Neural Network Gaze Tracking using Web Camera

David Bäck

2005-12-16

LiTH-IMT/MI20-EX--05/414--SE

Neural Network Gaze Tracking using Web Camera

Examensarbete utfört i bildbehandling
vid Tekniska högskolan i Linköping
av

David Bäck

LiTH-IMT/MI20-EX--05/414--SE

Handledare: **Joakim Rydell**
CMIV & IMT, Linköpings universitet

Examinator: **Magnus Borga**
IMT, Linköpings universitet

Linköpings tekniska högskola
Institutionen för medicinsk teknik

Rapportnr: LiTH-IMT/MI20-EX--05/414--SE

Datum: 2005-12-16

Svensk titel	Neuronnätsbaserad Blickriktningsdetektering genom användande av Webbkamera
Engelsk titel	Neural Network Gaze Tracking using Web Camera
Författare	David Bäck

Uppdragsgivare:	**Rapporttyp:**	**Rapportspråk:**
CMIV/IMT	Examensarbete	**Engelska/English**

Sammanfattning (högst 150 ord).
Abstract (150 words)

Gaze tracking means to detect and follow the direction in which a person looks. This can be used in for instance human-computer interaction. Most existing systems illuminate the eye with IR-light, possibly damaging the eye. The motivation of this thesis is to develop a truly non-intrusive gaze tracking system, using only a digital camera, e.g. a web camera.

The approach is to detect and track different facial features, using varying image analysis techniques. These features will serve as inputs to a neural net, which will be trained with a set of predetermined gaze tracking series. The output is coordinates on the screen.

The evaluation is done with a measure of accuracy and the result is an average angular deviation of two to four degrees, depending on the quality of the image sequence. To get better and more robust results, a higher image quality from the digital camera is needed.

Nyckelord (högst 8)
Keyword (8 words)
Gaze tracking, computer vision, image processing, face detection, rotational symmetries, neural network, backpropagation

Bibliotekets anteckningar:

Abstract

Gaze tracking means to detect and follow the direction in which a person looks. This can be used in for instance human-computer interaction. Most existing systems illuminate the eye with IR-light, possibly damaging the eye. The motivation of this thesis is to develop a truly non-intrusive gaze tracking system, using only a digital camera, e.g. a web camera.

The approach is to detect and track different facial features, using varying image analysis techniques. These features will serve as inputs to a neural net, which will be trained with a set of predetermined gaze tracking series. The output is coordinates on the screen.

The evaluation is done with a measure of accuracy and the result is an average angular deviation of two to four degrees, depending on the quality of the image sequence. To get better and more robust results, a higher image quality from the digital camera is needed.

Acknowledgments

First I would like to thank my supervisor Joakim Rydell at IMT, Linköping University for providing and mentoring this challenging thesis. Secondly, examiner Dr. Magnus Borga also at IMT. My appreciation also goes to Joel Petersson for the broadening opposition.

Many thanks to the other students at IMT, for making this autumn to an amusing time; to Joel, Anni and Sofia for standing by with their faces and contributing to an uplifting results chapter. And finally, Anni, thank you for discussing and proofreading and just being there.

David Bäck
Linköping, December 2005

– wysiwyg –

what you see is what you get

Contents

List of Figures

1

INTRODUCTION

1.1 Background

Gaze is the direction in which a person looks. Gaze tracking means to detect and follow that direction. This can be used in a variety of applications. For instance in human computer interaction, as a computer mouse based on eye-movement, or as an usability or advertising study of a web page.

Existing systems that detect and track gaze often use some illumination of the eye (e.g. infrared light), stereo cameras and/or some advanced head mounted system. This leads to cumbersome setups which may impair the system. Besides, IR light might not be entirely safe. Certainly IR affects the human eye since it is not totally reflected, due to the properties of the eye. Additionally it requires special hardware.

Therefore it would be interesting to develop a software gaze tracking system using only a computer and a digital video camera, e.g. a web camera. This would lead to truly non-intrusive gaze tracking and a more accessible and user friendly system.

1.2 Problem Description

The assignment of this work is, with aid of MATLAB and a web camera, to develop an algorithm that performs gaze tracking. Simply put; from an image sequence of a face detect its gaze. The aim is to make a functional real time system.

