likelihood image by using erosion and dilation increases the average likelihood in the face area.

We decide to use the combination of $1\times$ erosion and $6\times$ dilation for the rest of the face verification experiments. To find the threshold value to use when converting the likelihood image to a binary image, we look at Figure 7.8. In this we have plotted the average likelihoods of the face and background areas for each image in V8. Looking at this, a reasonable threshold value would be around 0.40 (indicated by the dotted line). Here most of the average face area likelihoods are above the line (except for a couple of places, where the change in CCT was to large for the system to follow). All the average background area likelihoods are below the threshold and it should therefore be possible to make a

Video No.	0×E and 0×D	1×E and 4×D	1×E and 5×D	1×E and 6×D
V8	0.299	0.408	0.456	0.497
V9	0.350	0.535	0.586	0.631
V10	0.375	0.548	0.606	0.653
Average	**0.341**	**0.497**	**0.549**	**0.594**

Table 7.8: Average Likelihood Distance when using Preprocessing: The table illustrates the distances between the average likelihood in the face and background areas. By using combinations of erosion and dilation this distance can be increased significantly. The letters **E** and **D** are short for Erosion and Dilation.

good distinction between the face and the rest of the image. We therefore decide to use a threshold value of 0.40 for the rest of the face verification experiments.

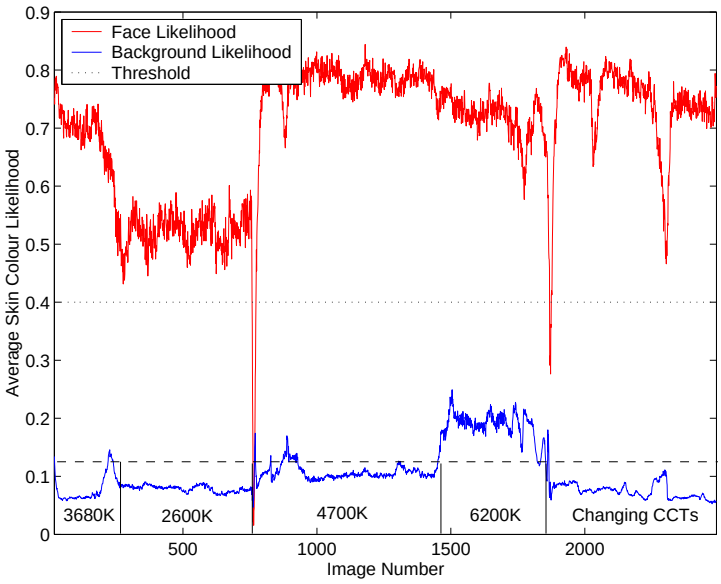


Figure 7.8: Choosing a Threshold Value: The figure illustrates the average likelihoods in the face and background areas. Using a threshold value just below most of the average likelihoods in the face area, should make it possible to make a good distinction between the face and the rest of the image.

7.3.3 Rectangular Size and Shape

We have decided to experiment on this method using the values in Table 7.9 to constrain the rectangular size and shape of the face candidates. These values should be suitable for the resolution of the images we use in this project (320×240 pixels). If images of different resolution are used the minimum and maximum width and height would have to be adjusted. MinRelation and MaxRelation indicates the minimum and maximum relation between the widths and heights of the rectangles.

The result of using the rectangular size and shape method on V3, V8, and V14 can been seen in Table 7.10. It should be noted that though the method is very simple (and fast – the average computation time could not be measured with the method we used, i.e. it was less than 1ms), it is capable of removing almost 70% of the false positives in average. At the same time almost no false negatives are made, so using the method as an initial rough filter, before using more advanced and computational demanding methods, should be wise. The number of false positives seems to increase with the amount of motion in

Parameter Name	Parameter Value
MinWidth	20 pixels
MaxWidth	100 pixels
MinHeight	20 pixels
MaxHeight	100 pixels
MinRelation	0.6
MaxRelation	2.0

Table 7.9: Rectangle Method Parameters: The table illustrates the values we have chosen to use for the parameters in the rectangle method experiments.

the images. I.e. in V3 we have two people doing hand-raises and a third walking around in the background, and in V14 two persons are doing very few hand-raises and are nearly not moving. The first gives reason to more than 50% of false positives and the second only around 12%.

Video No.	False Negatives	False Positives
V3	0.1%	52.1%
V8	1.2%	31.0%
V14	0.4%	11.9%
Average	**0.6%**	**31.7%**

Table 7.10: Rectangle Method Results: Although the rectangle method is simple and fast it removes almost 70% of the false positives. At the same time, the amount of false negatives is less than 1%. Using the method as an initial rough filter should therefore be wise.

7.3.4 Solidity

To find out what the solidity of faces are, we have calculated it for the fixed face areas in V3, V8, and V14 using the likelihoods in the preprocessed likelihood image. The result of this is illustrated in Figure 7.9 (only 525 images from each video have been used to make the figure easier to read). In the same figure we have illustrated the average solidity of all other face candidates, found by the segmentation method (i.e. the false positives). Using the two threshold values illustrated in the figure should make it possible to remove most of the non faces.

Based on the values in Figure 7.9, we choose to use a lower threshold of 0.35 and an upper threshold of 0.9 for the next solidity method experiment. I.e. only face candidates with a solidity between 0.35 and 0.9 are verified as faces. In Table 7.11 the amount of false positives and false negatives for the solidity method is illustrated. The average number of false positives is higher than when using the rectangle filter. Still, the solidity method is capable of removing almost half of the false positives and at the same time the amount of false negatives is kept below 3%. Its average computation time was 1.5ms, so it is also slightly more computational demanding than the rectangle method.

7.3.5 Nose-Eye Template Matching

To experiment on the efficiency of nose-eye template matching, we will first investigate how similar the faces in the face areas are to the average nose-eye template. By doing this we will find the threshold, which should be used to determine whether a face candidate is similar to the template or not. The template we use is made as an average of 15 cut-

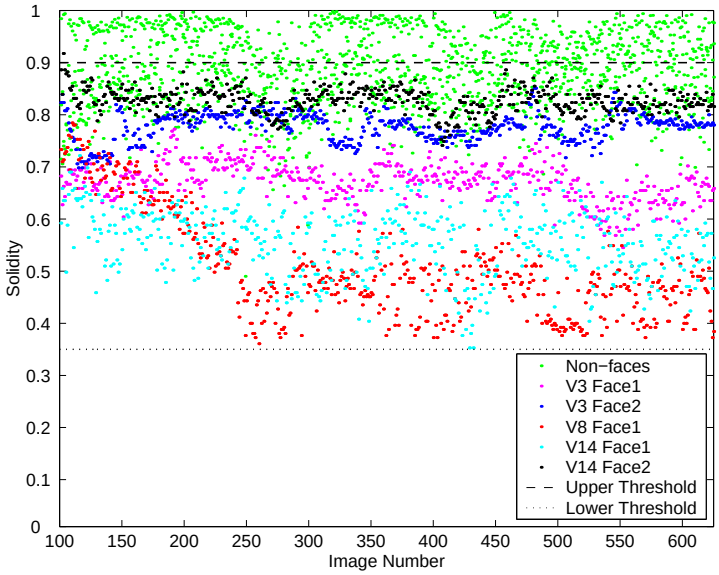


Figure 7.9: Finding Solidity Thresholds: Looking at the figure it can be seen that the solidity of most faces is between 0.35 and 0.9. The solidity of the false positives tend to be above 0.9 and these objects can therefore be removed using a threshold.

Video No.	False Negatives	False Positives
V3	2.2%	64.7%
V8	0.8%	50.3%
V14	4.6%	31.7%
Average	**2.5%**	**48.9%**

Table 7.11: Solidity Method Results: Using the solidity method more than half of the false positives can be removed. At the same time the amount of false negatives is kept at a minimum of 2.5%.

outs of the nose-eye area of upright faces randomly chosen from the videos described in Appendix D. The number of layers in the image pyramid is fixed at 6, going in intervals of 10% from 100% to 50% of the size of the width and height of the input image. To match a face candidate with the template, we have used the zero mean normalized cross correlation method (described in Section 3.5.2 on page 31).

In Figure 7.10 the similarities between the face areas and the template can be seen (a similarity of 1 is a perfect match). In the same figure we have illustrated the similarities for the false positives in V3, V8, and V14. From this we conclude, that most of the faces areas have a higher similarity with the template than the false positives. Looking at the figure it seems reasonable to place the threshold around 0.67 (indicated by the dashed line).

In Table 7.12 the number of false negatives and false positives when using template matching and a threshold of 0.67 are shown. It can be noted, that the number of false negatives is higher than when using the rectangle or solidity method. This is most likely because we use a template for upright faces, and therefore not are capable of recognizing the faces when they are rotated. Finally, we want to stress that the template matching method is rather computational demanding. Its average computation time was 122ms which is more than 60 times slower than the rectangle and solidity methods together. Using the rectangle and solidity methods in advance could probably remove a lot of uninteresting objects, and thereby decrease computation time of the template matching method. We will investigate this in Section 7.3.7.

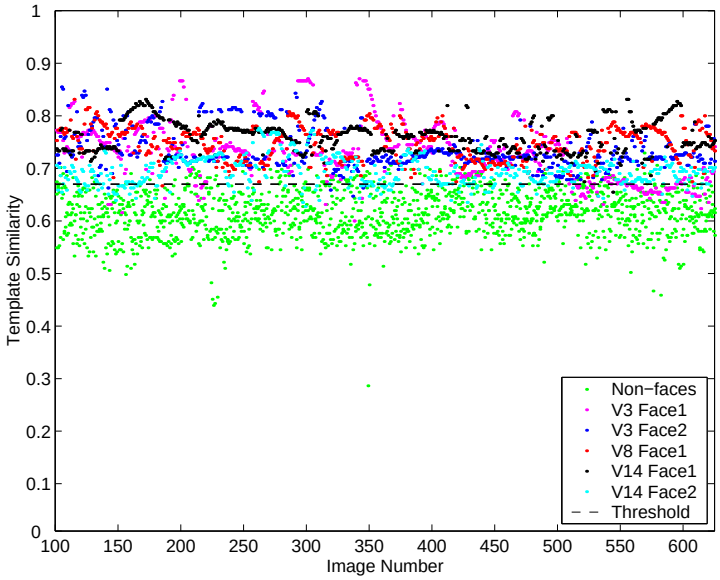


Figure 7.10: Finding the Template Matching Threshold: Looking at the figure it can be seen that the face areas in most cases have a similarity with the template of more than 0.67. At the same time the false positives do mostly have a lower similarity. Therefore, we will use a threshold of 0.67.

Video No.	False Negatives	False Positives
V3	13.0%	62.8%
V8	2.5%	33.5%
V14	5.2%	20.0%
Average	**6.9%**	**38.8%**

Table 7.12: Template Matching Method Results: Using the template method with a threshold of 0.67, more than 60% of the false positives generated by the segmentation method are removed. The number of false negatives is higher than when using the rectangle or solidity method. This is probably because of faces being rotated.

7.3.6 Ellipse Fitting

In Section 3.6 on page 32 we described two ways of measuring the fit of an ellipse by calculating the average gradient in its perimeter. These were best fit (BF) and first fit (FF). To find out what the highest, average gradients of ellipses placed around faces are, we have used the BF method. We made a gradient image from the intensity image of the input image, and calculated the best fitting ellipses around the face areas in V3, V8, V14. The result of this is illustrated in Figure 7.11, where we also have plotted the average gradients of the false positives in the videos. The face areas mostly have an average gradient of more than 0.4, and the false positives are mostly below this value. Therefore we choose to use a threshold of 0.4 in the following experiment to determine whether a face candidate should be verified as a face or not.

In Table 7.13 the numbers of false positives and false negatives when using the BF and FF methods can be seen. They are close to the same, so which method to use should not matter that much. The average computation time of the BF method was 32ms per image. Using the FF method the computation time was only 28ms per image. Based on this, we choose to use the BF method for the rest of the face verification experiments.

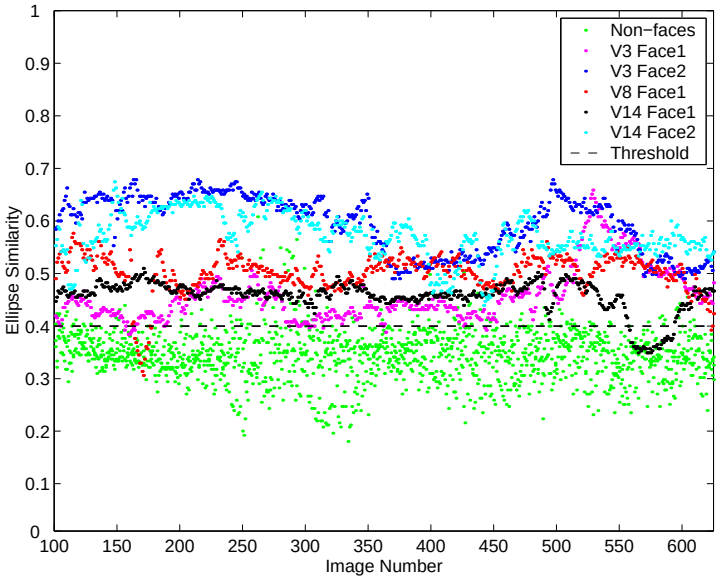


Figure 7.11: Finding the Ellipse Fitting Threshold: Looking at the figure it can be seen that the face areas in most cases have a best fitting ellipse with an average perimeter gradient of more than 0.4. Other objects (false positives) are mostly below this value.

Video No.	BF FN	FF FN	BF FP	FF FP
V3	3.4%	3.0%	38.6%	40.0%
V8	13.6%	13.4%	23.0%	24.1%
V14	5.5%	5.4%	21.2%	22.2%
Average	**7.5%**	**7.3%**	**27.6%**	**28.8%**

Table 7.13: Ellipse Fitting Method Results: Using the best fit (BF) or the first fit (FF) method gives more or less the same results. Since the FF is computational fastest, we choose to use that. FN means false negatives and FP false positives.

7.3.7 Combining the Methods

In this final section of face verification experiments, we will investigate whether the methods in combination can achieve better results than individually. In Table 7.14 we have compared the result made by each of the methods individually and the result when combining them all serially (the serial combination is explained in Section 6.4). It should be clear that by combining the methods, we get a significant better result. The number of false negatives is only just above 10%, and the average computation time is also lower than when using only template matching. The computation time gets shorter, because the first simple methods remove face candidates, that are very dissimilar to faces. In this way, the template matching method is made on fewer face candidates.

Method(s)	False Negatives	False Positives	Computation Time
Rectangle	0.6%	31.7%	$<$1ms
Solidity	2.5%	48.9%	1-2ms
Template Matching	6.9%	38.8%	122ms
Ellipse Fitting	7.3%	28.8%	28ms
Serial Combination of All	15.8%	10.7%	105ms

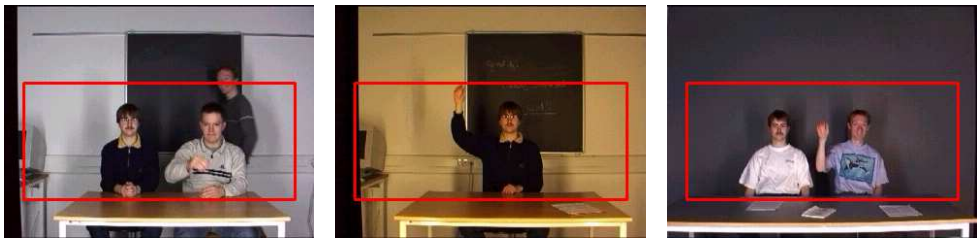
Table 7.14: Combining Methods: By combining the face verification methods it is possible to shorten the computation time compared to when using only template matching. Furthermore, the number of false negatives gets down to 10.7%.

The number of false negatives rises when combining the methods. However, as also explained in Section 3.1 on page 23, it is much more important to have few false positives than having few false negatives. This is because the face candidates that are verified as faces are used to start new trackers, which changes the areas used to search for hand-raises (we refer the reader to Chapter 5 for more about hand-raise detection). A non face object that is verified as a face, could therefore be cause of hand-raises being missed by the system. To avoid this, we need to make sure that the face verification methods are strict enough to only verify objects that actually looks like faces as faces. The 10.7% of false positives achieved by combining the face verification methods is still too large – i.e. in average it would mean, that approximately 1.25 non face objects would be verified as faces every second (with a frame rate of 12.5 Hz).

To lower the number of false positives, we have tried to use more strict threshold and parameter values for the face verification methods. These new values can be seen in Table 7.15, where we also have illustrated the old values that were used to create the results in Table 7.14. Furthermore, we have constrained the size of the area in the images in which we search for face candidates. We think that this is a reasonable constraint to make, since the faces of people sitting down normally will be situated around the center of the image. Defining this area before the start of every video conference should not be a large overhead to the use of the system. In Figure 7.12 we have illustrated the areas in V3, V8, and V14 in which we look for face candidates.

Parameter/Threshold	**Old Value**	**New Value**
Rectangle MinWidth	20 pixels	20 pixels
Rectangle MaxWidth	100 pixels	50 pixels
Rectangle MinHeight	20 pixels	20 pixels
Rectangle MaxHeight	100 pixels	60 pixels
Rectangle MinRelation	0.6	1.0
Rectangle MaxRelation	2.0	1.5
Solidity Lower Threshold	0.35	0.45
Solidity Upper Threshold	0.90	0.90
Template Matching Threshold	0.67	0.70
Ellipse Fitting Threshold	0.4	0.5

Table 7.15: New Threshold and Parameter Values: To make the face verification more strict we have chosen to experiment with a new set of parameters and thresholds values.



(a) V3 (b) V8 (c) V14

Figure 7.12: The Search Areas: For each of the videos V3, V8, and V14 we have defined the area in which we want to look for face candidates. These are indicated by the red boxes in (a)-(c).

The result of using these new values on the images in V3, V8, and V14 can be seen compared to the result of using the old values in Table 7.16 (the results are averages of

the three videos). We now have 0.0%[1] false positives, which must be said to be ideal. At the same time, the number of false negatives has only risen to 53.6%, which means that we in average should be able to find a face within 2-3 images. Since we have a video stream with 12.5 images per second, the time used to find a face should in average be almost not noticeable. Finally, the computation time is only 67ms per image when using the new values. This is about 36% faster than the computation achieved when using the old values. The reason for this is that the rectangle and solidity methods removes more face candidates when using the new values. Therefore, the template matching method, which is the most computational demanding method, is applied to much fewer face candidates.

Threshold/Parameter Values	False Negatives	False Positives	Avg. Comp. Time
Old Values	15.8%	10.7%	105ms
New Values	53.6%	0.0%	67ms

Table 7.16: Using Stricter Threshold and Parameter Values: Using stricter constraints in the face verification methods, makes it possible to have 0.0% false positives. At the same time only about half of the valid faces are verified as non faces. It should therefore be possible to find a face within 2-3 images.

7.4 Face Tracking Experiments

In this section, experiments done to evaluate the performance of the tracking algorithm and the tracker manager are described. First, suitable parameters for the system are chosen in Sections 7.4.1–7.4.3. In Section 7.4.4 it is determined how accurately the face tracker is able to track the face in the presence of clutter and occlusions. Then, in Section 7.4.5, the ability of the tracker manager to eliminate trackers started for non-face targets is examined.

7.4.1 Tracker Manager Parameters

As described in Section 4.6 on page 48, the tracker manager depends on five parameters:

- u_{min} – the minimum number of times a tracker must have fitted an ellipse to the object tracked.
- ξ_{max} – the maximal allowable value of the unstability measure ξ for a tracker. If $\xi > \xi_{max}$, the tracker is deleted.
- θ_{del} – the maximal allowable overlap between the estimates of two trackers. The overlap is measured as the ratio between intersection area and union area of the bounding boxes of the estimated face ellipses.
- θ_{upd} – the minimum overlap between between a detected face and a trackers estimate for the detected face to be used to update the tracker's estimate. The overlap is measured as for θ_{del}.

[1] Actually the number was 0.029%, since we had 2 false positives altogether in the three videos, which consist of 7003 faces. However, these two false positives were due to one person, which moved his head out of the fixed positioned face area two times. So what was identified as a false positives, was actually a face which was outside the face area. We will therefore claim, that it is more correct to say, that we did not have any false positives at all.

- θ_{time} – the maximum number of time steps (frames) that is allowed to pass without the tracker's estimate being updated (with either the result of Mean Shift and ellipse fitting, or a detected face).