1.3 Method

At first the problem was defined by the supervisor. A computer and a web camera was provided. Some possible solutions were shortly discussed resulting in few restrictions.

In the pilot study information was obtained about different kinds of methods for facial feature detection and gaze tracking through articles and books. A final method was determined – to some extent with trial-and-error – as the work proceeded.

The problem can be divided into two parts.

Image Analysis To detect and track facial features, sufficient to estimate gaze, using image analysis.

Machine Learning Create and train a neural network with these facial features as input and gaze target (computer screen coordinates) as output.

This thesis is written in LaTeX.

1.4 Thesis Outline

Chapter 2 gives further introduction to the subject via previous work and existing applications. In chapter 3 an overview of how the problem was solved is given.

A brief explanation of the human visual system can be read in chapter 4. Chapter 5 describes the theory of rotational symmetries and chapter 6 explains the theory and which methods that have been used to detect and track the facial features. Chapter 7 gives an introduction to neural networks. It describes more thorough back-propagation in multilayer perceptrons and how it is applied to this thesis. In chapter 8 the implementation will be explicated.

In chapter 9 and 10 the results are discussed and proposals for future work are presented.

To facilitate the reading every chapter begins with a short summary of its content, or some major questions that will be answered.

Due to printing reasons the color images are, when possible, gathered on the same page. For the same reasons some of the color images are rendered as grayscale images. The figure description text notifies if the image should be in color.

2

PREVIOUS WORK

To better understand gaze tracking, existing solutions have been studied and in this chapter some of the possible ways to perform gaze tracking will be discussed. Some gaze tracking applications will also be illuminated.

2.1 Existing Gaze Tracking Systems

2.1.1 Infrared Light

One method often used to track gaze is to illuminate the eyes with one or more infrared light sources, usually LEDs (light emitting diodes), due to their inexpensiveness. Then a digital video camera or camera pair capture an image of the eyes and a computer estimates the gaze.

The LEDs can be positioned in different ways; in various geometric arrangements, on a head mounted system or near the computer monitor. Regardless of LED-positioning and camera setup, the gaze tracking roughly follows the same pattern.

First the specular reflections of the IR-light sources are determined in the pupil or iris in the image of the eye. This is easy to detect due to the brightness of the reflection on the dark pupil. Then the center of the pupil and cornea is estimated using for instance circle/ellipse-estimation, template matching or a simplified model of the eye and cornea. From this information an estimation of the optical axis is done. The optical axis passes through these two center points, perpendicular to the eye, but differs slightly from the visual axis (line of sight) which passes through the fovea, see figure 2.1. The angle α can be estimated through some approximations of geometrical conditions of the eye and head alignment, provided some calibration. With an approximation of α the gaze point can be estimated. In some solutions the user has to be still and sometimes a head mounted system is used. A calibration process ensuring the right arrangement is often needed. [2], [23], [3] & [6]

This type of gaze tracking system appears to be dominant on the market, and often has a high accuracy. The non-head-mounted systems seems to be easy to use and requires little calibration. So to compete with these existing system a robust and easy-to-use system needs to be developed.

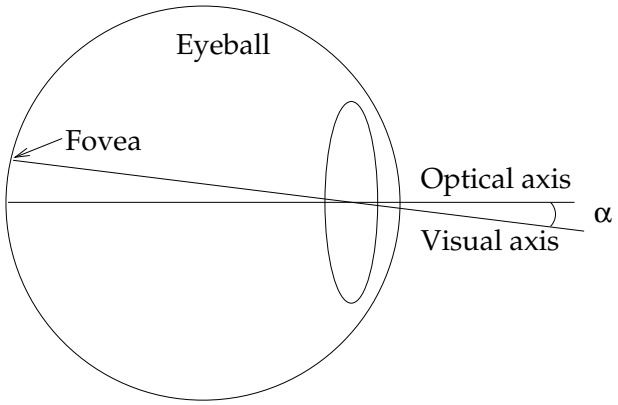

Figure 2.1: Schematic image of the human eye illustrating the difference between optical and visual axis.

2.1.2 Electro-Oculogram

Another way to determine gaze is to utilize an Electro-Oculogram (EOG). The EOG records eye movements by detecting impulses via electrodes near the eye. The recorded signals are typically low in amplitude and an amplifier must accurately be applied. The EOG-signals are with the aid of a training sequence on specific target points mapped to corresponding eye movements. [18]

It is difficult to comment the applicability of this system. It is not evident from the article how head movement is treated, whether or not the head must be still.

2.1.3 Truly Non-Intrusive Gaze Trackers

Some available systems are truly non-intrusive, meaning based only on passive computer vision systems. That is only measuring existing information, present in the scene. This in contrast to active systems that radiates light into the scene.

Even in this category a variety of solutions are available. For instance those based on neural networks – training the system with pictures of the entire eye, based on a model of the eye – estimating the visual axis from size and shape of the eyeball or some head mounted device to locate the eyes. [20], [16] & [25]

These types of systems also seems to be functional, but no consumer product based on it has been found. Only one numeral has been found on the accuracy of these types of gaze trackers, namely [25] reporting an accuracy that is slightly inferior to the IR-based gaze trackers. The developed system described in this paper will eventuate in this genre.

2.2 Applications

The possibility to estimate gaze can be used in many applications, both professional and consumer related. I will describe some applications suitable for the system de-

scribed in this paper, i.e. with the set-up of one person looking at a computer screen.

2.2.1 Human Computer Interaction

The most straightforward application for gaze tracking is human computer interaction (HCI). The desire is to improve the ability to interact with the computer. The use of gaze can possibly speed up and make interaction more intuitive and natural, e.g. in comparison to a computer mouse. It can also help a disabled user who has limited ability of controlling other interaction devices.

As will be mentioned in chapter 4 the best accuracy that can be achieved in a gaze tracking system is approximately one degree which is much coarser than a computer mouse. Therefore the application is limited. [11]

2.2.2 Usability and Advertising Studies

The usability and design of a website can be tested with a gaze tracking system. Test persons can perform different tasks by browsing the web site in question. Feedback from the system can be an activity map of the pages visited, showing what has been looked at. To improve usability design the web page can be planned from what has been looked at and be improved accordingly. This can also be applied on Internet advertising, studies of a target group can show what draws their attention. [1]

2.2.3 Video Compression

When transmitting real-time video with limited bandwidth (e.g. from an unmanned vehicle to an observer) it is desirable, but difficult, to get high image quality. To increase the perception of image quality it is possible to compress different parts of the image more or less than others. For instance higher quality is wanted on those sections that the observer is looking at. This can be implemented with the information from a gaze tracker. [21]

2.2.4 More

There are also other applications for gaze tracking, for instance in vehicle security and medical diagnostics. [1]

3

FRAMEWORK

This chapter intend to answer the following: How will this problem be solved, and on what conditions? Which interesting facial points should be tracked?

3.1 My Concept of Tracking Gaze

The setup of the system is described in figure 3.1. A user is looking at a computer screen and is at the same time being recorded by the web camera.

A short description of the framework and basic concept is presented in the form of a list. (With a reference to the section describing the item further.)

- The idea is to determine what facial features are necessary and sufficient to estimate gaze. (Section 3.2)
- Detecting and tracking these features, using appropriate image analysis techniques, will give a number of coordinates and parameters in the captured image sequence. (Chapter 6)
- With a set of predetermined gaze tracking series, a neural network will be trained using these coordinates. (Chapter 7)
- The neural network will be implemented as a back-propagation-trained multilayer neural network with one hidden layer. The output from the neural network is coordinates on the screen which the user is looking at. (Chapters 7 and 9)
- Gaze tracking is done by recording of new images, extracting coordinates of the facial points and sending them through the trained neural network.

3.1.1 Setup Restrictions

Some restrictions have been made designing this system. The users head is estimated to be at the same altitude as the web camera and approximately centered in the image. The setup can be seen in figure 3.1. For robust results the distance between head and camera is 30 to 70 centimeters and, because of the color based face finder, the

background is best not yellow-reddish. The user can unfortenately not wear glasses, contact lenses are however not an issue.

The interesting facial features are described in this chapter. The rest will be explained in connection with the presentation of relevant theory.