As discussed in Chapter 4, the purpose of u_{min} is to ensure that new trackers – that may have been created due to false positives from the face detection – do not affect the width of the hand-raise ROI belonging to the tracker to the right. Only when the system has been tracking an object for a while, the object will be accepted as being a face, as it then will be more likely that it actually is a face (because it has been verified several times that the shape is elliptic). Moreover, it is also used to allow the unstability measure ξ for a tracker to be large in the beginning. We have set $u_{min} = 12$ in the following experiments. This value ensures that at least a second will pass before the object is accepted as a face, and it seems to provide sufficient time for the unstability measure to drop to a "safe" level (i.e. less than ξ_{max}).

In Section 7.4.5, a reasonable value for ξ_{max} is determined for the purpose of eliminating trackers caused by false positives from the face detection. In Sections 7.4.3–7.4.4 a "large" ξ_{max} has been used to avoid that trackers tracking faces were deleted.

Since the faces of the persons participating in a video conference are unlikely to overlap, the threshold θ_{del} has been set to 0.05. Overlap of the tracker estimates is very rare in the videos that we use, and is always caused by false positives from the face detection.

$\theta_{upd} = 0.5$ appears to be a reasonable degree of overlap to require. It has been verified that this will usually result in the detected face to be used for updating the tracker's estimate. This value should not be set too low, as it then can happen that a false positive from the face detection is used to update the tracker.

The time that is allowed to pass for a tracker without Mean Shift and ellipse fitting producing any measurements for the Kalman filter, or the face detection supplying any measurements, has been set to $\theta_{time} = 15$. This corresponds to 1.2 seconds. Our preliminary experiments have shown that this is enough in most cases. As this value is used to delete a tracker if it was started due to a false positive from the face detection or if it has lost the face it was tracking, it should not be set higher than necessary to handle occlusions and temporary maladaptions of the skin-colour detection.

7.4.2 Mean Shift and Ellipse Fitting Parameters

As described in Section 4.5 on page 42, Mean Shift depends on three parameters:

- β which controls the size of the search window.
- The maximum number of iterations allowed for a single image.
- The distance that the search window must move less than for the iterations to end for the current image (unless the maximum number of iterations has been reached).

We have found that a value for $\beta = 16$ results in a search window that usually is large enough to contain the entire head and perform ellipse fitting on the outline of the head. If the search window size is increased beyond this size, the risk will be increased that "noise" such as hand movement in the hand-raise ROI will disturb Mean Shift. Therefore, we set $\beta = 16$.

Usually, Mean Shift only requires one or two iterations to find the center of mass of the face, see Figure 7.13. Hence, we restrict the number of iterations for Mean Shift to 3. If the search window has only moved 1 pixel after an iteration of Mean Shift, it is likely that it will not move if another iteration is done. This is because the search window is large enough to contain the entire face, and if the face is at the border of the search window, it will move more than 1 pixel. Therefore, we will set the minimum distance that the search window must move for another iteration to be done to 1.

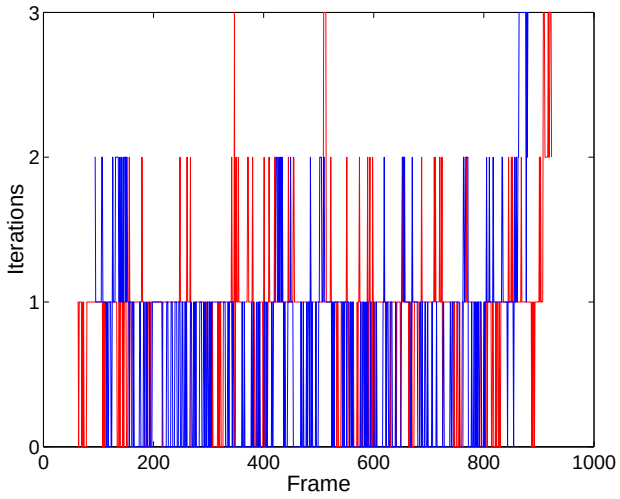


Figure 7.13: Mean Shift Iterations Required in V11: These plots show how many iterations the Mean Shift algorithm performed for each image for each person while tracking the faces of two persons.

Since the face that is being tracked already has been verified as a face, we will use a lower threshold for the ellipse fitting, such that a poorer fit is accepted. This is necessary, as the rather strict threshold used for face detection often is too high. We have found that with a threshold of 0.2, the ellipse fitting will fit an ellipse to the face in most images. However, instead of using the first fit approach as described in Section 3.6, we will have to use the best fit, as the low threshold otherwise can result in the ellipse sometimes being fitted to the background instead of the face.

7.4.3 Kalman Filter Parameters

As described in Section 4.5 on page 42, we use a zeroth order Kalman filter for estimating the position and size of the face, that is, we predict that the parameters does not change from one image to the next. In all experiments, the process noise of the Kalman filter has been fixed to:

$$Q = \begin{bmatrix} 4z^2 & 0 & 0 \\ 0 & 4z^2 & 0 \\ 0 & 0 & 0.25z^2 \end{bmatrix} \tag{7.8}$$

where z is the scaling constant depending on the distance to the persons (described in Section 4.5 on page 42). For the videos we use, this constant is $z = 1.0$ or $z = 0.8$. The three non-zero entries in Q correspond to the variances of x position, y position, and width of the face ellipse, respectively. We found in a previous project [5] using similar image resolution and slightly higher frame rate that the face on the average moved approximately one pixel per image in a video conferencing setup. We wish to allow for a

little more movement, hence the variance of $4z^2$ for the position entries. The size of the face (that is, its projection onto the image plane) is not likely to change very much during a video conference, but in case the size found by face detection is not accurate, it should be possible for the estimate of the width of the face to adapt. Therefore, the variance for the width has been set to $0.25z^2$.

As described in Section 4.5 on page 42, the measurement noise

$$R = \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_3 \end{bmatrix} \tag{7.9}$$

for the Kalman filter is computed as follows:

$$r_i = s_i(k_H H + F^2 + k_i z) \tag{7.10}$$

$$H = (\widehat{ln}(\frac{c_1}{N_1}\sum_{\tau=0}^{N_1}\frac{M_{t-\tau}^{ADP}}{A}))^2 + (\widehat{ln}(\frac{c_1}{N_1}\sum_{\tau=0}^{N_1}\frac{M_{t-\tau}^{ADP,N}}{A}))^2 \tag{7.11}$$

$$\widehat{ln}(x) = \begin{cases} ln(x), & x > 1 \\ 0, & \text{otherwise} \end{cases} \tag{7.12}$$

$$F = \frac{c_2}{N_2}\frac{\sum_{\tau=0}^{N_2}\omega_{M,t-\tau}\Delta M_{t-\tau}}{z^2} + \frac{c_3}{N_3}\frac{\sum_{\tau=0}^{N_3}\omega_{\sigma,t-\tau}\Delta\sigma_{t-\tau}^2}{z} \tag{7.13}$$

where M^{ADP} is the zeroth moment of the hand-raise ROI belonging to the face being tracked and $M^{ADP,N}$ is the zeroth moment of the right neighbour hand-raise ROI (if there is a neighbour to the right in the image). ΔM is the difference between the zeroth moment of the Mean Shift search window and a weighted average for this value that has been computed such that recent values are weighted most. $\Delta\sigma^2$ is the difference between the variance for the search window and a similar average value. These averages depend on a parameter α that controls the speed with which these averages adapt to the current values.

Thus, there are 8 constants for which reasonable values must be determined: s_i, k_i, k_H, k_F, N_1, N_2, N_3, and α. The values must ensure that the measurement noise variances are large when something happens that can cause a bad measurement by Mean Shift and ellipse fitting, but at the same time, the measurement noise should be low when the measurement done using Mean Shift and ellipse fitting can be expected to be reliable. Moreover, the values must ensure that the measurement noise variances are within reasonable ranges compared to the process noise.

The events that could cause bad measurements include:

- Occlusion of the face, e.g. by a person in the foreground.
- Hand-raise gestures by the person being tracked or by his right neighbour in the image.
- Skin-colour movement in the background, e.g. by a person moving in the background.

- Changing illumination, causing changes in the skin-colour likelihoods of the face and the background.

The values presented in Table 7.17 have been determined empirically by adjusting them to ensure reasonable performance in all of the above cases. That the values are reasonable has been verified by plotting the measurement noise standard deviation for videos including the four types of events.

s_1 and s_2	s_3	c_1	c_2 and c_3	k_i	k_H	N_1	N_2	N_3	α
1	2	32	1	1	25	3	3	3	0.05

Table 7.17: Measurement Noise Parameters.

The plots of measurement noise standard deviation ($\sqrt{r_1}$) in Figure 7.14 were produced for video V4, where a person walks back and forth in front of two other persons. The interesting events in this video are shown in Figure 7.15.

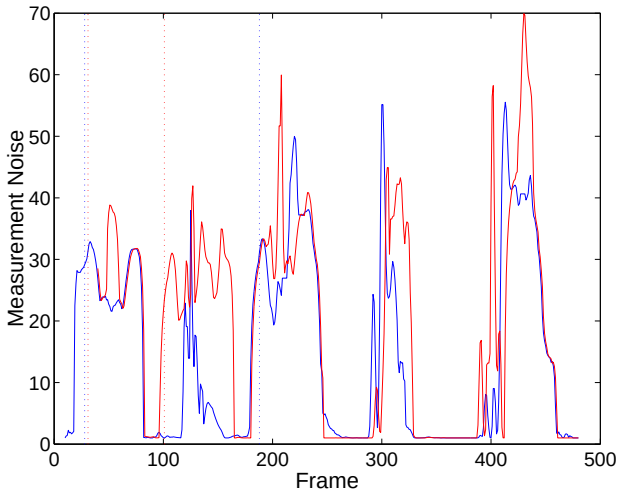


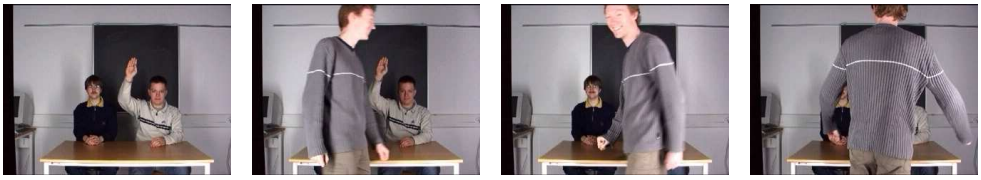
Figure 7.14: Measurement Noise for Kalman Filter: The plot shows the measurement noise standard deviation ($\sqrt{r_1}$) for each person for each frame in V4. The graph for the person to the left is plotted with red, while the graph for the person to the right is plotted with blue (see Figure 7.15). The vertical lines indicate when hand-raises were performed (the blue line corresponds to the person with the blue measurement noise graph).

In Figure 7.16, the two terms F^2 and

$$k_H\,(\widehat{ln}(\frac{c_1}{N_1}\sum_{\tau=0}^{N_1}\frac{M_{t-\tau}^{ADP}}{A}))^2$$

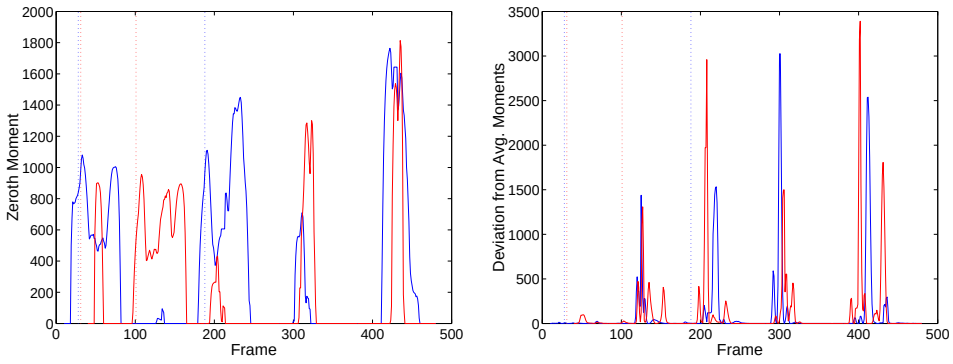
are plotted to illustrate what contribution each of them makes to the measurement noise. Some of the peaks in Figure 7.16(a) do not coincide with hand-raises (the vertical lines) – these peaks are caused by the person walking by in the foreground. Note that for the person to the left (red graphs), both the red and blue graphs in Figure 7.16(a) contribute to the measurement noise (see Equation 7.11). The peaks in Figure 7.16(b) all coincide with the occlusions due to the person in the foreground. As it can be seen from the ADP in Figure 7.17, temporary occlusion may cause the system to adapt to the colour of the occluding object. This results in noise in the ADP if the adaption cause an increase in the skin-colour likelihoods for the background pixels. As this increases the zeroth moment of the hand-raise ROI, it will contribute to the measurement noise. This will only happen

(a) Image 5 (b) Image 27 (c) Image 101 (d) Image 129



(e) Image 186 (f) Image 204 (g) Image 300 (h) Image 410

Figure 7.15: Events in V4: First, both persons raise their hands and take them down. Then the person to the left raise his hand. While the hand is raised, another person walks by in the foreground, temporarily occluding each of the persons. Then the person to the left takes down his hand (around image 150). The person to the right raises his hand, and a person walks by in the foreground. He takes down his hand around image 230. The person in the foreground then walks back and forth one time more.

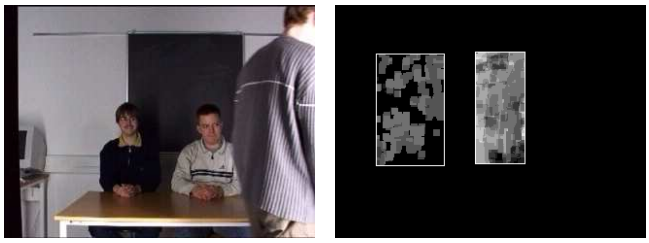


(a) Hand-Raise ROI Log Zeroth Moment (b) Face Search Window Moment Changes

Figure 7.16: Terms Contributing to the Measurement Noise for V4: The red graphs are for the person to the left in the image in Figure 7.15 and the blue graphs for the person to the right. The vertical lines indicate when hand-raises occur.

if the occluding object has a colour that under some kind of normal illumination could have been the colour of skin.

As the plots for video V4 in Figure 7.14 show, the measurement noise will become large compared to the process noise when a hand is raised and when the face becomes occluded. To verify that the measurement noise also becomes large when a person is moving in the background, similar plots have been made for image video V6. In this video, two persons are making hand-raises while a third person is moving and writing on a blackboard in the background. This is illustrated with the images in Figure 7.19. Note the low contribution from the face region in Figure 7.20(b), compared to the contribution when the face becomes occluded in Figure 7.16(b). From image 40, where the person in the background



(a) Image 864 (b) Image 864 ADP

Figure 7.17: Skin-Colour False Positives after Occlusion: If the face being tracked is occluded for so long that the tracker adapts to the colours of the object occluding the face, the re-adaption to the colours of the face when the occlusion ends may cause a lot of noise in the ADP, as it can be seen in (b).

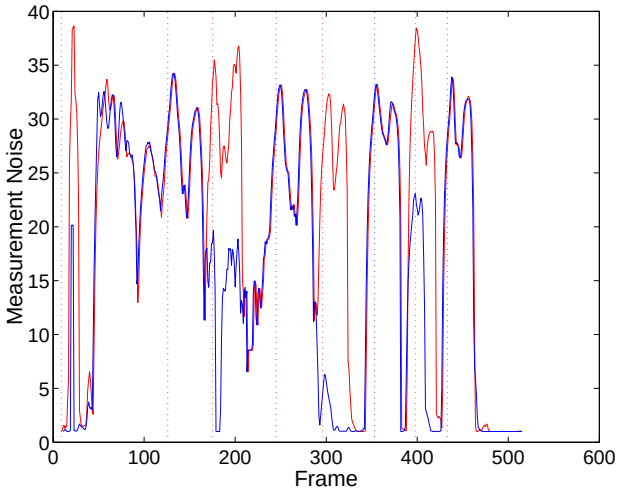


Figure 7.18: Measurement Noise for Kalman Filter: The plot shows the measurement noise standard deviation for each person for each frame in V6. The graph for the person to the left is plotted with red, while the graph for the person to the right is plotted with blue (see Figure 7.19). The vertical lines indicate when hand-raises were performed.

enters the hand-raise ROI of the person to the right, to image 300 where he leaves, the measurement noise in Figure 7.18 is relatively high between hand-raises. After he has left, the measurement noise between hand-raises becomes low.

The measurement noise should also be increased when changes in illumination occur, as these can result in false positives that can confuse the tracker. That this indeed does happen can be seen in Figure 7.21(a), which shows the changes in F^2 as the illumination colour changes three times. Each change result in a peak in the plot for F^2, this resulting in an increased measurement noise. As it can be seen from Figure 7.21(b), each type of illumination has it own averages. This confirms that when computing the average, the recent values should be weighted significantly higher than old values.

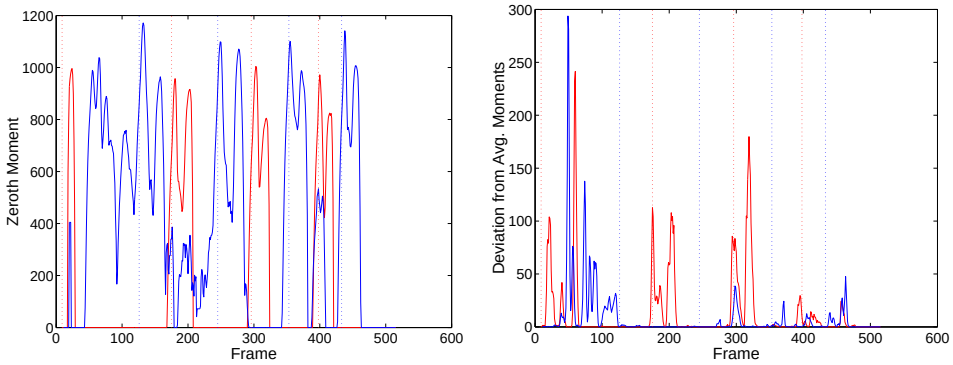
7.4.4 Tracker Accuracy Test

The tracking ability of the tracking algorithm – Mean Shift, ellipse fitting, and Kalman filter – has been evaluated for videos V1–V16, using the parameters described in the previous sections. This has been done by observing in which situations the tracker's

(a) Image 30 (b) Image 46 (c) Image 106 (d) Image 172

(e) Image 243 (f) Image 320 (g) Image 389 (h) Image 456

Figure 7.19: Events in V6: Two persons make hand-raise gestures while a third writes on a blackboard in the background.



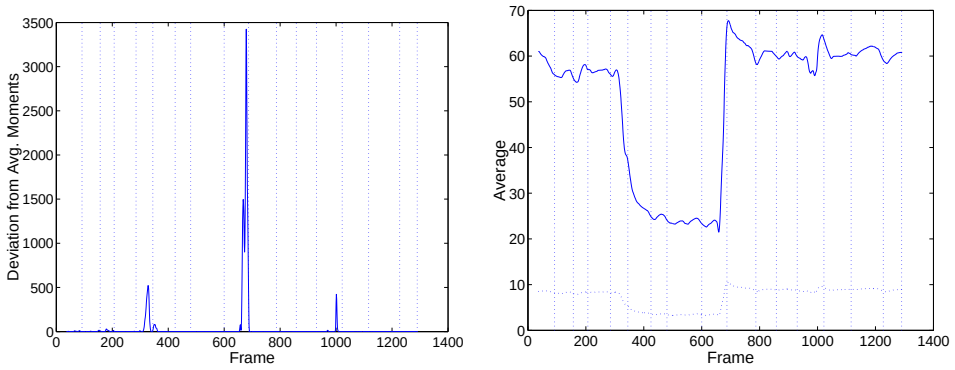
(a) Hand-Raise ROI Log Zeroth Moment (b) Face Search Window Moment Changes

Figure 7.20: Terms Contributing to the Measurement Noise for V6: The red graphs are for the person to the left in the image in Figure 7.15 and the blue graphs for the person to the right. The vertical lines indicate when hand-raises occur.

estimate deviates significantly from the center and size of the face. We consider the deviation significant, when the tracker's estimate is outside the border of the face.

In general, there is no problem tracking the face in the absence of occlusion, clutter, changes in illumination, and fast face movement. In fact, in most cases of occlusion and clutter, and in all cases of illumination change, the tracker's estimate does not deviate significantly from the center and size of the face, and even if it does, it will readjust to the correct center and size within a few seconds. The images in Figure 7.22 illustrate a small selection of situations that are handled well by the tracker.

In some cases where the face is occluded by a person walking by in the foreground, the tracker's estimate will move away from the center of the face while the face is occluded, because the center of mass of the skin-colour likelihood image moves. In all cases it has moved back when the occlusion ended. This is illustrated in Figure 7.23. The same



(a) Face Search Window Moment Changes (b) Average Moments

Figure 7.21: Moment Changes during Change of Illumination: The plot in (a) shows the deviation from the average value for F^2. The three peaks correspond to changes in illumination colour from white (3680 K) to yellow (2600 K), yellow to blue (6200 K), and blue to "less blue" (4700 K). The vertical blue lines indicate when hand-raises occur. The graphs in (b) show what the average moments are. The dotted line is the variance, the other is the zeroth moment.

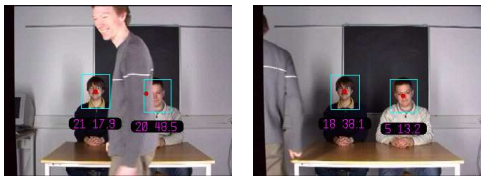
(a) V2 Image 107 (b) V2 Image 263 (c) V11 Image 510 (d) V12 Image 49

(e) V12 Image 329 (f) V13 Image 884 (g) V15 Image 500 (h) V15 Image 564

Figure 7.22: Examples of Clutter and Occlusion: These images are a few examples of the many cases, where the tracker handled clutter and occlusion well. The red dots are the *a posteriori* position estimates. The boxes are the Mean Shift search windows after the last iteration for the image. The number to the left below the search window is the unstability measure ξ, and the number to the right is the measurement noise standard deviation.

thing sometimes happens when the illumination changes, as illustrated in Figure 7.24. This happens because the skin-colour detection generates too high likelihoods for the background pixels.

Fast movement of the face is handled less well by the tracker. If the face moves fast, it may move outside the Mean Shift search window, which is centered at the estimated position from the Kalman filter. If this happens, the Mean Shift algorithm will not be able to center the search window at the face. A situation where this happens is illustrated in Figure 7.25, where the tracker looses the face and is eliminated because it cannot find any ellipses in the background.



(a) t = 0 sec. (b) t = 1.1 sec.

Figure 7.23: Tracker Estimate during Occlusion in V4: During an occlusion, the estimate of the tracker (red dot) moves away from the center of the face, but returns in 1.1 second when the occlusion has ended.

(a) t = 0 sec. (b) t = 1.8 sec. (c) t = 7.8 sec.

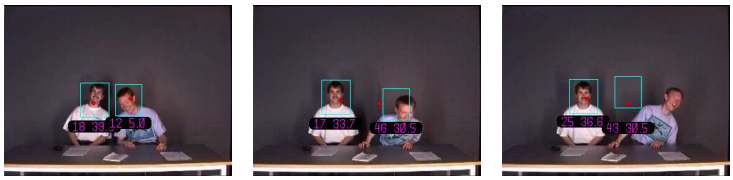
(d) t = 0 sec. (e) t = 5.6 sec.

Figure 7.24: Tracker Estimate during Illumination Change in V8 and V9: During change in the illumination, the tracker's estimate (red dot) moves away from the center of the face, but as the skin-colour detection adapts to the new illumination, the estimate moves back.

The face movement also results in an increased measurement noise. This means that the use of the Kalman filter's estimate as the initial center of the search window for each frame will cause the distance between the face and the tracker's estimate of the position to become larger for each frame, eventually causing the tracker to loose the face.[2]

In conclusion, the proposed face tracking scheme appears to work quite well for the videos we have used, although fast face movement can cause problems. However, fast face movement is not likely to happen when the person is seated and wish to get the attention of the system, but rather when the person is leaving. Therefore, it is acceptable that the face is lost in such cases.

[2] If the Mean Shift algorithm was used alone, perhaps it would be able to track the face, as the position found by Mean Shift in the previous image would be used as the initial center in the current image. However, without the Kalman filter the tracker would be more vulnerable to occlusion and clutter. Perhaps the tracker could be made to handle this situation, if it used both the previous center found by Mean Shift as well as the estimate from the Kalman filter as initial positions for Mean Shift for the current image. Of the two positions found by Mean Shift, the position where ellipse fitting is most successful could then be used as the new measurement for the Kalman filter.



(a) t = 0 sec. (b) t = 2.9 sec. (c) t = 4.3 sec.

Figure 7.25: Trackers Estimate during Movement in V15: During the somewhat exaggerated movement, the tracker looses the face because the distance between the face and the tracker's estimate of its position becomes to large.

7.4.5 Elimination of Trackers for Non-Face Objects

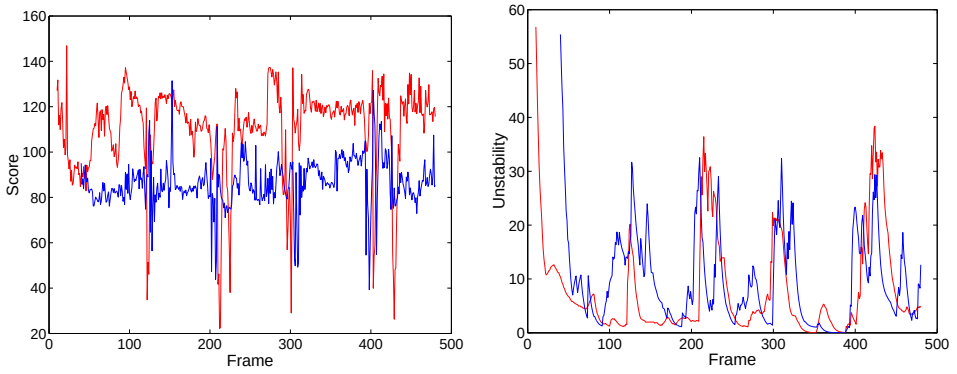
Trackers that are tracking non-face objects are eliminated using the two thresholds θ_{time} and ξ_{max}. θ_{time} limits the number of time steps that a tracker can "survive" without an ellipse being successfully fitted to the object it is tracking. ξ_{max} limits the average amount of jitter that is allowed for the measurements.

Figure 7.26(a) shows the ellipse fitting score for the two trackers tracking the faces in V4. This video was described in Section 7.4.3. The four drops in the score coincide with the occlusions by the person walking by in the foreground. The ellipse fitting threshold of 0.2 that we use corresponds to a score of 51 in the figure. As it can be seen from the figure, the score drops below this level during the occlusions (as one would expect). For images where the ellipse fitting score is below the threshold, the measurement is not used to update the tracker's estimate. θ_{time} should therefore be chosen to be at least as large as the longest amount of time that the ellipse fitting score is below the threshold, unless it is acceptable that the tracker looses the face during the occlusion.

On the other hand, θ_{time} should be as small as possible, since it is used to eliminate trackers caused by false positives. In Figure 7.27(a), the black curve is the ellipse fitting score for a hand, for which a tracker was started due to a false positive from the face detection. The score quickly drops to a low level when the hand is taken down and the tracker looses it, as the background does not resemble an ellipse. The same thing can be seen in Figure 7.28(a). Thus, it appears that trackers for false positives caused by hands can be eliminated by limiting θ_{time}. However, as long as the hand is raised, the ellipse fitting score may be above the threshold, and therefore the hand may be considered a face by the system. To reduce the risk that this happens, u_{min} – the minimum number of times that the tracker's estimate must be updated before the object it is tracking is considered a face – can be set to a "high" value, although this would mean that the video conference participants would have to wait for, perhaps, several seconds before their hand-raises will be detected.

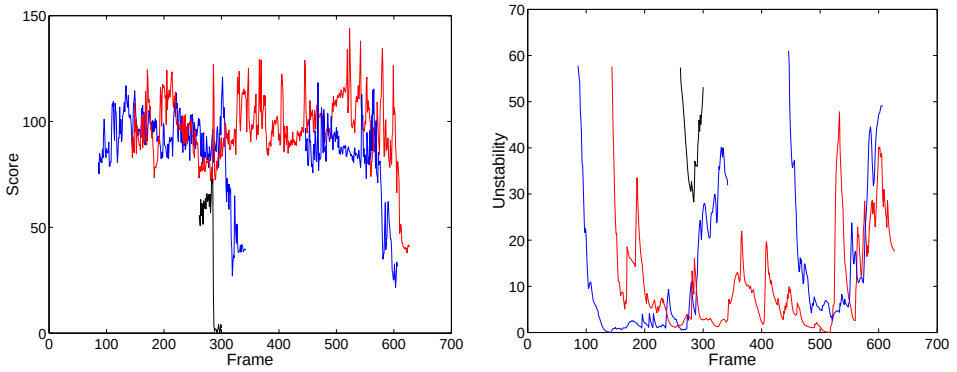
The computation of the unstability measure depends on a parameter a (see Equation 4.16 on page 47), that controls to which degree the past influence the current value of the unstability measure. We use the value $a = 0.2$, which has been determined empirically.

From the unstability plots in Figures 7.26(b)–7.28(b), it can be seen that the unstability measure is high for the false positives compared to the unstability measures for faces, except when the persons are entering and leaving. Unless it is accepted that the faces are lost during occlusions, ξ_{max} must be at least as high as the peaks in Figure 7.26(b), i.e. approximately 40. This value of ξ_{max} will also allow the trackers for faces in Figures



(a) Ellipse Fitting Score (b) Unstability Measure

Figure 7.26: Ellipse Fitting Score and Unstability Measure for V4: The ellipse fitting score for each of the two faces being tracked in V4 drops as the face becomes occluded. At the same time, the unstability measure increases.



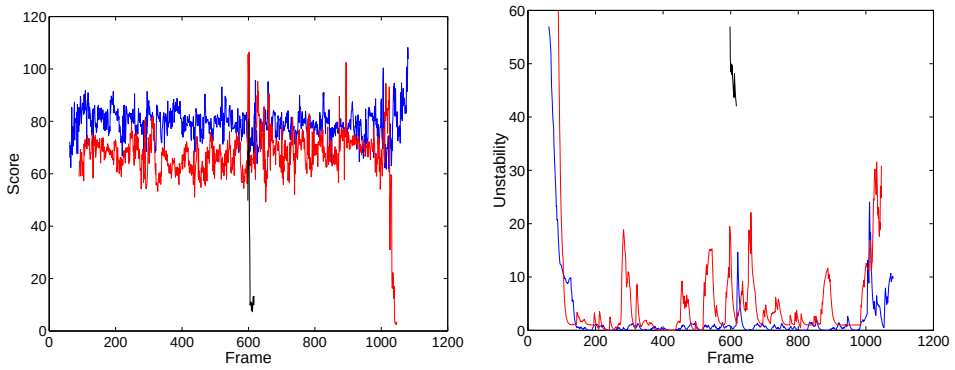
(a) Ellipse Fitting Score (b) Unstability Measure

Figure 7.27: Ellipse Fitting Score and Unstability Measure for V5: The blue and red graphs correspond to trackers tracking faces, while the black graphs correspond to a tracker created for a hand due to a false positive from the face detection.

7.27(b)–7.28(b) to survive, except when the persons are leaving in Figure 7.27(b), though it may be necessary to lower the initial value of the unstability measure for trackers. However, the hands have a higher unstability measure some of the time, and will thus be deleted.

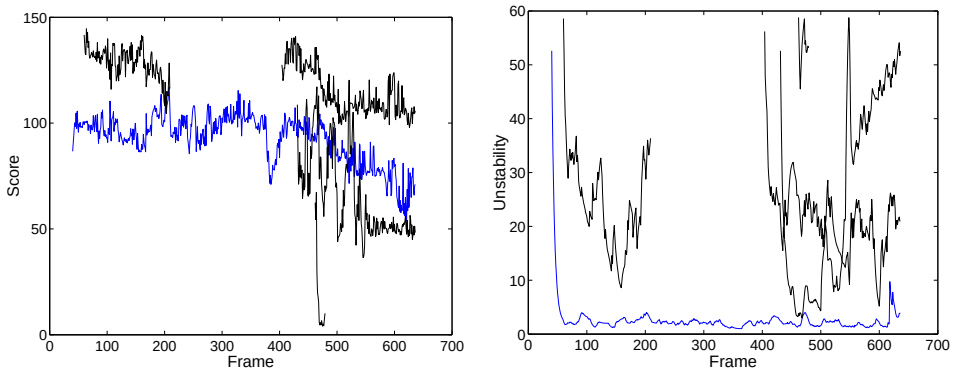
In Figure 7.29, the ellipse fitting score and unstability measure can be seen for V19. In V19, only one of the faces is detected. The black curves are for trackers due to false positives. The ellipse fitting score only drop to a sufficiently low level for one of the false positives. This false positive was caused by a hand. The other false positives are caused by other objects in the image, see Figure 7.30, and they all get a relatively high ellipse fitting score. Some of them can be eliminated with a ξ_{max} of 40. However, it appears that not all trackers for false positives can be eliminated using ξ_{max} and θ_{time} without at the same time causing trackers for faces to be deleted.

In conclusion, the method used for eliminating trackers for non-face trackers is only sufficient for eliminating trackers started for a hand while it is being raised or taken down.



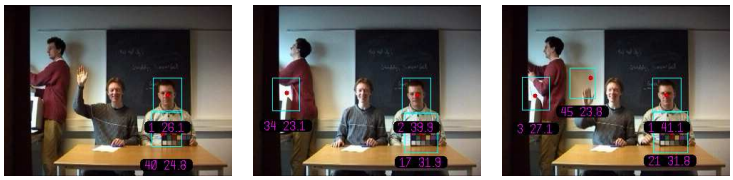
(a) Ellipse Fitting Score (b) Unstability Measure

Figure 7.28: Ellipse Fitting Score and Unstability Measure for V6: The blue and red graphs correspond to trackers tracking faces, while the black graphs correspond to a tracker created for a hand due to a false positive from the face detection.



(a) Ellipse Fitting Score (b) Unstability Measure

Figure 7.29: Ellipse Fitting Score and Unstability Measure for V19: The blue curve is for a tracker tracking a face, while the black curves are for trackers created for false positives from the face detection.



(a) Image 63 (b) Image 433 (c) Image 465

Figure 7.30: False Positives from Face Detection in V19: The tracker for the false positive in (a) moves towards the face and is deleted as the estimates of the two trackers overlap, but later a new false positive appears in the same place. Two other false positives appear on the wall and on a raised hand.

More information will be needed to eliminate trackers started for other types of objects.

7.5 Hand-Raise Detection Experiments

In this section, we describe experiments done to evaluate the ability of the system to detect hand-raise gestures, and to discriminate between hand-raise gestures and other types of events. Both of the Naive Bayesian Classifiers (NBCs) described in Chapter 5 are tested using several different combinations of parameters for the hand-raise detection.

7.5.1 Experiments Description

The hand-raise detection has been tested on V4, V6, V8–V10, and V12-V16. These videos cover illumination change, clutter, and occlusions, as well as valid hand-raise gestures. The performance is measured by the number of false positives and the number of false negatives. Here, a *false positive* is the detection of a hand-raise gesture when a hand-raise has not occurred, and can be caused by noise in the hand-raise ROI ADP. A *false negative* is when the system does not detect a hand-raise gesture even though it was performed. Ideally, both these numbers should be zero. However, as the experiments will show, this cannot be archived with the current system. Therefore, we attempt to find some sets of parameters for the hand-raise detection that provide reasonable trade-offs between these two performance measures.

The parameters that are examined for the hand-raise detection are:

- The amount of upward movement in the skin-colour likelihood images that must occur for something classified as a hand-raise by the NBC to be considered a hand-raise by the system.
- The number of times in a row that a hand-raise must be detected, before the system redirects the attention (i.e. camera) to the person that has raised his hand.

Summarizing from Chapter 5 on page 51, if the NBC classifies a blob in the ADP as a hand-raise, a score based on the shift in skin-colour likelihood center of mass inside in the bounding box of the blob is computed. The previous N likelihood images are considered. If the vertical shift from one image to the next is greater than ΔY and is directed upwards, a counter u is incremented. If it is greater then ΔY and directed downwards, a counter d is incremented. When all of the N images have been considered, the score is computed as $(u - d)/N$. This results in a number between -1 and 1.

If this score exceeds a threshold $score_{min}$, a hand-raise counter r is incremented. At the same time, a timer is reset. This timer is incremented for each image, and if it exceeds a threshold T, r is set to 0. If r reaches a threshold r_{min}, the system will assume that a hand-raise has occurred. Thus, r_{min} blobs must be classified as hand-raises by the NBC with no more than T images between the blobs for a hand-raise to be detected.

For practical reasons, we have chosen to fix the values of N and T, and only experiment with ΔY, $score_{min}$, and r_{min}. T has been set to 10, corresponding to 0.8 second. This threshold is rarely, if ever, exceeded in the videos we use, but appears to be small enough to avoid that blobs that are not caused by hand-raises are grouped with blobs that are caused by hand-raises. This must be avoided as one of the performance measures is false positives.

N has been set to 9, since the ADP is based on 9 likelihood images. With this choice of N, one end of the blob will, when the hand is being raised or taken down, correspond to

the position of the hand in the current image, and the other end of the blob will correspond to the position of the hand 9 images ago. The number of images used for the ADP cannot be changed without estimating new pdfs for the NBCs, which would require much work. Therefore, we will not experiment with this number.

The experiments have been done by counting the numbers of false positives and false negatives for different combinations of values for ΔY, $score_{min}$, and r_{min}. This has been done for both of the NBCs. For brevity, we will refer to the NBC in Figure 5.3 on page 56, which use a position attribute and a size attribute, as NBC1, and the NBC in Figure 5.4, which split each of these attributes in 2 and instead use x position, y position, area, and height/width-ratio attributes, as NBC2.

The videos have been divided into different groups, depending on the type of events that they contain. These groups are:

- Videos containing mostly valid hand-raises.
- Videos containing valid hand-raises, but also hand-movements that are not valid hand-raises, as well as persons walking by in the background and foreground.
- Videos containing valid hand-raises and illumination changes.

Images from the videos can be found in Appendix D on page 145.

Below, a section for each of these groups can be found, where data illustrating the performance of the hand-raise detection for each group is presented. Afterwards, data to illustrate the performance for the entire set of videos is presented.

7.5.2 Valid Hand-Raises

The videos V14–V16 contain mostly valid hand-raises, all performed with bare arms. The total number of hand-raises in these videos is 37.

In Figure 7.31, the hand-raise counter r is plotted for fixed values of ΔY and $score_{min}$. It is indicated with vertical dotted lines when hand-raises occur (this has been manually recorded). As it can be seen from the figure, r is often high when hand-raises occur, but sometimes r is non-zero when a hand-raise is not occurring and sometimes r is zero even though a hand-raise does occur. The effect of r_{min} is that of thresholding the graph. When doing the thresholding, a trade-off is made between the number of false positives and false negatives. This can be seen from the bar graphs in Figure 7.32, where the numbers of false positives and false negatives are shown for different combination of the parameters r_{min} and $score_{min}$ and a fixed ΔY. (The graphs for other values of ΔY show a similar dependency on the parameters.)

From these histograms, it appears that when $\Delta Y = 3$ reasonable values for r_{min} are in the range 1–4 and for $score_{min}$ in the range 0.3–0.5, as both the number of false positives and the number of false negatives are low for these parameter-ranges.