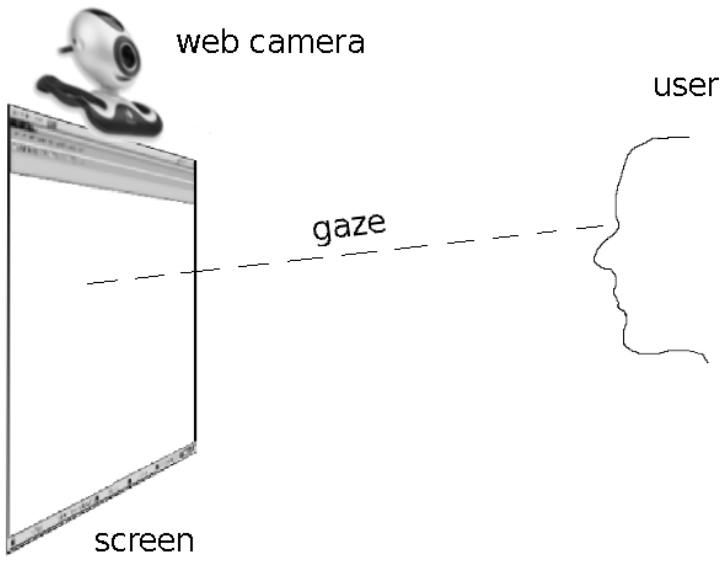

Figure 3.1: Schematic image of experimental setup.

3.1.2 Linear Regression

As an alternative to neural networks, linear regression was implemented. However the problem proved to be more complex than being solved linearly. With simple setups, few gazing points and very similar training and test sequences, it might function. But this will not be described in detail since the neural network solution had better performance and was the chosen solution.

3.2 Interesting Facial Features

The features that will be tracked must be easy to detect and must not differ from face to face. And they must also be quite rigid and not vary with different facial expression. Hence, the corners of the mouth, despite that they are easy to detect, are not interesting. Another relatively rigid feature is the ear. However it can be covered with hair and is not visible when gazing askew, therefore rejected.

What facial features are needed to estimate gaze is hard to determine, but previous work and an educated guess has lead the following, see figures 3.2 to 3.5.

Center of Iris Naturally this point is of interest, since it is on the visual axis and determining what is focused on. It is also relatively easy to detect.[1] It will be referred to as COI.

Corner of Eye It is much easier to find the outer (as against the inner) corner of the eye (COE) due to the resolution of the camera and a more visible contrast between the sclera and surrounding skin. In relation to the center of iris these points reveal, to a large extent, the gaze.

Nostrils The nose is relatively difficult to find, since it has no sharp boundaries. In spite of that a shadow is often seen between the nose and cheek irrespective of lighting conditions. Because they are positioned on a different distance from the camera, relative to the eyes, it gives an indication of head alignment. Hence these two points, the outsides of the nostrils, are also wanted.

Head Angle The angle of the head relative the camera gives information of head alignment. It is determined by the line that crosses the two COIs, the eyeline, see figure 3.2.

Eye Angles The sclera is here defined as two regions in each eye. The specific point, COS (Center of Sclera) that defines the angle, is its center of mass. The mass corresponds to the intensity in the image. To get further information about eye positioning two angles, for each eye, will be calculated. They are the angles between the eyeline and the COI-COS lines, see figure 3.4. This gives two angles for each eye, γ_1 and γ_2.

Relative Distance Iris-Sclera The distances between the COSs and COI, b_1 and b_2, are projected onto the eyeline, resulting in d_1 and d_2, see figure 3.3. These two gives a relative measure of eye positioning $\mathit{diff} = d_1 - d_2$.

This results in 13 features to track and follow.

L_COI	R_COI	L_COE	R_COE	L_NOSE	R_NOSE	
β	γ_{L1}	γ_{L2}	γ_{R1}	γ_{R2}	diff_L	diff_R

The first row is represented as (x, y)-coordinates, in the image of the user, and the second row with angles in radians and *diff* in pixels. L and R indicates the left and right eye respectively.

None of the described systems in section 2.1.3 (the Truly Non-Intrusive Gaze Trackers) use facial features in this way to determine gaze. They use instead for instance grayscale images of the eye as input to a neural network. Hence, the features described above have not before proved to be enough for determining gaze so that remains to be seen.

[1] The knowledgeable may object to using the word iris instead of pupil. The reason is that it is difficult to distinguish the pupil from the iris due to poor quality in the images used.