Based on histograms as those in Figure 7.32, the plots in Figure 7.33 have been produced. This has been done by, for each set of parameter-pairs that correspond to a particular number of false positives, finding the numbers of false negatives that correspond to these parameter-pairs. Thus, pairs consisting of a number of false positives and a number of false negatives are produced. The locations of these pairs are shown with diamonds in the figure. The graphs in Figure 7.33 show how low a number of false negatives one can

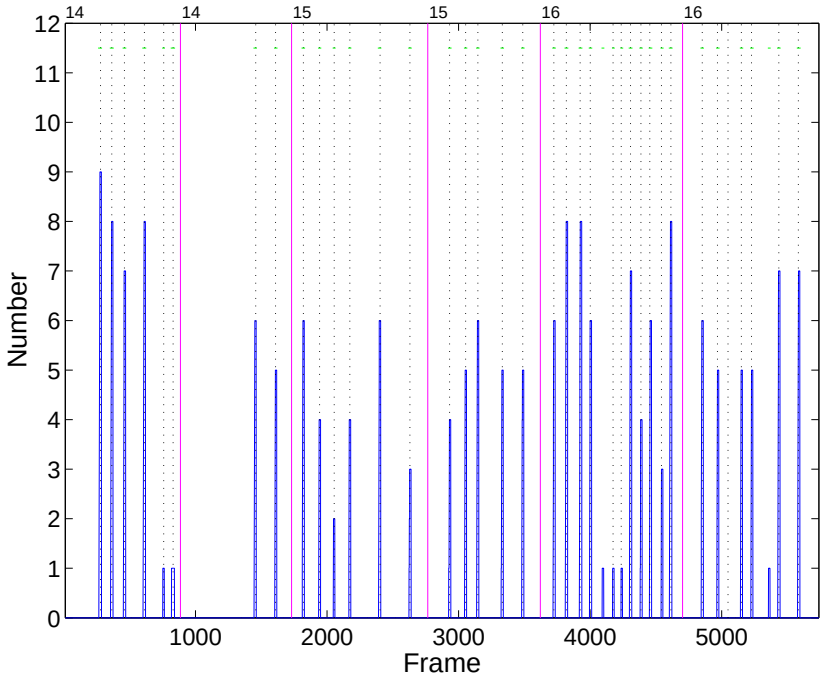
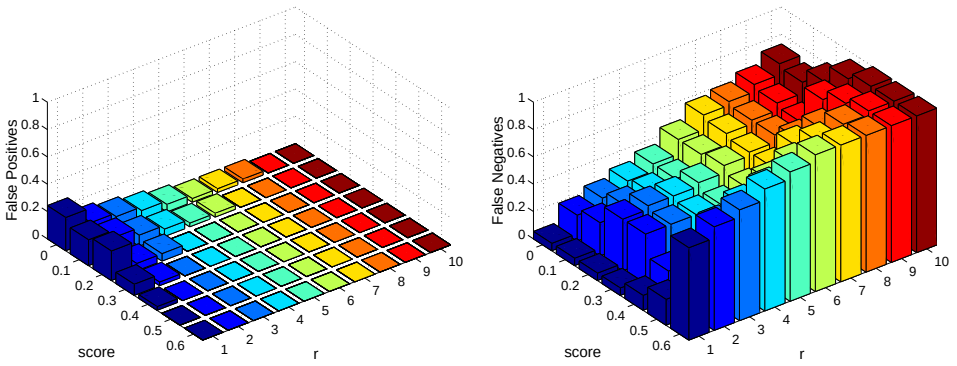


Figure 7.31: Hand-Raises in V14–V16: The black dotted lines indicate when hand-raises occur. The magenta lines indicate when one video ends and another starts. The number at the top indicates which video that starts at that line. Some of the videos are included two times, because they contain two persons; that is, there is a section for each person. The blue graph is the value of the hand-raise counter. This graph has been produced for $score_{min} = 0.4$ and $\Delta Y = 3$ using NBC1.



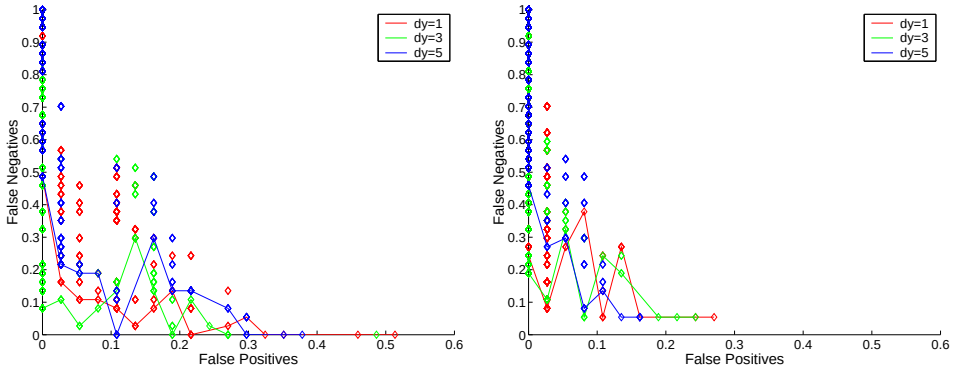
(a) False Positives

(b) False Negatives

Figure 7.32: False Positives and Negatives for V14–V16: The histograms shows the frequencies of different combinations of parameters r_{min} and $score_{min}$ in the sets of false positives and false negatives for $\Delta Y = 3$. The numbers on the vertical axes have been normalized by dividing with the number of hand-raises.

achieve for this particular set of videos if a particular number of false positives is accepted and the value of ΔY fixed.

As one would expect, the number of false negatives falls as the number of false positives grows. The number of false negatives in general appears to be smaller for NBC1 than for NBC2, but the number of false positives larger. During the experiments, it was observed that NBC1 tended to classify far too many events as hand-raises, while NBC2 performed



(a) NBC1 (b) NBC2

Figure 7.33: False Negatives vs. False Positives for V14–V16: The graphs show the minimum number of false negatives that occurs for a particular number of false positives, while the diamonds show all the numbers of false negatives that occur for this number of false positives. The numbers on the axes have been normalized by dividing with the number of hand-raises.

better in this respect. This may be the cause of the difference, as an increase in the number of hand-raise candidates will be followed by an increase in the number of false positives, unless the subsequent hand-raise verification discards all of the extra candidates. Thus, the experiments with NBC1 could be considered more a test of the hand-raise verification that follows the classification with the NBC, than a test of the NBC.

For both of the NBCs in Figure 7.33, diamonds appear near 0, that is, the parameter pairs that correspond to these diamonds produce relative low numbers of false positives and negatives. A few of these parameter-pairs are presented in Table 7.18.

NBC	ΔY	r_{min}	$score_{min}$	False Positives		False Negatives		Distance
				abs.	%	abs.	%	
1	3	1	0.4	2	5.4	1	2.7	0.06
2	1	1	0.4	1	2.7	3	8.1	0.09
1	5	1	0.3	4	10.8	0	0.0	0.11
1	3	2	0.4	0	0.0	5	13.5	0.14

Table 7.18: Hand-Raise Detection Results for V14–V16: The numbers of false positives and negatives are listed together with the parameters that were used to produce them. The false positives and negatives are expressed both as an absolute number which can be compared to the total number of hand-raises in these videos (37), and as a percentage of this number. The distance shown is the distance from $(0, 0)$ to the diamond corresponding to the set of parameters in Figure 7.33. Note that several other combinations of parameters can produce results similar to those presented in this table.

As it can be seen from Table 7.18, it is possible to obtain either no false positives or no false negatives, but not both at the same time.

7.5.3 Clutter and Occlusion

V4, V6, V12, and V13 contain many examples of clutter and occlusion, as well as valid hand-raises with both bare and covered arms. These videos have been used to test the ability of the hand-raise detector to discriminate between valid hand-raises and several other types of events. Figure 7.34 show where hand-raises occur in these videos. The

total number of hand-raises is 24.

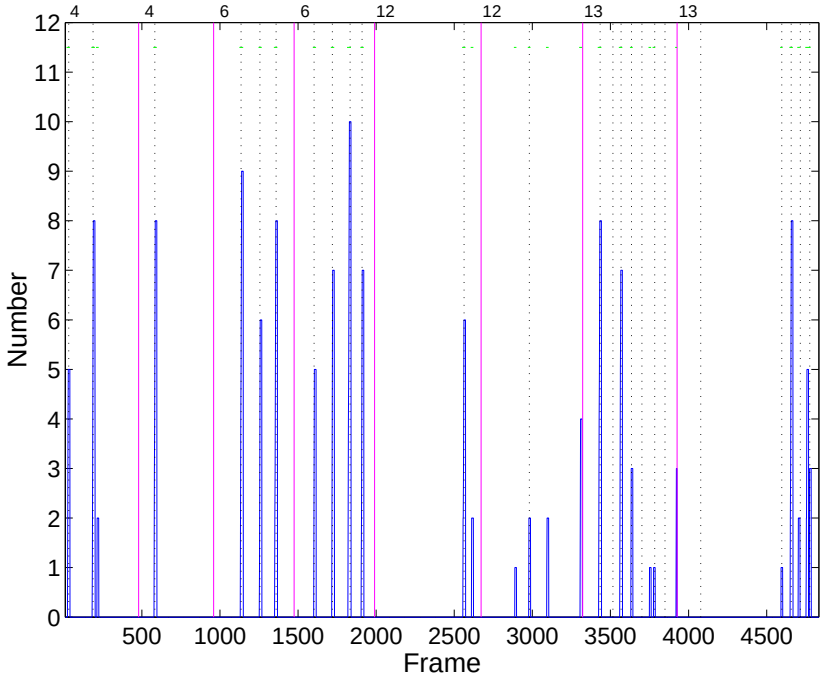


Figure 7.34: Hand-Raises in V4, V6, and V12–V13: The black dotted lines indicate when hand-raises occur. The magenta lines indicate when one video ends and another starts. The number at the top indicates which video that starts at that line. Some of the videos are included two times, because they contain two persons; that is, there is a section for each person. The blue graph is the value of the hand-raise counter. This graph has been produced for $score_{min} = 0.4$ and $\Delta Y = 3$ using NBC1.

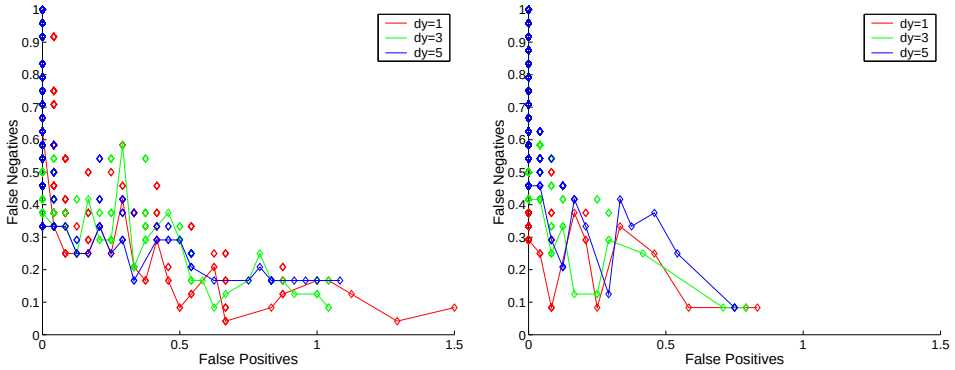
In V4, a person walks back and forth in the foreground two times, occluding the video conference participants (see Figure 7.15 on page 95). In V6, a person in the background is writing on a blackboard during most of the video (see Figure 7.19 on page 97). V12 contains mostly hand-movements that are not hand-raises, but also two valid hand-raises as it can be seen in Figure 7.34. V13 contains valid hand-raises performed while a person is walking back and forth in the background, and a single occlusion by a person walking by in the foreground. Six of the hand-raises coincide with the person walking in the background being inside the hand-raise ROI.

To evaluate the performance, a graph similar to that in Figure 7.35 has been produced. This graph is shown in Figure 7.35. For some of the diamonds close to 0 in this figure, the numbers of false positives and negatives are shown together with the corresponding parameters in Table 7.19.

In spite of the presence of clutter and occlusions, it is possible to obtain no false positives, but only at the expense of a rather high number of false negatives.

7.5.4 Illumination Changes

V8–V10 each contain several changes of illumination colour, and several valid hand-raises are performed under different kinds of illumination and in some cases also while the illumination is being changed. All hand-raises are done with covered arms. Figure 7.36 shows when hand-raises occur. The total number of hand-raises is 41, and 13 of the hand-raises are done during an illumination change.



(a) NBC1 (b) NBC2

Figure 7.35: False Negatives vs. False Positives for V4, V6, and V12–V13: The graphs show the minimum number of false negatives that occurs for a particular number of false positives, while the diamonds show all the numbers of false negatives that occur for this number of false positives. The numbers on the axes have been normalized by dividing with the number of hand-raises.

NBC	ΔY	r_{min}	$score_{min}$	False Positives		False Negatives		Distance
				abs.	%	abs.	%	
2	1	1	0.4	2	8.3	2	8.3	0.12
2	1	1	0.5	1	4.2	6	25.0	0.25
2	1	2	0.4	0	0.0	7	29.2	0.29
1	5	7	0.3	0	0.0	8	33.3	0.33

Table 7.19: Hand-Raise Detection Results for V4, V6, and V12–V13: The numbers of false positives and negatives are listed together with the parameters that were used to produce them. The false positives and negatives are expressed both as an absolute number which can be compared to the total number of hand-raises in these videos (24), and as a percentage of this number. The distance shown is the distance from $(0, 0)$ to the diamond corresponding to the set of parameters in Figure 7.35. Note that several other combinations of parameters can produce results similar to those presented in this table.

Figure 7.37 shows the number of false negatives vs. the number of false positives, and Table 7.20 lists the parameters for some of the diamonds close to 0 in Figure 7.37.

At it can be seen from Table 7.20, with the right parameters, illumination change can be handled without any false positives.

NBC	ΔY	r_{min}	$score_{min}$	False Positives		False Negatives		Distance
				abs.	%	abs.	%	
1	1	3	0.4	1	2.4	2	4.9	0.05
1	3	4	0.3	0	0.0	4	9.8	0.10
2	1	1	0.5	0	0.0	4	9.8	0.10
2	3	1	0.4	0	0.0	4	9.8	0.10

Table 7.20: Hand-Raise Detection Results for V8–V10: The numbers of false positives and negatives are listed together with the parameters that were used to produce them. The false positives and negatives are expressed both as an absolute number which can be compared to the total number of hand-raises in these videos (41), and as a percentage of this number. The distance shown is the distance from $(0, 0)$ to the diamond corresponding to the set of parameters in Figure 7.37. Note that several other combinations of parameters can produce results similar to those presented in this table.

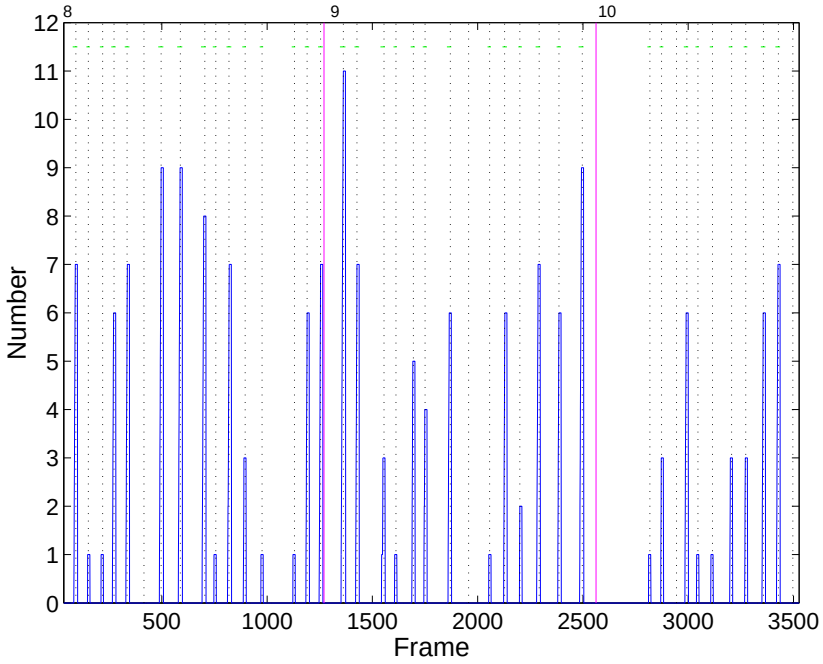
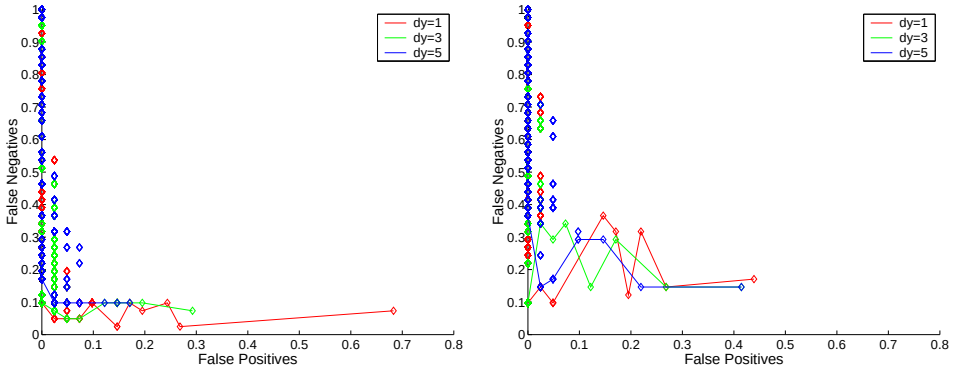


Figure 7.36: Hand-Raises in V8–V10: The black dotted lines indicate when hand-raises occur. The magenta lines indicate when one video ends and another starts. The number at the top indicates which video that starts at that line. Some of the videos are included two times, because they contain two persons; that is, there is a section for each person. The blue graph is the value of the hand-raise counter. This graph has been produced for $score_{min} = 0.4$ and $\Delta Y = 3$ using NBC2.



(a) NBC1 (b) NBC2

Figure 7.37: False Negatives vs. False Positives for V8–V10: The graphs show the minimum number of false negatives that occurs for a particular number of false positives, while the diamonds show all the numbers of false negatives that occur for this number of false positives. The numbers on the axes have been normalized by dividing with the number of hand-raises.

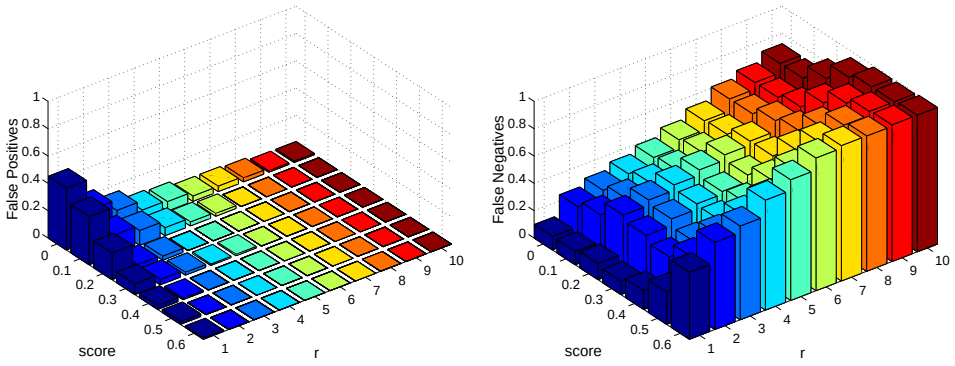
7.5.5 Overall Performance

To get an impression of the overall performance that can be obtained, the hand-raise detection has been tested on all the videos used above, i.e. V4, V6, V8–V10, and V12–V16. The total number of hand-raises in all of these videos is 102.

Histograms showing the frequencies of different combinations of the parameters in the

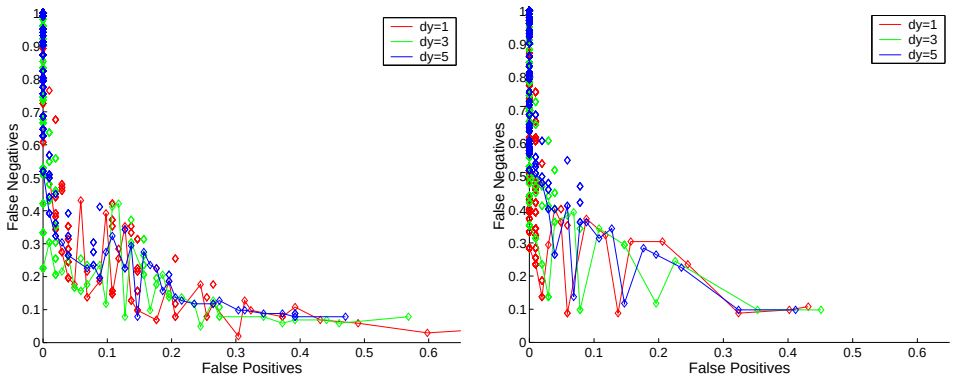
sets of false positives and false negatives can be found in Figure 7.38. A plot of false negatives vs. false positives is presented in Figure 7.39, and examples of the numbers of false positives and negatives for different parameter combinations are given in Table 7.21.

Zero false positives can be obtained, but only if a large number of false negatives is accepted. The false positives are usually caused by events that are within the control of the video conference participants. Therefore, some of the false positives that occur in these videos would probably be acceptable, as the participants could behave in such a way that the false positives were avoided. I.e. a trade-off, e.g. 5.9% false positives and 8.8% false negatives, might be acceptable.



(a) False Positives (b) False Negatives

Figure 7.38: False Positives and Negatives for All Videos: The histograms shows the frequencies of different combinations of parameters r_{min} and $score_{min}$ in the sets of false positives and false negatives for $\Delta Y = 3$. The numbers on the vertical axes have been normalized by dividing with the number of hand-raises.



(a) NBC1 (b) NBC2

Figure 7.39: False Negatives vs. False Positives for All Videos: The graphs show the minimum number of false negatives that occurs for a particular number of false positives, while the diamonds show all the numbers of false negatives that occur for this number of false positives. The numbers on the axes have been normalized by dividing with the number of hand-raises.

NBC	ΔY	r_{min}	$score_{min}$	False Positives		False Negatives		Distance
				abs.	%	abs.	%	
2	1	1	0.4	6	5.9	9	8.8	0.11
2	1	1	0.5	2	2.0	14	13.7	0.14
1	3	1	0.4	13	12.7	8	7.8	0.15
1	1	1	0.5	18	17.6	7	6.9	0.19
1	3	7	0.3	0	0.0	23	22.5	0.23
2	1	2	0.4	1	1.0	24	23.5	0.24
2	1	3	0.4	0	0.0	29	28.4	0.28

Table 7.21: Hand-Raise Detection Results for All Videos: The numbers of false positives and negatives are listed together with the parameters that were used to produce them. The false positives and negatives are expressed both as an absolute number which can be compared to the total number of hand-raises in these videos (102), and as a percentage of this number. The distance shown is the distance from $(0, 0)$ to the diamond corresponding to the set of parameters in Figure 7.39. Note that several other combinations of parameters can produce results similar to those presented in this table.

7.6 Experiments Conclusions

In this chapter, we have described the experiments made to evaluate the performance of the methods for FoA, face verification, face tracking, and hand-raise detection. The performance of the FoA methods was measured by computing the distance between the average skin-colour likelihood in the face and background area. Furthermore, the average computation per image time was used to detect whether a method which increased the face-background distance also increased the computation time. The experiments showed that the best method to use was the weighted Gaussian, which gave a face-background distance of 0.286. At the same time, the computation time of 34.7ms was close to the average of the four evaluated methods, so the weighted Gaussian method should be a good compromise between distance and computational speed. Afterwards, experiments were made using moment constraints on the weighted Gaussian method. These made the face-background distance increase to 0.341 and at the same time the computation time increased by less than 1ms. It was therefore concluded that moment constraints definitely are worth using.

Before evaluating the performance of the face verification methods, experiments were made to find out if the use of erosion and dilation on the likelihood image made by the FoA methods could increase the face-background distance. The results of the experiments clearly indicated that this was the case – i.e. when using $1\times$erosion and $6\times$dilation, the distance increased from 0.341 to 0.594. To measure the performance of the face verification methods, experiments were first made on each method alone to find its optimal threshold values. Thereafter, the performance of each method was found by counting the number of false positives and false negatives using the same threshold values. These experiments showed that the rectangle method, although it was very simple and fast, was able to remove almost 70% of the false positives and at the same time it only made 0.6% false negatives. This method should therefore be used as an initial rough filter, before the more computational demanding methods are used. The template matching method was without doubt the most computational demanding, having an average computation time of 122ms. However, by combining the methods serially, it was possible to decrease the average computation time to 105ms, because many face candidates were removed by the simple and fast rectangle and solidity methods before template matching was done. Serial combination of the methods also made the false positives drop from 28.8% when using

the ellipse fitting method to 10.7%. At the same time, the number of false negatives rose from 7.3% when using the ellipse fitting method to 15.8%. However, because false positives may be used to start new trackers and thereby can have influence on the hand-raise ROIs, it is more important to have a low number of false positives than a low number of false negatives. A final face verification experiment was therefore made using stricter thresholds, which made the number of false positives drop to 0.0% and the number of false negatives rise to 53.6%. At the same time, the average computation time per image decreased to 67ms due to more face candidates being removed by the rectangle and solidity methods, before template matching and ellipse fitting were done.

To be able to test the face tracker and the tracker manager, we determined a set of parameters to use empirically, and verified that these parameters were reasonable. Using these parameters, the performance of the face tracker was evaluated. We found that the face tracker works as desired in the presence of clutter, occlusions, and illumination change, but fast face movement can cause the tracker to loose the face. However, this is acceptable as it rarely will occur when the person is seated and participating in the video conference. We also tested ability of the tracker manager to eliminate trackers for non-face objects. Such trackers may be started due to false positives from the face detection. We found that it only was possible to eliminate such trackers when they were started for a raised hand.

Finally, experiments were performed to evaluate the hand-raise detection. This was done by measuring the numbers of false positives and false negatives for different sets of parameters for the hand-raise detector. In these experiments, we discovered that false positives, that is, the detection of a hand-raise when none has occurred, could only be eliminated if a large number of false negatives is accepted. When using the complete set of videos used for the hand-raise detection experiments, the smallest number of false negatives that could be obtained if no false positives was accepted, was 22.5%, that is, 22.5% of the hand-raises were not detected by the system. However, as the events causing false positives are within the control of the video conference participants, a low number of false positives will be acceptable. If 2% false positives are accepted for the complete set of videos, the number of false negatives can be reduced to 13.7%. If 5.9% false positives are accepted, the number of false negatives is 8.8%.

Chapter 8

Future Work

In this chapter we will describe future work which may be made to the VICOWIJOY system.

8.1 Focus of Attention

The methods for focus of attention, which we have investigated and experimented upon in this project, all expect that the faces are illuminated by a single colour. This will not always be the case if we want to be able to do video conferences in less controlled environments. If e.g. the conference takes place in a room with fluorescent lamps in the ceiling and the sun is shining through a window, half of a face may be illuminated by the sun and the other half by the fluorescent lamps. The CCT of a fluorescent lamp is around 3000K and the CCT of direct sunlight around 5700K (refer to Figure C.1 on page 138), so if a Gaussian model is estimated from the face, the center of mass would be right between the two illumination sources. Furthermore, the chromaticity r variance would be very large and therefore be constrained if moment constraints are used. This would lead to that none of the actual skin-colours in the face would be covered by the Gaussian model, and it would therefore not be found in the next image. One way of solving this problem could be to use LUTs instead of Gaussian models to detect the skin-colours. I.e. it is possible to have several skin chromaticity distribution groupings in a LUT, because it is computed directly from the skin-colours in the face and not estimated as the Gaussian models are. However, the experiments made on LUTs and Gaussian models in Section 7.2 on page 74 showed that Gaussian models outperform LUTs when the face is illuminated by a single colour. A more promising method is the use of Gaussian mixture models described by McKenna *et al.* in [37]. A Gaussian mixture model describes the skin-colours using several Gaussian models and should therefore be able to describe the skin-colours in the above mentioned example using a combination of two Gaussian models instead of one. It would be interesting to investigate and experiment further on the use of Gaussian mixture models. When using those instead of single Gaussian models it should be possible to do video conferencing in environments with fewer restrictions on the illumination.

8.2 Face Verification

The experiments made on the face verification methods in Section 7.3 on page 81 showed that it was possible to avoid (almost) false positives. This was at the cost of a high number

(53.6%) of false negatives, which does not matter that much since a video stream of 12.5 Hz still makes it possible to find a face several times per second. However, it would be nice if the number of false negatives could be lowered, because the face trackers use the outcome of the face verification to ensure that they continue to track the right objects. Finding more of the faces in the images should therefore make the trackers able to perform better.

One way to improve the face verification is to make it possible to find faces that are rotated in the image plane. In [34] Saber and Tekalp computes the center of mass and the covariance matrix for the part of the skin-colour likelihood image that is covered by a face candidate. Afterwards, they find the eigenvalues and the corresponding eigenvectors of the covariance matrix and computes an ellipse from these values. Thereafter, ellipse fitting is done in the gradient image by computing the average gradient value of the perimeter of the ellipse. Based on the eigenvalues upper an lower sizes of the radii in the ellipse are used to constrain the search for the best fitting ellipse. A great advantage of this method is that it makes it possible to find rotated ellipses (faces). If the temporal positions of the template matching and ellipse fitting in the face verification process were swapped, the ellipse fitting could give information to the template matching about the orientation of the face. Furthermore, the size of the face could also be given to the template matching. This would lower the computation time of the template matching since it would not be necessary to use an image pyramid – instead the face candidate could simply be resized to fit the size of the template (or vice versa). Moreover, a bank of templates rotated at different angles could be loaded at the start of the system. Based on the orientation of the ellipse, a template with the same orientation could then be used for template matching. This should make it possible to detect rotated faces in the template matching method.

Another and more simple way of computing the orientation of a face candidate is simply to compute the height/width ratio of its bounding box. If e.g. the height/width ratio of an upright face is set to 1.25, lower values would indicate that the face is rotated and a value of 0.8 that the face is rotated $\pm 90^{\circ}$. A problem with this method is that the rotation direction is unknown. Therefore, the following ellipse fitting method would have to be done at two orientations. It could nevertheless be interesting to implement both this method and the previously mentioned method where eigenvalues and eigenvectors are computed. Experiments could then be made to compare their accuracy and computational efficiency.

In [2] Menser and Wien use the compactness of the face candidates in the thresholded (binary) likelihood image to classify them as faces or non-faces. The compactness is found by dividing the area of a face candidate with the squared number of pixels along its perimeter – i.e. $compactness = \frac{Area}{Perimeter^2}$. Since a face is elliptic in shape its compactness is high and a lower threshold can therefore be used to remove non-faces. This method is simple and computational efficient and if used before the ellipse fitting and template matching, it could probably decrease the number of face candidates which have to be verified by these methods. Furthermore, we already follow the perimeters of the face candidates in the contour segmentation method and can therefore compute their lengths at the same time. This will make the compactness method even more computational efficient.

Finally, we have observed that significant for the faces of people that are participating in video conferences is that they do not move very much. This knowledge can be used for face verification by setting a maximimum average movement threshold of the face candidates in the likelihood image. The method should especially make it possible to handle the situations where hands are detected as face candidates during a hand-raise.

I.e. these hands will inevitably have a large amount of movement and can therefore be removed from the face candidate list.

8.3 Face Tracking

The face tracking scheme presented in this report was found to work quite well. However, two problems were discovered in the experiments. The first is that the tracker's estimate of the position of the face, when the face is occluded and when the illumination is changed, may move away from the center of the face. The second is the tracker's inability to track fast moving faces.

The first problem could be solved by using better methods for detecting occlusions and illumination changes. We have primarily done this detection of occlusion and illumination changes by considering the skin-colour likelihood moments of the face area, but information deducted from the rest of the image would also be useful. A change in illumination colour will not only affect the pixels in the face area, but the entire image. Therefore, we expect it would be possible to detect the change in illumination colour by considering the colours of the pixels in the rest of the image, or perhaps just a subset of it. This could be used to increase the measurement noise level for the Kalman filter during illumination change, to avoid that the estimate moves too far.

Occlusions by a person walking by in the foreground could probably be detected by considering the overall amount of movement in the image, as this person will be closer to the camera than the video conference participants, and move significantly more when he is walking. Even when he is not walking it would be possible to detect his presence by comparing the current image to an average image computed using the previous images. The knowledge obtained this way could be used to control the measurement noise level. It could also reduce the risk of false positives from the hand-raise detection, as a video conference participant probably would not raise his hand while occluded, and if he did, it would make sense to ignore it, as the camera cannot zoom in on his face.

The second problem is primarily related to the measurement noise, which was found to be too high during fast face movement. If occlusions and illumination changes were detected in other ways, e.g. as suggested above, the moments of the face area could be given less weight in the computation of the measurement noise. The measurement noise would then be lower when the face is moving, since this movement does not affect the contents of the hand-raise ROIs, unless the tracker looses the face, and the face enters one of these ROIs.

In addition to considering more information in the computation of the measurement noise, the information could be integrated into the set of numbers needed for the Kalman filter in other ways. One way would be to train a Bayesian network to compute the probabilities of various events, such as occlusion and illumination change, using data collected from different video conference situations. The measurement noise could then be based on these probabilities.

The face detection occasionally results in false positives. When this happens, a face tracker is started for something that is not a face. Our experiments showed that the methods used for eliminating these trackers could not handle all cases of false positives. This also means that if a tracker looses the face it is tracking, it may not be eliminated. Therefore, more should be done to continuously verify that the object being tracked is a face, e.g. using some or all of the methods used for face detection.

We only attempt to track the faces of persons that are seated, as this appears to be suf-

ficient for our purposes, though also knowing the positions of the faces of persons who are coming, leaving, or walking by could be used for adjusting the measurement noise. However, the tracking system could be extended to allow persons to e.g. get up, and walk over to a blackboard to illustrate something. This would require the tracker to be able to track the head of the person when he is not facing the camera. The tracking could still be done using Mean Shift, but instead of using a skin-colour likelihood image, a skin-or-hair-colour likelihood image must be used.

Finally, it must be investigated how the control of the PTZ-camera can be done using the information available from the face tracker, i.e. 2D position and size. Since two cameras are available, the 3D position could also be obtained, but this may not be necessary, since the size of the face in the PTZ-camera image could be used to control the zoom.

8.4 Hand-Raise Detection

It would be worthwhile to investigate ways to improve the hand-raise detection. This could be done by including more or different information variables in the Bayesian network, and by training the network using a larger dataset. In particular, the dataset should probably include a more varied selection of noise to reduce the number of false positives. Since Gaussian approximations may not be appropriate for all probability density functions, the use of other functional approximations or look-up tables could be examined.

Additional information variables could provide more information about the shape of the hand-raise trace in the ADP. Alternatively, the idea of turning the trace into a single connected component using morphological operations could be abandoned, and other measures, e.g. the moments of the hand-raise ROI, could be used.

Additional verification steps could be added to eliminate false positives from the Bayesian network. For instance, the gradient image could be searched in an attempt to find the outline of the arm, using e.g. an approach similar to the one presented in [43]. Since the face position is known and the hand position can be determined from the skin-colour likelihood image, the search space for the elbow should not be large, in particular because it can be assumed that the hand is raised. Another approach would be to produce a silhouette image for the hand-raise ROI, as described in [25]. This is done by subtracting a background image from the current image. The background image could be an average image for the hand-raise ROI, since there – at least in some setups – will not be much change in this area, except when a hand is being raised.

Chapter 9

Conclusions

In this report we have investigated and experimented upon methods for use in a video conferencing system, where a speaker gets the attention by raising his right hand. The system consists of two cameras; a panorama camera in which the faces of the participants are tracked and hand-raises detected, and a PTZ-camera which must automatically zoom in on a speaker when he raises his hand. Zooming in on the speaker ensures that other participants located in other video conference rooms can see the facial expressions of the speaker. In the implemented version of the system no cameras are used. Instead they are emulated using images from AVI-files as the panorama camera and digital zooming in the same images as the PTZ-camera.

Methods for the initial focus of attention (FoA) in the system have been investigated. These have been based on skin-colour detection, using either lookup tables (LUTs) or Gaussian models to describe the skin chromaticity distribution. The skin-colour likelihoods of the pixels in the input image are found using the LUT or Gaussian model and a likelihood image made. A large focus was placed on making it possible for the system to adapt to changes in illumination colour. This is because the skin chromaticity distribution moves along the skin locus when the correlated colour temperature (CCT) of the illumination source changes. Therefore, using a LUT or a Gaussian model computed from a collection of training images would only work as long as the illumination colour is the same as in the training images. Four methods of illumination adaption were investigated – simple LUT, ratio LUT, ratio Gaussian, and weighted parameters of Gaussian. Experiments were made by updating the LUT or Gaussian model using a fixed position and size of the face in the images, and calculating the average skin-colour likelihood in the face area and the background area. When using the weighted Gaussian method the best result was achieved – i.e. it gave the longest distance between the average skin-colour likelihood in the face area and in the image area. To avoid the situation where inaccurate tracking of a face leads to a "wrong" skin chromaticity distribution in the LUT or Gaussian model, we investigated a skin-colour model invented by Störring *et al.* in [27]. This investigation revealed that under changing CCTs the center of mass of the skin chromaticity distribution follows a locus which is close to the Planckian locus of Blackbody radiators. Moreover, the areas of the distributions were close to the same and the distributions were in most cases rotated clockwise along the chromaticity r axis. Using this knowledge to constrain the position, shape, size, and rotation of the skin chromaticity distributions, it was possible improve the experiment results achieved when using the weighted parameters of Gaussian method.

To build a list of face candidates based on the likelihood image made by the FoA meth-

ods, the image was first eroded and dilated. Experiments showed that this significantly increased the distance between the average skin-colour likelihood in the face and image area. Afterwards, a binary image was made using a threshold which made it possible to make a clear distinction between the face and the background area. Finally, a list of face candidates was created using contour segmentation of the binary image. To classify the face candidates as faces or non-faces, four different methods were used and experimented upon. First, the face candidates which had wrong rectangular shapes and sizes were removed. This was followed by a check of the solidity of the face candidates, where candidates with a too low or too high solidity were removed from the list. Afterwards, a nose-eye template was used to remove face candidates with a too low similarity to the template. Finally, ellipse matching was done in a gradient image using either the first fitting or the best fitting ellipse around the face candidates. Candidates with a too low average gradient value in the perimeter of the ellipse were removed. Experiments were made by counting the number of false positives and false negatives and by recording the computation time of each method. First each method was experimented upon alone to find out what the threshold values should be and afterwards experiments were made on the methods in serial combination. The experiments showed that the computational fastest method was the rectangle verification and the slowest the template matching. However, by combining the methods it was both possible to lower the computation time of the template matching and decrease the number of false positives from 28.8% achieved when using only ellipse fitting to 10.7%. To lower the number of false positives even further, an experiment was made using stricter threshold values. This made the average computation time of all the methods in serial combination decrease from 105ms to 67ms and the number of false positives drop to 0%. At the same time the number of false negatives rose from 15.8% to 53.6%. However, a high number of false negatives is not that much of a problem, since the videos run at 12.5 Hz – i.e. in average it should still be possible to detect a face 6 times per second.

Several tracking methods were investigated to make it possible to track the detected faces. Based on these investigations we decided to use a combination of the Mean Shift algorithm applied to the skin-colour likelihood image and ellipse fitting in the gradient image to measure the positions and sizes of the faces. To handle the uncertainties of these measurements, a Kalman filter was used. Finally, a tracker manager algorithm which deletes the unstable, dead, and overlapping trackers and creates and updates trackers was described. Suitable parameters to use for the tracker manager, Mean Shift algorithm, ellipse fitting, and Kalman filter were found empirically and experimentally. Afterwards, experiments were performed to determine how illumination changes, occlusion, fast movement, and clutter influenced on the tracking accuracy. These showed that the trackers in general did not have any problems when clutter, occlusion, and illumination changes occurred. Only when fast movement of the faces took place, the trackers were unable to continue tracking and were eventually eliminated by the tracker manager. Finally, experiments were made to determine the ability of the tracker manager to eliminate trackers tracking non-face objects. In these it was determined that the ellipse score of trackers tracking moving non-face objects often quickly drops to a value below the scores of faces. Increasing the time before an object being tracked may be considered as a face to the maximal time an ellipse can be fitted to a non-face object, should ensure that only faces will be used to define hand-raise detection areas. However, this could mean that several seconds must pass from a person is detected till his hand-raises can be detected. The unstability measures for faces and non-faces showed the same tendency – i.e. the unstability measure for non-faces was in general higher than that of faces. During occlusions the unstability measure for faces nevertheless got higher than the unstability measure for some of the

non-face objects. Unless it is acceptable for face trackers to be deleted during an occlusion, the maximal allowable unstability will in some cases also include the non-face objects.

To be able to do hand-raise detection several hand gesture recognition methods were investigated. Based on these investigations and the fact that we only need to recognize one type of gesture (a hand-raise), it was decided to solve the problem in a somewhat different way by using changes in the skin-colour likelihood images over time as an indication of when the hand-raises occurred. Based on the position of a face tracker, a hand-raise region of interest (ROI) was defined. Within this ROI, an accumulated difference picture (ADP) was produced. The contour of the largest connected component in this ADP was classified as being caused by a hand-raise or not using one of the two Naive Bayesian Classifiers, that we implemented. This classification was, if the contour was classified as a hand-raise, followed by a verification step that examined the vertical shift in the center of mass in the previous skin-colour likelihood images. Experiments were performed to evaluate the performance of the hand-raise detection. These experiments showed that false positives could be eliminated, that is, it could be avoided that the system classified an event that was not a hand-raise as a hand-raise. However, this could only be obtained if, for the videos we used, 22.5% false negatives were accepted. Since the events that lead to false positives generally are within the control of the video conference participants, a low number of false positives will be acceptable. For 5.9% false positives (measured as a percentage of the total number of hand-raises), the corresponding number of false negatives in our videos were 8.8%.