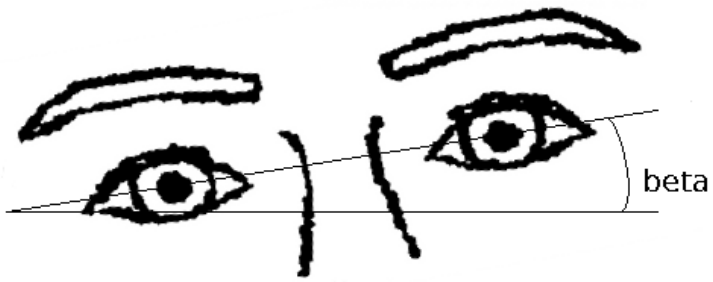

Figure 3.2: Image showing the facial feature head angle, β, between the eyeline and the horizontal line.

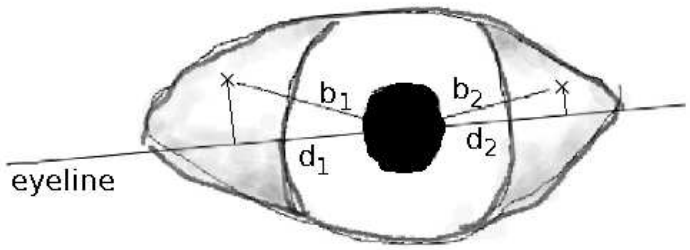

Figure 3.3: Schematic image showing the center of sclera, COS. The crosses indicate the center of mass for each side of the iris. b and d are distances from COI to COS and COI to the projection of COS on the eyeline respectively.

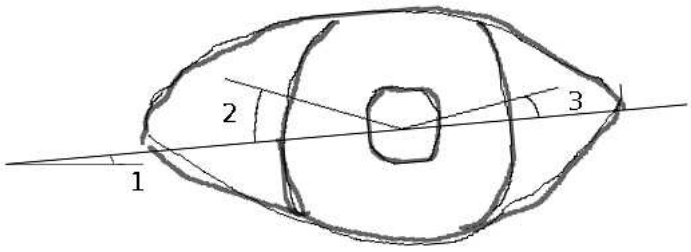

Figure 3.4: Schematic image showing the facial features (1) β, (2) γ_1 and (3) γ_2.

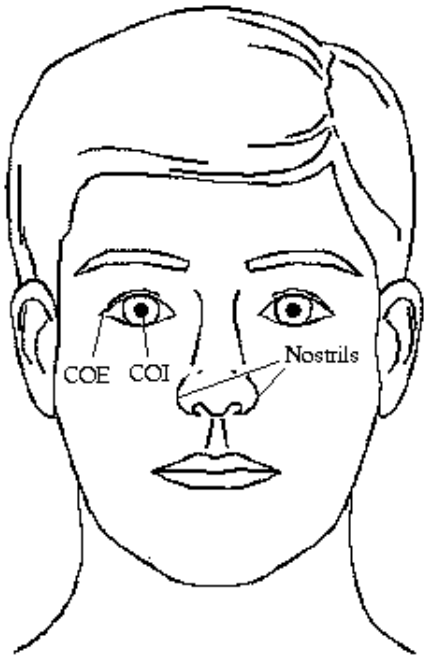

Figure 3.5: Image showing interesting facial features, corner of the eye (COE), center of iris (COI) and the nostrils. Original image from [19].

3.3 Provided Resources

To help solve this problem the following was provided:

Web Camera A Logitech QuickCam ® Pro 4000 is the image acquisition tool. The maximum resolution 640x480 pixels is needed for desirable image quality. A frame rate up to 30 frames per second is available.

Software The implementation is done in MATLAB 7, Service Pack 3, and with the Image Processing Toolbox. To access image sequences in MATLAB the free software Vision For MATLAB (VFM) is used as a frame-grabber.

Computer Mainly a Sun Ray 1G client is used, but a laptop computer with Microsoft Windows XP is needed to collect training and test sequences. This because of compatibility issues.

4

HUMAN VISUAL SYSTEM

When making a gaze tracking system it is important to understand the human visual system, partly to understand how it is possible to estimate gaze and partly to understand the built-in limitations.

4.1 Functional Description

The visual system starts with the eyes where incident light is refracted by the cornea and the lens. The photoreceptors on the retina transduces light into receptor potentials which are further converted to nerve impulses propagating the optic nerve to the visual area in the cerebral cortex. [24] See figure 4.1.