Bibliography

[1] Staff at Intel. Intel computer vision library reference manual. `http://www.intel.com/research/mrl/research/opencv/OpenCVreferencemanual%.pdf`, jun 2000.

[2] Bernd Menser and Mathias Wien. Segmentation and tracking of facial regions in color image sequences. In *SPIE Visual Communications and Image Processing 2000*, volume 4067, pages 731–740, Perth, Australia, jun 2000.

[3] Bernhard Andersen and Erling B. Andersen. *Grundlæggende statistik*. Gyldendal, 1978.

[4] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, Santa Barbara, California, USA, June 1998.

[5] Bjarke Andersen and Peter Zinck Nielsen. Vicowijoy – investigation and implementation of methods for a video conferencing system, jan 2001.

[6] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. `http://www.andygrove.com/technology/itj/q21998/articles/art_2.htm`, 1998.

[7] Henrik I. Christensen. *Aspects of Real-Time Image Sequence Analysis*. PhD thesis, Laboratory of Image Analysis, Aalborg University, University, Fr. Bajersvej 7D, DK-9220 Aalborg East, Denmark, aug 1989.

[8] Charles Elkan. Boosting and naive bayesian learning. technical report no. cs97-557, September 1997.

[9] Enrico Di Bernardo, Luis Goncalves, and Pietro Perona. Monocular tracking of the human arm in 3d: Real-time implementation and experiments. In *Proceedings of ICPR '96*, pages 622–626. IEEE, 1996.

[10] Volker Krüger et al. Teleconferencing using an attentive camera system. In *Second International Conference on Audio- and Video-based Biometric Person Authentication*, Washington, D.C., mar 1999.

[11] G. Finlayson and G. Shaefer. Single surface colour constancy. In *7th Color Imaging Conference*, pages 106–113, Scottsdale, Arizona, November 1999.

[12] G. J. Awcock and R. Thomas. *Applied Image Processing*. McGraw-Hill Publishing Company, 1995.

[13] Greg Welch and Gary Bishop. An introduction to the kalman filter. `http://www.cs.unc.edu/~welch/kalman`, dec 1995.

[14] Allan Gut. *An Intermediate Course in Probability*. Springer-Verlag, 1995.

[15] John B. Fraleigh and Raymond A. Beauregard. *Linear Algebra*. Addison-Wesley Publishing Company, Inc., 1995.

[16] Karl Schwerdt and James L. Crowley. Robust face tracking using color, 2000.

[17] William Karush. *Politikens matematiske opslagsbog*. politikens Forlag, 1993.

[18] M. Clay Belcher and Ronald N. Helms. Lighting - the electronic textbook. `http://www.arce.ku.edu/book/contents.htm`, may 2001.

[19] Maricor Soriano, Birgitta Martinkauppi, Sami Huovinen, and Mika Laaksonen. Skin color modeling under varying illumination conditions using the skin locus for selecting training pixels. 2000. to appear in PRL.

[20] Maricor Soriano, Birgitta Martinkauppi, Sami Huovinen, and Mika Laaksonen. Skin detection in video under changing illumination conditions. In *4th Intermational Conference on Automatic Face- and Gesture-Recognition*, Grenoble, France, 2000. Machine Vision and Media Processing Unit, University of Oulu.

[21] Peter S. Maybeck. *Stochastic models, estimation, and control. Volume 1*. Academic Press, 1979.

[22] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. International Journal of Computer Vision, 1998.

[23] Michael J. Black and Allan D. Jepson. Recognizing temporal trajectories using the condensation algorithm, 1998.

[24] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 274–280, 1999.

[25] Thomas B. Moeslund and Erik Granum. Multiple cues used in model-based human motion capture. In *4th Intermational Conference on Automatic Face- and Gesture-Recognition*, pages 362–367, Grenoble, France, March 2000.

[26] Moritz Störring, Hans J. Andersen, and Erik Granum. Skin colour detection under changing lighting conditions. In *7th Symposium on Intelligent Robotics Systems*, Coimbra, Portugal, jul 1999. Laboratory of Image Analysis, Aalborg University.

[27] Moritz Störring, Hans J. Andersen, and Erik Granum. Physics-based modelling of human skin colour under mixed illuminants, jun 2000.

[28] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers.

[29] PictureTel. Picturetel concorde 4500. `http://www.picturetel.com/flash/index.html`, 2000.

[30] Polycom. Polycom viewstation 128/512k. `http://www.biz2bizonline.com/gbh/polycom_viewstation1.htm`, 2000.

[31] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunk. *Machine Vision*. McGraw-Hill, Inc., 1995.

[32] Roel Hoogenboom and Michael Lew. Face detection using local maxima, 1996.

[33] Rogrio Schmidt Feris, Tefilo Emdio de Campos, and Roberto Marcondes Cesar Junior. Detection and tracking of facial features in video sequences, apr 2000.

[34] Eli Saber and A. Murat Tekalp. Frontal-view face detection and facial feature extraction using color, shape and symmetry based cost functions. *Pattern Recognition Letters*, 1998.

[35] Shaogang Gong, Stephen J McKenna, and Alexandra Psarrou. *Dynamic Vision*. Imperial College Press, 2000.

[36] Karin Sobottka and Ioannis Pitas. Segmentation and tracking of faces in color images. In *2nd International Conference on Automatic Face- and Gesture-Recognition*, pages 236–241, Killington, Vermont, October 1996.

[37] Stephen J. McKenna, Shaogang Gong, and Yogesh Raja. Modelling facial colour and identity with gaussian mixtures. Pattern Recognition Vol. 31, No 12, 1998.

[38] Moritz Störring, Hans J. Andersen, and Erik Granum. Estimation of the illuminant colour from human skin colour. In *4th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 64–69, Grenoble, France, March 2000.

[39] Q. B. Sun, W. M. Huang, and J. K. Wu. Face detection based on color and local symmetry information. In *3rd International Conference on Automatic Face- and Gesture-Recognition*, pages 130–135, Nara, Japan, April 1998.

[40] Tat-Jen Cham and James M. Rehg. A multiple hypothesis approach to figure tracking, 1999.

[41] VTEL. Smarttrak. `http://www.vtel.com/newsinfo/resource/products/smrttrk.pdf`, 2000.

[42] Jie Yang and Alex Waibel. A real-time face tracker. In *Third IEEE Workshop on Applications of Computer Vision*, pages 142–147, Sarasota, Florida, USA, 1996.

[43] Yusuf Azoz, Lalitha Devi, and Rajeev Sharma. Reliable tracking of human arm dynamics by multiple cue integration and constraint fusion. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 905–910. IEEE, 1998.

[44] Berjamin D. Zarit, Boaz J. Super, and Francis K. H. Quek. Comparison of five color models in skin pixel classification. In *Int. Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63, Corfu, Greece, September 1999.

Appendix A

Probability Theory

This appendix will introduce some of the concepts related to probability theory that are used in the report. The reader is assumed to be familiar with basic concepts such as probability, probability densities/distributions, random variables, etc. Otherwise we recommend reading some of the literature that we used when preparing this appendix[14, 3, 21, 22, 13].

A.1 Univariate Probability Distributions

A probability distribution (or density if it is continuous[1]) may be *unimodal* or *multimodal*. A unimodal distribution has a single peak (or "mode"), i.e. a single local maximum, as illustrated in Figure A.1. A multimodal distribution is a distribution with several peaks, as illustrated in Figure A.2.

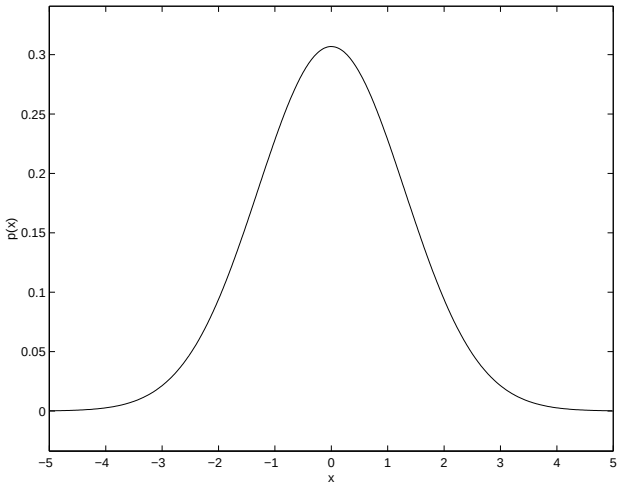


Figure A.1: Unimodal Probability Distribution. This unimodal distribution is also a normal probability distribution. The mean (and mode) of this distribution is $x = 0$ and the variance is 0.8.

The *mean* (or expected value, or first moment) of the distribution for the random variable X is given by:

[1]What is written about distributions here also applies to densities (the continuous counter-part of distributions), or is easily adapted to the continuous case.

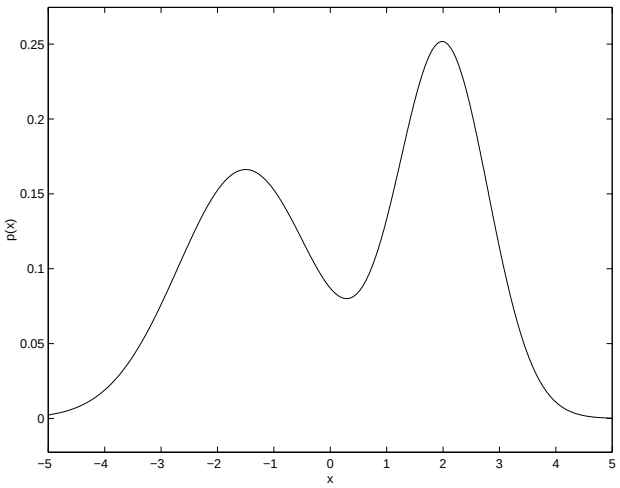


Figure A.2: Multimodal Probability Distribution.

$$EX = \sum_k x_k p_X(x_k)$$

where $p_X(x)$ is the probability that $X = x$. The *variance* (or second central moment) is given by:

$$VarX = E(X - EX)^2 = \sum_k (x_k - EX)^2 p_X(x_k)$$

The variance is a measure of how concentrated the probability "mass" – the areas under the graphs in Figures A.1-A.2 – is around the mean. A low variance means that the probabilities are high close to the mean and a high variance that they are distributed over a larger area (the total area under the graph must equal 1, as it should correspond to the probability of getting a value in the interval under the graph).

The *standard deviation* σ_X of a random variable X is given by:

$$\sigma_X = \sqrt{VarX}$$

The *covariance* of two random variables X and Y is given by:

$$Cov(X, Y) = E(X - EX)(Y - EY)$$

Variance is a special case of covariance, corresponding to the covariance of two identical variables:

$$VarX = Cov(X, X)$$

A particular kind of unimodal distribution is the normal or Gaussian distribution, which is actually what was illustrated in Figure A.1. The mean and mode of this distribution coincide, and the distribution is characterized completely by its mean and variance. That a random variable X has a normal distribution with mean μ and variance σ^2 is written:

$$p(X) \sim N(\mu, \sigma^2)$$

The probability distribution of a standard normal random variable has mean 0 and variance 1.

A.2 Multivariate Probability Distributions

A multivariate probability distribution is the probability distribution for a random vector, i.e. a vector whose components are random variables. It shows the probability of each combination of values for the variables.

A multivariate probability distribution normal if and only if every linear combination of its components is normal[2], i.e. each variable in the random vector must be normal. A multivariate normal distribution is characterized by its *mean vector* and *covariance matrix*.

Let $\mathbf{X}$ be a random vector. The *mean vector* of $\mathbf{X}$ is

$$\boldsymbol{\mu} = E\mathbf{X}$$

It has the components $\mu_i = EX_i$, $i = 1, 2, ..., n$, where X_i are the components of $\mathbf{X}$.

The *covariance matrix* of $\mathbf{X}$ is

$$\Lambda = E(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T$$

It has the components $\lambda_{ij} = E(X_i - \mu_i)(X_j - \mu_j)$, $i, j = 1, 2, ..., n$. Note that λ_{ii} (located on the diagonal) is the variance of X_i.

That a random vector $\mathbf{X}$ has a normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Λ is written:

$$p(\mathbf{X}) \sim N(\boldsymbol{\mu}, \Lambda)$$

[2] Several definitions of the multivariate normal distribution exist.

Appendix B

Estimators

In this appendix, two estimators for object tracking are presented. Familiarity with some of the basic concepts from probability theory will be needed. We refer the reader to Appendix A for an introduction to these concepts.

B.1 The Discrete Kalman Filter

B.1.1 Process and Measurement Models

The Kalman filter [13, 21, 35, 7] can be used for estimating the current state $\mathbf{x}_t \in \mathbb{R}^n$ and predicting the state at the next time step of a process governed by the linear difference equation

$$\mathbf{x}_{t+1} = A_t\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t$$

given a measurement $\mathbf{z}_t \in \mathbb{R}^m$ of the state at the current time step

$$\mathbf{z}_t = H_t\mathbf{x}_t + \mathbf{v}_t$$

$\mathbf{x}_t$ is the state at time step t and is related to the state at time step $\mathbf{x}_{t+1}$ by the $n \times n$ matrix A_t.[1]

$\mathbf{u}_t \in \mathbb{R}^l$ represents the control input to the system, which is related to the state $\mathbf{x}_{t+1}$ by the $n \times l$ matrix B. If this cannot be measured, as it is the case in this project, it may be assumed that $\mathbf{u}_t = 0$.

The $m \times n$ matrix H relates the state to what is measured.

$\mathbf{w}_t$ represents the process noise, and $\mathbf{v}_t$ the measurement noise. They are assumed to be independent of each other, white, and with normal probability distributions with covariances Q and R:

$$p(\mathbf{w}_t) \sim N(0, Q)$$

[1] The result of multiplying a matrix with a vector is a linear combination of the columns of the matrix, e.g.

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_1x_1 + a_2x_2 \\ a_3x_1 + a_4x_2 \end{bmatrix}$$

$$p(\mathbf{v}_t) \sim N(0, R)$$

Whiteness implies that the noise value is not correlated in time, and that the power of the noise is equal at all frequencies.[2]

In order to use the Kalman filter, A, B, H, Q, and R must be known.

B.1.2 Time and Measurement Update

The *a priori* estimate error covariance matrix is:

$$P_t^- = E((\mathbf{x}_t - \hat{\mathbf{x}}_t^-)(\mathbf{x}_t - \hat{\mathbf{x}}_t^-)^T) = E\begin{bmatrix} (\Delta x_1)^2 & \Delta x_1 \Delta x_2 & \cdots & \Delta x_1 \Delta x_n \\ \Delta x_1 \Delta x_2 & (\Delta x_2)^2 & \cdots & \Delta x_2 \Delta x_n \\ \vdots & \vdots & \ddots & \vdots \\ \Delta x_1 \Delta x_n & \Delta x_2 \Delta x_n & \cdots & (\Delta x_n)^2 \end{bmatrix}$$

where $\Delta x_i = x_i - \hat{x}_i^-$ and x_i, $\hat{x}_i^-$, $i = 1, 2, ..., n$ are the components of the vectors.

Similarly, the *a posteriori* estimate error covariance matrix is:

$$P_t = E((\mathbf{x}_t - \hat{\mathbf{x}}_t)(\mathbf{x}_t - \hat{\mathbf{x}}_t)^T) \tag{B.1}$$

The Kalman filter maintains estimates of the mean and error covariance of the state probability distribution. This is done recursively using two sets of equations:

The *time update* equations use the *a posteriori* state estimate to compute the *a priori* state and covariance estimates for the next time step (i.e. a prediction):

$$\hat{\mathbf{x}}_{t+1} = A_t \hat{\mathbf{x}}_t + B\mathbf{u}_t$$

$$P_{t+1}^- = A_t P_t A_t^T + Q_t \tag{B.2}$$

The *measurement update* equations use the measurement to compute the *a posteriori* state and covariance estimates for the current time step:

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \tag{B.3}$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + K_t(\mathbf{z}_t - H_t \hat{\mathbf{x}}_t^-) \tag{B.4}$$

$$P_t = (I - K_t H_t) P_t^-$$

[2] That the power of the noise is equal at all frequencies implies that white noise has infinite power, so it cannot exist. However, because of the limited bandwidth of real systems, they will not be able to distinguish white noise from a noise which "looks like" white noise within the range of frequencies that the system responds to. If the noise is not of equal power within this range of frequencies, or if it is correlated in time, a so-called "shaping filter" may be added to the model of the system, which converts white noise to the particular kind of non-white noise – then the system can be assumed to be driven by white noise.

K_t is the Kalman gain. It determines to which degree the measurement affects the *a posteriori* estimates, as it can be seen from Equation B.4, where the prediction error (in measurement space) is weighted by K_t and added to the *a priori* state estimate. Equation B.3 has been chosen such that the *a posteriori* error covariance P_t (Equation B.1) is minimized.

B.1.3 Kalman Filter Order

The order of a Kalman filter denotes the number of derivatives of the parameters being tracked (e.g. the position) that is included in the state vector.

A zeroth order Kalman filter includes no derivatives and uses the identity matrix[3] as A, i.e. the mean of the *a posteriori* estimate for time step $t-1$ becomes the mean of the *a priori* estimate for time step t (assuming that the control input $\mathbf{u}_t$ is zero). The error covariance is updated as indicated by Equation B.2 to reflect the increased uncertainty.

First order Kalman filters use state vectors of the form:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_1' \\ \vdots \\ x_n \\ x_n' \end{bmatrix}$$

A is a $n \times n$ block-diagonal matrix:

$$A = \begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D \end{bmatrix}$$

where

$$D = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

The effect of multiplying A with a state vector x containing positions and their derivatives (velocities) is that the product of the velocity and the time passed since the positions was estimated is added to each position. This will correspond to the new positions if the object moves with constant velocity and the velocities stored in x are correct.

A second order Kalman filter includes the second derivative (acceleration) as well, and uses

$$D = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

[3] In an identity matrix, all components are 0, except the ones on the diagonal from upper left corner to lower right corner, which are all 1.

B.1.4 The Extended Kalman Filter

If it is impossible to model the process using the linear difference equation given above, or the relationship between the measurement and the actual state of the process is non-linear, an *extended Kalman filter* (EKF) may be used. The EKF can be used for estimating and predicting the state of a process governed by the non-linear difference equation

$$\tilde{\mathbf{x}}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)$$

given measurements that are related to the state by

$$\tilde{\mathbf{z}}_t = h(\mathbf{x}_t, \mathbf{v}_t)$$

The functions f and h must be known in advance. At each time step, at set of matrices is computed from these functions. These matrices are used in time and measurement update equations very similar to those of the (linear) Kalman filter to obtain *a priori* and *a posteriori* estimates. For further information about the EKF, see [13].

B.2 CONDENSATION

The CONDENSATION algorithm [22] is designed to track curves in visual clutter. It is, however, sufficiently general to be applicable to other things than curves. CONDENSATION works by propagating a conditional density over time using a technique known as "factored sampling". The conditional densities are represented as sample sets. This means, that unlike the Kalman filter (and the EKF), CONDENSATION is able to handle multimodal probability distributions. Furthermore, it does not assume that the process can be modeled with a linear differential equation like the Kalman filter, and does not make assumptions about the statistical nature of the noise (the Kalman filter and the EKF assumes normally distributed noise).

B.2.1 Assumptions

Let $X_t = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$ be the history of state vectors $\mathbf{x}_i$. It is assumed that they form a temporal Markov chain, i.e.:

$$p(\mathbf{x}_t \mid X_{t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

I.e. information about the state at the previous time step provides as much relevant information as information about all previous states.

Let $Z_t = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_t\}$ be the history of measurements $\mathbf{z}_i$. It is assumed that:

$$p(Z_{t-1}, \mathbf{x}_t \mid X_{t-1}) = p(\mathbf{x}_t \mid X_{t-1}) p(Z_{t-1} \mid X_{t-1}) = p(\mathbf{x}_t \mid X_{t-1}) \prod_{i=1}^{t-1} p(\mathbf{z}_i \mid \mathbf{x}_i)$$

This means that the measurements $\mathbf{z}_i$ must be mutually independent (implied by the product of the probabilities $p(\mathbf{z}_i \mid \mathbf{x}_i)$), and that they must be independent of the process.

B.2.2 The CONDENSATION Algorithm

Let the sample set at time step $t-1$ be

$$S_{t-1} = \{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}) \mid n = 1, 2, ..., N\}$$

and let the sample set at time step t be

$$S_t = \{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}) \mid n = 1, 2, ..., N\}$$

The sample sets have a constant size N.[4] $\mathbf{s}_t^{(n)}$ is a state vector. $\pi_t^{(n)}$ is the probability of the state vector, and $c_t^{(n)}$ the cumulative probability:

$$c_t^{(n)} = \sum_{i=1}^{n} \pi_t^{(i)}$$

The samples of S_t are computed as follows:

1. Select samples $\mathbf{s'}_t^{(n)}$, $n = 1, 2, ..., N$ from S_{t-1}.
2. Predict new samples $\mathbf{s}_t^{(n)}$, $n = 1, 2, ..., N$ by sampling $p(\mathbf{x}_t \mid \mathbf{x}_{t-1} = \mathbf{s'}_t^{(n)})$.
3. Weight each sample $\mathbf{s}_t^{(n)}$, $n = 1, 2, ..., N$ according to the probability of the measurement $\mathbf{z}_t$ given $\mathbf{s}_t^{(n)}$.

The initial sample set S_0 can, for instance, be obtained by sampling a Gaussian function which has the state vector of the detected object as its mean. This reduces the problem of obtaining S_0 to the problem of constructing a state vector for the object from the information provided by the previous phases of the system (e.g. position, but not velocity and acceleration).

B.2.3 Selection of a Sample

The sample $\mathbf{s'}_t^{(n)}$ is selected from S_{t-1} using the following procedure:

1. Generate a random number $r \in [0, 1]$. The numbers should be uniformly distributed.
2. Find the smallest j for which $c_{t-1}^{(j)} \geq r$.
3. Let $\mathbf{s'}_t^{(n)} = \mathbf{s}_{t-1}^{(j)}$.

The procedure chooses $\mathbf{s}_{t-1}^{(j)}$ with the probability $\pi_{t-1}^{(n)}$. It may result in the the same sample from S_{t-1} being chosen several times, especially if the sample has a high weight. Some samples may not be chosen at all, especially those with low weights.

[4] A constant size – or at least an upper limit on the size – of the sample set is required for real-time operation.

B.2.4 Prediction of New Sample

The new sample $\mathbf{s}_t^{(n)}$ in S_t is, in principle, computed by sampling:

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1} = \mathbf{s'}_t^{(n)})$$

How this is done depends on how the process dynamics is modeled (this will be discussed below). The result can be divided into a deterministic drift, which displaces $\mathbf{s'}_t^{(n)}$ in state space, and a diffusion step, which displaces the sample randomly according to the stochastic component of the process model. If the same sample in S_{t-1} was chosen several times for S_t, the identical copies of this sample will become non-identical in S_t due to the diffusion.

B.2.5 Weighting of the New Sample

The probability weight for the new sample given the new measurement $\mathbf{z}_t$ is computed as follows:

$$\pi_t^{(n)} = kp(\mathbf{z}_t \mid \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

I.e. $\pi_t^{(n)}$ is a measure of how well the sample $\mathbf{s}_t^{(n)}$ explains the current measurement $\mathbf{z}_t$.

k is a normalization constant chosen such that:

$$\sum_{n=1}^{N} \pi_t^{(n)} = 1$$

Since k cannot be computed before all probabilities $p(\mathbf{z}_t \mid \mathbf{x}_{t-1} = \mathbf{s'}_t^{(j)})$ have been computed, the normalization must be postponed until all samples have been processed.

The cumulative probability is computed as follows:

$$c_t^{(0)} = 0$$
$$c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)}$$

where $n = 1, 2, ..., N$.

B.2.6 Process Model

To use the CONDENSATION algorithm, models describing the process dynamics and the measurement process must be established.

The *process model* must specify $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. If the process can be modeled as a linear difference equation, the prediction of the new sample value can be done as follows:

$$\mathbf{s}_t^{(n)} = A\mathbf{s'}_t^{(n)} + \mathbf{w}_t^{(n)}$$

A is a $N \times N$ matrix, and is responsible for the deterministic drift. $\mathbf{w}_t^{(n)}$ is a random vector with a covariance matrix that reflects the process noise. It is responsible for the

diffusion that ensures that identical samples from S_{t-1} get different values in the new sample set S_t.

The state vectors and A may have the form presented in the section about the order of Kalman filters on page 131. The number of derivatives of the parameters being tracked that are included in the state vectors will be referred to as the order of the CONDENSATION based tracker.

B.2.7 Measurement Model

The *measurement model* must specify $p(\mathbf{z}_t \mid \mathbf{x}_t)$, or, if the measurement process is assumed to be stationary in time, $p(\mathbf{z} \mid \mathbf{x})$.

Let

$$\mathbf{z} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_M \end{bmatrix}$$

where $\mathbf{m}_1, ..., \mathbf{m}_M$ are the measurements for a particular time step (they can be scalars, or, as indicated by the bold font, vectors).

$\mathbf{z}$ contains more than one measurement because of clutter. For instance, when tracking a face, the presence of other faces than the one being tracked may cause $\mathbf{z}$ to have several components, because all faces have the same basic characteristics (e.g. colour, size, movement). It may be assumed that only one of the measurements correspond to the object being tracked.[5] Then the event ϕ_m that $\mathbf{m}_m$ is the measurement corresponding to the object is true with the probability $P(\phi_m)$ and the event $P(\phi_{none})$ that none of the measurements correspond to the object is true with the probability:

$$P(\phi_{none}) = 1 - \sum_{m=1}^{M} P(\phi_m)$$

Then

$$p(\mathbf{z} \mid \mathbf{x}) = p(\mathbf{z} \mid \phi_{none})P(\phi_{none}) + \sum_{m=1}^{M} p(\mathbf{z} \mid \mathbf{x}, \phi_m)P(\phi_m)$$

where the first term reflects that the contents of the measurement vector $\mathbf{z}$ could be due to clutter rather than the object being tracked, and the second term reflects the possibility that the measurements would arise given $\mathbf{x}$.

The conditional probability distribution given by $p(\mathbf{z} \mid \mathbf{x})$ will have peaks corresponding to the different measurements $\mathbf{m}_m$ and a "background" probability reflecting the possibility that none of the measurements correspond to the object.

If it is assumed that the measurement corresponding to the object is normally distributed with a mean corresponding to the state vector (mapped from state space to measurement

[5] Obviously, this is not always true. If, for instance, the measurements are regions found in the image using colour segmentation, an object may split into two objects due to occlusion or changing illumination.

space), $p(\mathbf{z} \mid \mathbf{x})$ may be approximated by setting $P(\phi_m) = 0$ for measurements that are too far away from $\mathbf{x}$, i.e. ignoring measurements outside a search window around $\mathbf{x}$. The size of the necessary search window will depend on the covariance matrix of the measurement normal distribution.

Appendix C

The Skin-Colour Model

To define the skin chromaticity distributions we have used a *skin-colour model* reported by Störring *et al.* in [27]. Using this model we have calculated a set of Gaussian models $N(\mathbf{\Sigma}^2, \mathbf{m})$ for r chromaticities between 0.2 and 0.8. The area in chromaticity plane covered by these models can handle illuminations of correlated colour temperatures (CCTs – we refer the reader to the beginning of Section 2.5.1 on page 14 for a description of CCTs) from 1750 Kelvin (K) to 15000K. This should be enough to cover the most common every day light sources (see Figure C.1). The model was white balanced using a canonical illuminant with a CCT of 3600K. The centers of mass of the Gaussian models make out the *skin locus* in Figure C.1, and as it can be seen this locus has close resemblance to the *Planckian locus*. The Planckian locus is the locus along which the light colours of different Blackbody radiators lie. According to [11] all other kinds of everyday light sources lie close to this locus. Therefore, the Planckian locus of Blackbody radiators should be usable for approximating general purpose light sources. The dotted lines in Figure C.1 indicate the relation between the CCT of the light source and the corresponding skin chromaticity distribution center of mass. The distance of this relation gets longer the lower the CCT gets. This is the reason why the skin chromaticity distribution changes in shape and size when the CCT changes. We will discuss how the size and shape of the skin chromaticity distribution can be constrained in the following sections.

C.1 Model Constraints

Our main purpose of using the skin-colour model, is to be able to constrain the moments of the LUTs or Gaussian models (see Section 2.5.4 on page 17) estimated by the VICOW-IJOY system at run-time. We know that the skin-colours should be found close to the skin locus and can use this to constrain the position of the center of mass of the LUTs or the Gaussian models.

In Figure C.2, the minor/major *eigenvalue* ratio with increasing r chromaticities is illustrated. The eigenvalues are calculated from the covariance matrix $\mathbf{\Sigma}^2$ and indicate the variances along the two directions of highest variance[1]. As it can be seen, the relative size of the minor eigenvalue compared to the major gets smaller and smaller with increasing chromaticity r values. I.e. the shape of the skin chromaticity distribution starts out close to circular and then it gets more and more flattened. (see Figure C.3). Having this information it is possible to constrain the shape of the skin chromaticity distribution

[1] We refer the reader to [15] for more information about eigenvalues

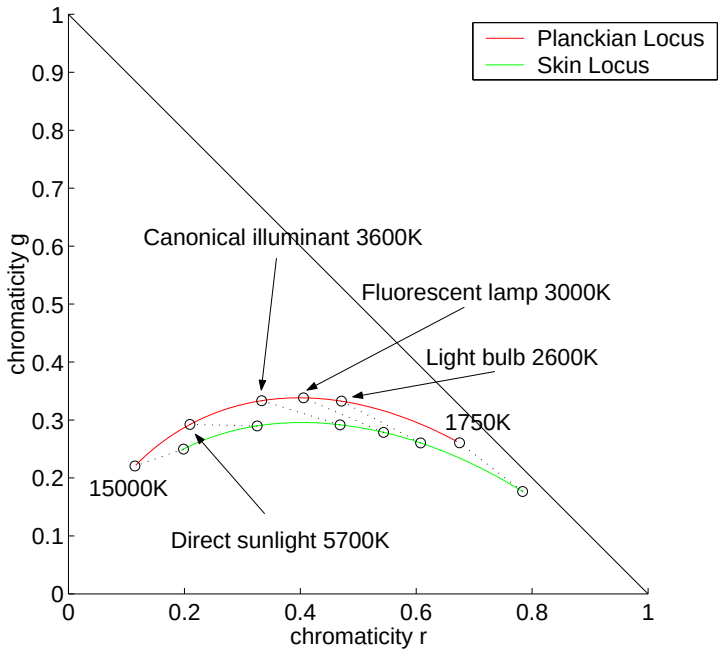


Figure C.1: The Skin Locus: The skin chromaticity distribution moves along a locus similar to the Planckian locus of Blackbody radiators when the CCT changes. The dotted lines indicate the relation between the CCT of an illumination colour, and the corresponding skin chromaticity distribution center of mass.

in either the LUTs or the Gaussian models estimated in the VICOWIJOY system. This can be done by using minimum and maximum borders for the minor/major eigenvalue ratio. This is also illustrated in Figure C.2.

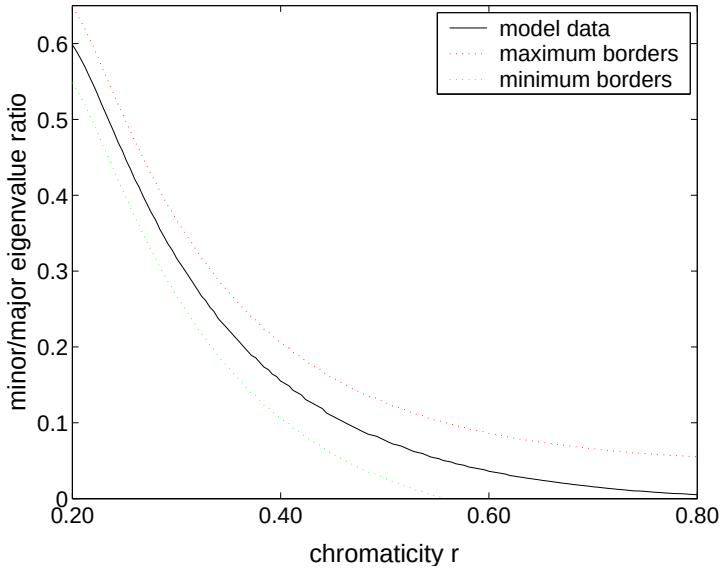


Figure C.2: Shape Constraints: The figure illustrates the minor/major eigenvalue ratio. Minimum and maximum borders can be used to constrain the shape of the skin chromaticity distribution in either LUTs or Gaussian models.

In Figure C.4 the normalized area sizes of the skin chromaticity distributions along the chromaticity r axis are illustrated. These areas have been calculated by using the equation for the area of an ellipse, i.e.

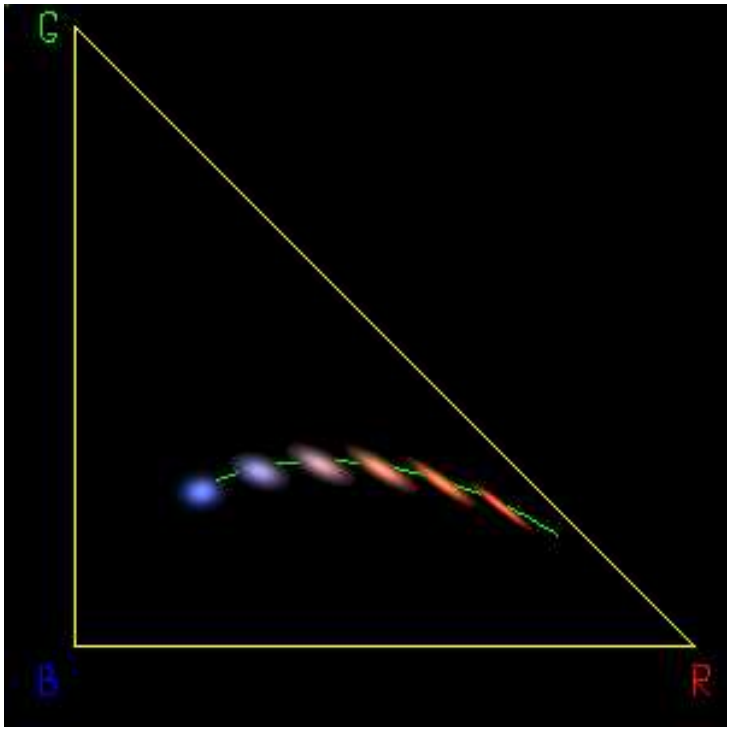


Figure C.3: Skin Chromaticity Distribution Shape Change: The shape of the skin chromaticity distribution goes from being close to circular at low r-values to a flattened ellipse at high r-values.

$$area = \pi AB$$

where A and B are the major and minor axis of the ellipse. Since the eigenvalues correspond to the variances of the distribution, we use the square root of them as the radii of the ellipse. I.e.

$$A = \sqrt{major\ eigenvalue}$$

and

$$B = \sqrt{minor\ eigenvalue}$$

The curve in the figure peaks at $r = 0.357$. Moving in either direction along the chromaticity r axis from this point the area gets smaller. Again, we should be able to use this knowledge to make minimum and maximum borders for the area size.

Looking at Figure C.3 it can be seen that the skin chromaticity distribution seems to rotate clockwise, when it moves to the right along the chromaticity r axis. In Figure C.5, we have illustrated the degrees of clockwise rotation of the major eigenvalue in relation to the chromaticity r axis. As it can be seen, the skin chromaticity distribution does indeed rotate clockwise when r increases. Using this knowledge, we can verify that the rotation of the skin chromaticity distribution in a LUT or a Gaussian model is inside minimum and maximum borders.

Using the skin-colour model should therefore enable us to control the position, shape, size, and rotation of the skin chromaticity distribution. The minimum and maximum borders must nevertheless be rather non-restrictive, since awkward illuminations (such

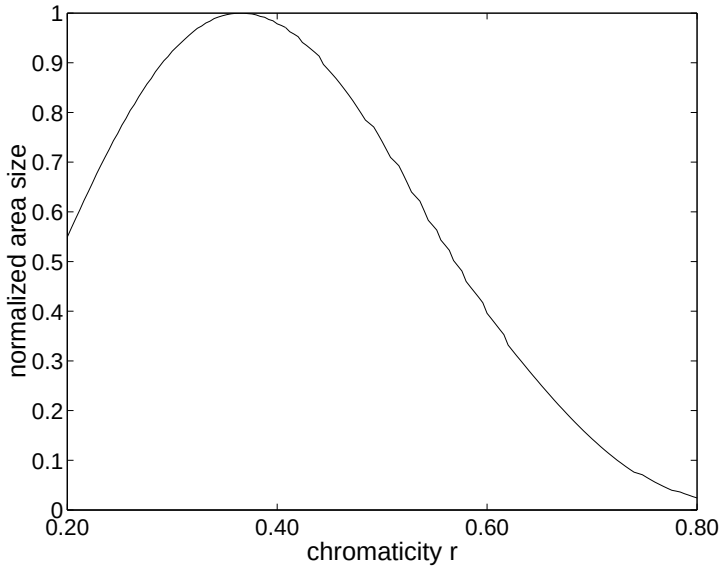


Figure C.4: Area Size Constraints: The area size of the skin chromaticity distribution is largest when chromaticity $r = 0.357$. Moving in either direction from this point, the area gets smaller. This information can be used to define a minimum and maximum size of the skin chromaticity distribution, according to where a LUT or Gaussian model has its chromaticity r center of mass.

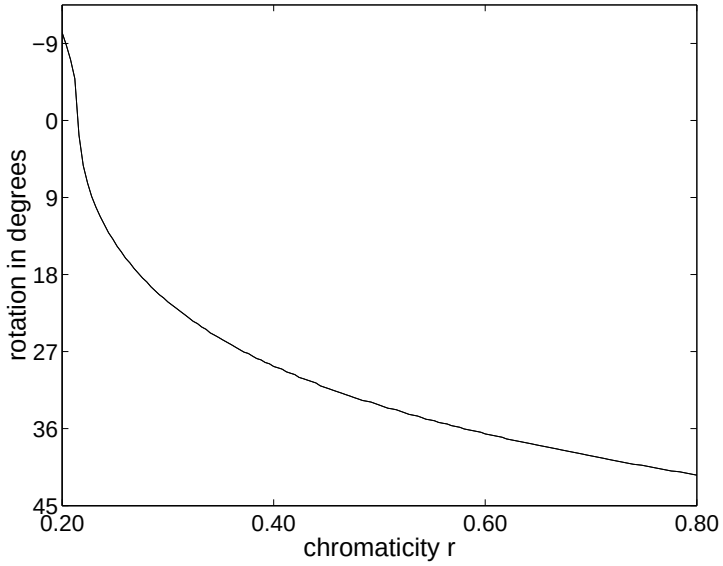


Figure C.5: Area Rotation Constraints: When moving to the right along the chromaticity r axis, the skin chromaticity distribution rotates clockwise.

as when several light sources of different CCTs are focused on the same face) do exist. Using the borders as guidelines should nevertheless be a good idea. In Section 2.5.4 on page 17 the use of moment constraints is discussed further.

C.2 Model Verification

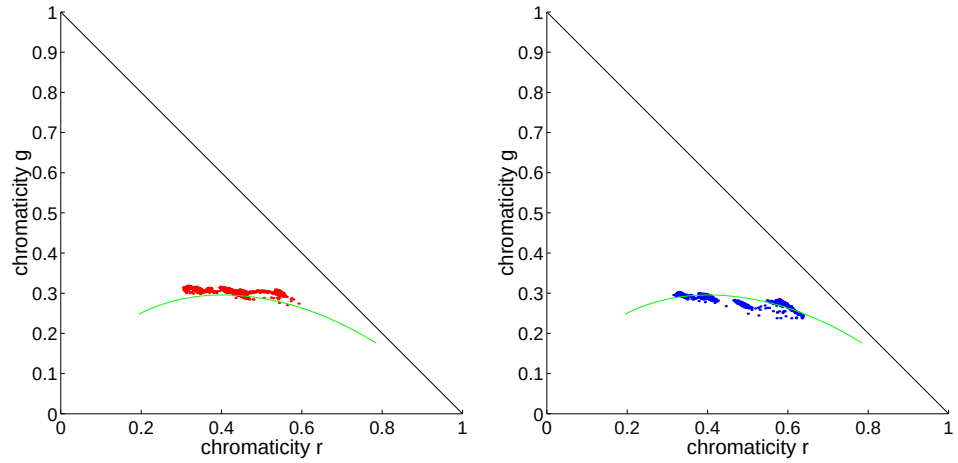
To verify the skin-colour model, we have used the videos V8 and V10 which are described in Appendix D. In both V8 and V10 the person does not move his face very much, and therefore it was possible to define an area at a fixed position to use for the whole

video. For each image in V8 and V10 the pixels in the defined area were converted to chromaticity plane and used to generate a Gaussian model. The parameters $(\mathbf{\Sigma}^2, \mathbf{m})$ of these models were saved to a file and afterwards compared to the results of the skin-colour model. In both V8 and V10, the CCT is changed artificially by rotation of an arrangement of fluorescent lamps. This makes sure that a rather large area along the chromaticity r axis is covered (see Figure C.6).

C.2.1 Center of Mass

In Figure C.6(a) the centers of mass of the Gaussian models are plotted. As it can be seen, they are rather close to the skin locus, where V8 is slightly below and V10 slightly above. Using the skin locus defined by the skin-colour model to constrain the position of the centers of mass of LUTs and Gaussian models should therefore be a good idea.

Looking at the figures it may also be notified, that the plotted values from V8 and V10 group in 4 places. This can be seen more clearly if we instead look at the frequencies of the chromaticity r centers of mass, which are illustrated in Figure C.7). The 4 groups correspond with the number of different light sources that were used when recording the videos.



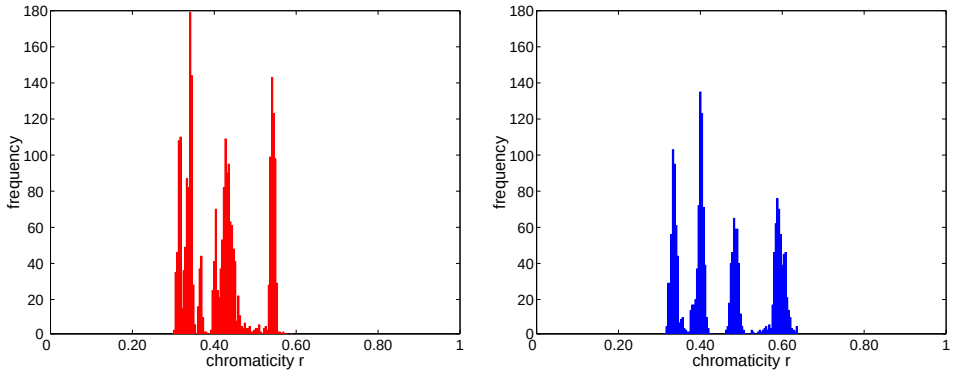
(a) Samples from V8

(b) Samples from V10

Figure C.6: Skin Locus Verification: The sampled centers of mass are slightly above the skin locus in V8 (a) and slightly below in V10 (b).

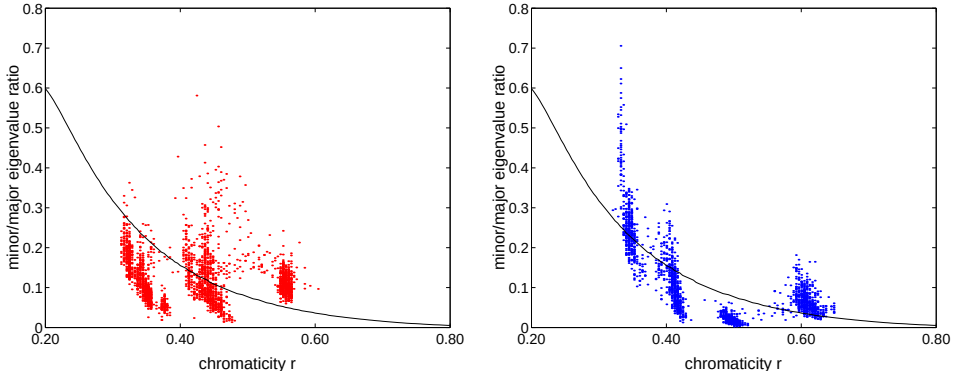
C.2.2 Minor/Major Eigenvalue Ratio

In Figure C.8 we have plotted the minor/major eigenvalue ratios in V8 and V10. V10 can to some extend be said to follow the model, but V8 seems to have more or less the same ratios no matter what the chromaticity r value is. Constraining the minor/major eigenvalue ratio based on the chromaticity r value is therefore not a good idea. It is better to make a soft constraint that demands that the variance along the chromaticity r axis always must be larger than the variance along the chromaticity g axis. I.e. since we in almost all cases have clockwise rotation between 0° and 45° (refer to Section C.2.4), the variance along the chromaticity r axis is bound to be larger than the variation along the chromaticity g axis.



(a) Frequency for V8 (b) Frequency for V10

Figure C.7: Chromaticity r Center of Mass Frequency: The chromaticity r centers of mass group in 4 places which corresponds to the number of illuminations that were used when recording the videos.

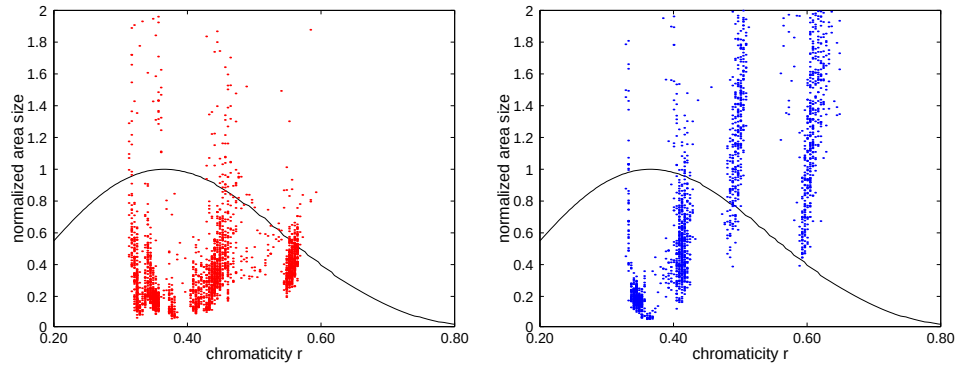


(a) Samples from V8 (b) Samples from V10

Figure C.8: Minor/Major Eigenvalue Ratio Verification: The samples of V8 (a) show that the minor/major eigenvalue ratio more or less is the same all the time. For V10 (b) the ratio has a tendency of following the skin-colour model curve. Constraining the minor/major eigenvalue ratio might not be a good idea. It would probably be better with a softer constraint that demands that the variance along the chromaticity r axis always must be larger than the variance along the chromaticity g axis.

C.2.3 Skin Chromaticity Distribution Area Size

Constraining the area size of the skin chromaticity distribution might also be a possibility. In Figure C.9 the sampled area sizes are shown. As it can be seen, the results of both V8 and V10 do not evolve in the same way as the skin-colour model curve. However, the sizes of the areas in V8 and V10 are almost identical at different chromaticity r values (i.e. they go from about 25% to twice the size of the maximum area size in the skin-colour model). Therefore, it might be reasonable to make overall constraints about the area size. This can be done by making overall constraints about the minimum and maximum size of the variances along the chromaticity r and g axis. Using the amount of chromaticity r as an indicator of these maximum and minimum sizes is nevertheless not a good idea.

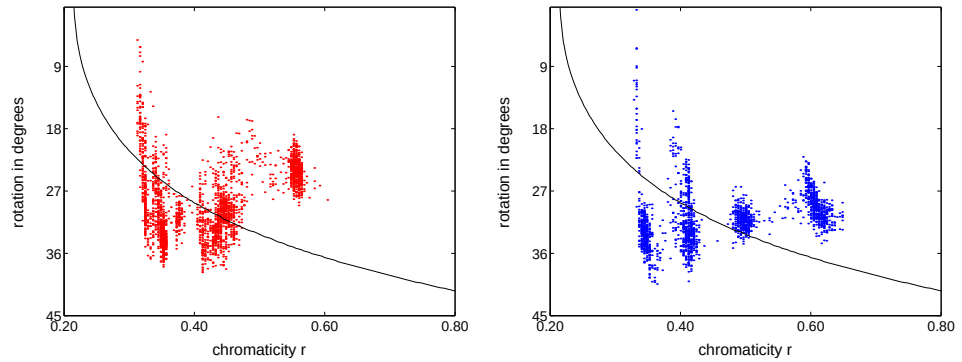


(a) Samples from V8 (b) Samples from V10

Figure C.9: Area Size Verification: The data plotted from V8 (a) and V10 (b) do not evolve in the same way as the skin-colour model curve. Using overall borders for the variances along the chromaticity r and g axis to constrain the area size might be better than using chromaticity r dependent borders.

C.2.4 Skin Chromaticity Distribution Rotation

The last characteristic of the skin-colour model we want to verify, is the use of constraints on the rotation of the major eigenvalue of skin chromaticity distribution. In Figure C.10 the data made by V8 and V10 can be seen. Again, the data does not evolve in the same way as the skin-colour model along the chromaticity r axis. However, it may be noted that the rotation always seems to be clockwise in relation to the chromaticity r axis. Therefore an overall constraint could be to ensure that rotation always is clockwise in relation to the chromaticity r axis.



(a) Samples from V8 (b) Samples from V10

Figure C.10: Rotation Verification: The sampled data from V8 (a) and V10 (b) do not evolve in the same way as the skin-colour model along the chromaticity r axis. Not allowing the skin chromaticity distribution to rotate counter-clockwise in relation to the chromaticity r axis could be used as an overall constraint.

C.3 Skin-Colour Model Conclusions

The verification of the skin-colour model has shown a close resemblance with data acquired from two video sequences, V8 and V10, when it comes to the centers of mass of the Gaussian models. Therefore, the chromaticity r center of mass should be possible to use as an indicator of where the chromaticity g center of mass should be. Regarding the evolvement of the relative size of the eigenvalues along the chromaticity r axis, only the results of V8 could to some extend be said to follow the skin-colour model. It would probably be better to use a softer constraint about that the variance along the chromaticity r axis always must be larger than the variance along the chromaticity g axis. When verifying the skin chromaticity distribution area size, the data from V8 and V10 did again not follow the evolvement of the skin model along the chromaticity r axis. It was nevertheless noted, that the area sizes of V8 and V10 always was close to the area size of the skin-colour model. This can be used to make overall limitations of the area size by constraining the variances along the chromaticity r and g axis. Finally, the rotation of the major eigenvalue in relation to the chromaticity r axis was verified and again the results from V8 and V10 did not follow the skin-colour model curve. The rotation was nevertheless always clockwise, which can be used to constrain the clockwise rotation to always be above or equal to 0°.

Appendix D

Video Collections

To be able to do experiments in this project, we have made a number of video sequences, where people do hand-raises in many different ways. In some of the videos the CCT (the illumination colour) is changed either artificially by using fluorescent lamps of different CCTs or more naturally by pulling curtains back to let in sunlight through a window. The videos were recorded with 25 Hz using a Sony DSR-PD150P digital camcorder which have a 1/3 type CCD with a resolution of approximately 400000 pixels. In all the videos the camera was white balanced at a CCT of 3680K, and afterwards the auto white balance function was turned off. The videos were afterwards transfered to the program iMovie on an iMac via a firewire connection. The videos was then saved as QuickTime movies using Indeo5 compression. Thereafter, they was converted to AVI-files using a small program called Movie Translator. Finally, we downscaled the videos to 320×240 pixels using the program VirtualDub.

The use of the Indeo5 compression changed the movement made from image to image in the videos. Therefore, almost no movement is detectable in every second image. We have not investigated the technical details behind this "feature", but a few tests have shown that the system runs fine when using only every second image of the videos. Therefore the videos are said to be recorded at a framerate of 12.5 Hz throughout the report.

In the following sections we will describe the scenario for each of the videos.

D.1 Videos with Constant CCTs

We have made 7 video sequences using a controlled CCT of 3680K. This illumination colour is created by having two pairs of vertical fluorescent lamps in front of the scene. All other indoor lights are turned off and dark curtains are drawn in front of the windows to block out the outdoor light. The following list describes each of the video sequences. Example images taken from the videos can be seen in Figure D.1.

V1 Two persons making correct hand-raises.

V2 Two persons making incorrect hand-raises.

V3 Two persons making correct hand-raises. A third person walks back an forth in the background.

V4 Two persons making making a few hand-raises. A third person walks back and forth in the foreground.

V5 Two persons walks into the scene and sits down and do a couple of correct hand-raises. The one person then leaves the scene for a short while and the comes back. Both persons then do a couple of hand-raises before they both leave the scene.

V6 Two persons do correct hand-raises. A third person enters the scene in the background, writes something on the black border, and leaves the scene again.

V7 Two bare-armed persons making correct hand-raises.

(a) Image Example from V1

(b) Image Example from V2

(c) Image Example from V2

(d) Image Example from V3

(e) Image Example from V3

(f) Image Example from V4

(g) Image Example from V4

(h) Image Example from V5

(i) Image Example from V5

(j) Image Example from V6

(k) Image Example from V6

(l) Image Example from V7

Figure D.1: Image Examples from Videos with Fixed CCT: Figures (a) - (l) illustrates image examples from the videos V1-V7.

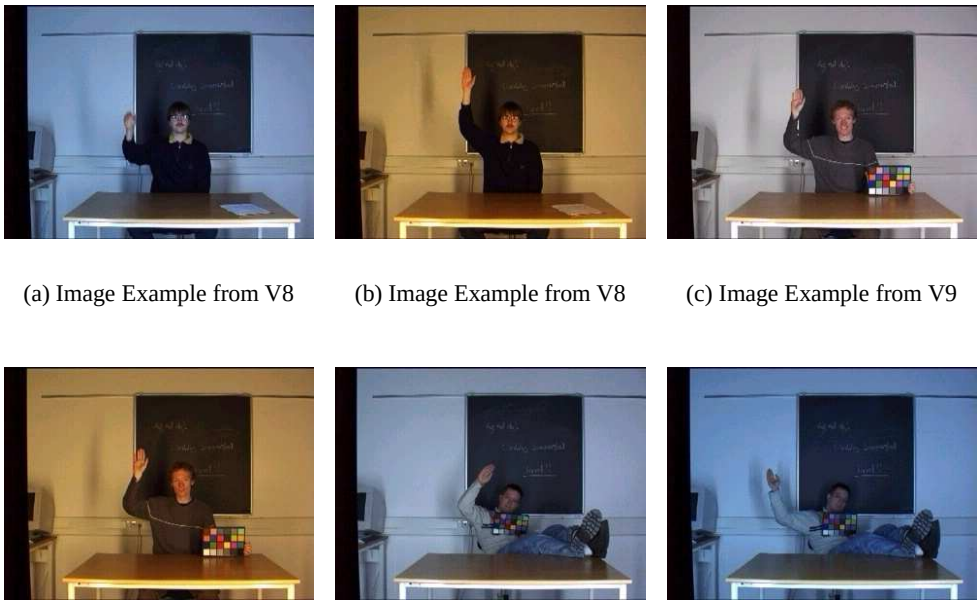
D.2 Videos with Fast Changes in CCT

To be able to do experiments on VICOWIJOYs capability of adjusting to rather fast changes in CCT, we have made 3 videos, where the CCT is changed artificially about every 15th second. This is done by turning two vertical arrangements of fluorescent lamps. Each arrangement holds 4 pairs of fluorescent lamps with CCTs of 2600K, 3680K, 4700K, and 6200K. The fluorescent lamps are turned in the same direction such that we do not get a mix of illumination colours. All other lights are turned off and outdoor light coming through the windows is blocked out by dark curtains. In the list below each of the videos are described. Example images from the videos can be seen in Figure D.2.

V8 One person making correct hand-raises.

V9 Another person making correct hand-raises.

V10 A third (quite relaxed) person making correct hand-raises.



(a) Image Example from V8 (b) Image Example from V8 (c) Image Example from V9

(d) Image Example from V9 (e) Image Example from V10 (f) Image Example from V10

Figure D.2: Image Examples from Videos with Fixed CCT: Figures (a) - (f) illustrates image examples from the videos V8-V10.

D.3 Videos with Constant CCTs and Uniform Background

In these 6 videos we have used a very controlled environment. The recordings were made in a closed room with no windows and with a grey screen as background. The CCT is 3680K in all the videos and is made by having two pairs of vertical fluorescent lamps in front of the scene. Each of the scenarios of the videos are described in the following list. Example images from the videos can be seen in Figure D.3.

V11 Two persons making correct hand-raises.

V12 Two persons making few correct hand-raises and many incorrect hand-raises. One of the persons demonstrates his crawling talent.

V13 Two persons making correct hand-raises. The first one gets up, walks back and forth in the background and then sits back down. Inspired by this, the second person does the same.

V14 Two persons making making a few correct hand-raises. Only very small face rotations are made.

V15 Two persons making a few correct hand-raises. Faces are rotated without limits.

V16 Two persons making mostly correct hand-raises.

D.4 Videos with Mixed Illumination Colours

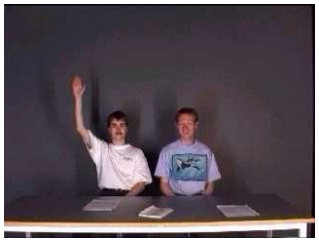
The four last videos we have made have CCTs made of mixed illumination colours. In V17 the only lights we have turned on are the fluorescent lamps in the ceiling. In V18 we combine this with artificial sunlight, which is made by an incandescent lamp which is slowly increased and decreased in intensity. In V19 we draw back the curtains to mix sunlight into the video. Finally, in V20 we do the same but have the fluorescent lamps in the ceiling turned off. What happens in the four videos is described in the list below. In Figure D.4 example images from the videos can be seen.

V17 Two persons making both correct and incorrect hand-raises.

V18 Two persons making correct hand-raises.

V19 Two persons making both correct and incorrect hand-raises. A third person pulls back the curtains to let in the sunlight. He then walks a bit around in the background before he finally draws the curtains in front of the windows again.

V20 Two persons making both correct and incorrect hand-raises. A third person pulls back the curtains and sunlight enters the scene. He then leaves then scene for a while, comes back, and pulls back the curtains. Finally he leaves the scene in the background.



(a) Image Example from V11

(b) Image Example from V11

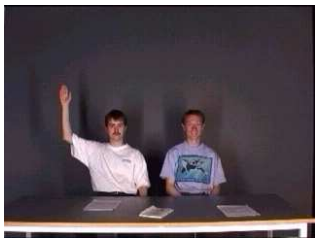
(c) Image Example from V12

(d) Image Example from V12

(e) Image Example from V13

(f) Image Example from V13

(g) Image Example from V14



(h) Image Example from V14

(i) Image Example from V15

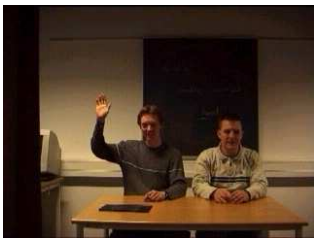
(j) Image Example from V15

(k) Image Example from V16

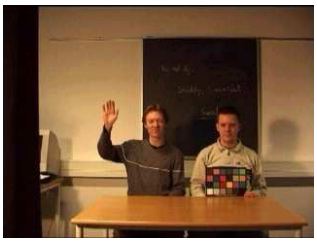
(l) Image Example from V16

Figure D.3: Image Examples from Videos with Fixed CCT: Figures (a) - (l) illustrates image examples from the videos V11-V16.

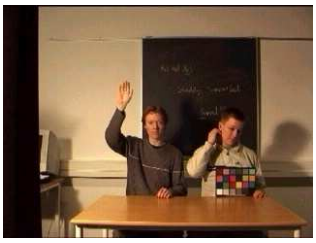
(a) Image Example from V17



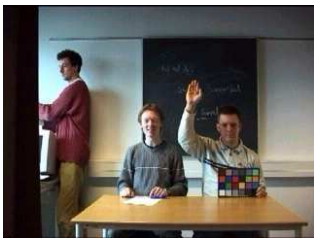
(b) Image Example from V17



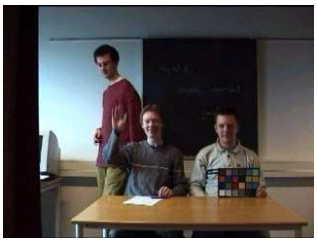
(c) Image Example from V18



(d) Image Example from V18



(e) Image Example from V19



(f) Image Example from V19

(g) Image Example from V20

(h) Image Example from V20

(i) Image Example from V20

Figure D.4: Image Examples from Videos with mixed: Figures (a) - (i) illustrates image examples from the videos V17-V20.