There are two types of photoreceptors on the retina, rod and cone cells. Rods are very sensitive to light, making it possible to see in poor lighting conditions. On the contrary they are not capable of producing color vision, which leads to that all cats are gray in the dark. The cones on the other hand need brighter light to be activated, but are then capable of rendering color vision. It is shown that with three (or more) light sources with well separated wavelengths any color can be produced.[1] This is probably the reason that the cones have three color filters (proteins) sensitive to different wavelengths of light: red, green and blue light. [24] & [8]

The sclera makes it possible to detect gaze, fixing the center of the pupil would be much more difficult without it. One of the reasons humans have sclera is because we are social beings. Since we have developed a sophisticated communication technique it is interesting and important for the attenders to know where we look. Therefore humans have one of the largest sclera/iris-quote amongst animals, making it easy to see what we look at. This in contrast to some animals with wider field of view whose communication is more simple and limited, they must also move their entire head to look at something. [17]

The acuity of human vision is concentrated to the fovea, placed on the center of retina, see figure 4.1. The fovea is a small pit, about 2 millimeter in diameter, with a dense accumulation of receptors, mostly cone cells allowing color-perception. The other

[1] Not really “any color”, but all in the available color gamut.

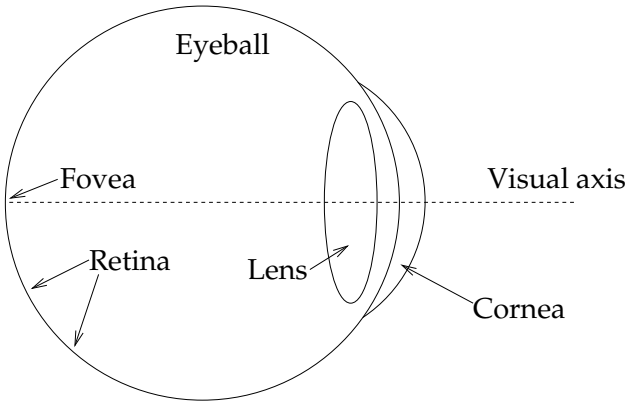

Figure 4.1: Schematic image of the human eye.

part of the retina has an acuity of 15 to 50 percent of that of the fovea. This leads to about one degree field of view with high resolution. Hence, to see an object clearly it has to be within that angle. For instance the peripheral vision is not sufficient to read a text. Therefore, even though it is possible to focus on an object smaller than the fovea, it is impossible to determine gaze with a higher accuracy than one degree. [11]

4.2 Eye Movements

When we move our gaze attention it is often done with small high-speed movements, called saccades. They are typically in the range of 15 to 20 degrees and with a duration of 30 to 120 milliseconds. During these periods the vision is largely impaired. This leads to that before one saccade starts, the target of next gaze point must be chosen. Since the destination often lies in the area of peripheral vision, the target must be chosen with lower resolution. [11]

Between the saccades there is fixation, a state distinguished by relatively stable gazing and with small motions of the eye (microsaccades, mostly within one-degree radius). These last typically between 200 to 600 milliseconds. To make the eyes move smoothly they need something to follow. When looking at a static computer screen, saccades are the only occurring motion. [11]

This leads to some expectations on the gaze tracking results. Steady gaze for periods of a few hundred milliseconds and rapid saccades in-between. The ability to detect these fast movements depends obviously on the image capture device. Unless a slowly moving flash movie or similar appears on the screen, no smooth eye movements ought to be detected.

5

ROTATIONAL SYMMETRIES

This chapter will shed light on the theory of rotational symmetries. The outcome enables eye detection.

To make this theoretical review more comprehensible there will be a continuous referring to the actual problem. Specifically a test image, see figure 5.1, will illustrate many steps in the algorithm.

This entire chapter has been inspired by [8], [12] & [13].

5.1 Introduction

Because of the web camera position and direction the user is approximately at the same altitude as the camera and is facing it frontally. This implies that the iris can be estimated to a circle. To find this circle the theory of rotational symmetries will be used.

The idea is to represent the local orientation of the image and then correlate with a template filter with the corresponding appearance of the searched area.

Figure 5.1: Original test image, $im(x, y)$, resolution 640x480 pixels, in color.

5.2 Local Orientation in Double Angle Representation

A description of the local orientation is needed, which can be actualized in many ways. A simple starting point is the image gradient, since it indicates the locally predominant direction. This is often represented as a 2D-vector indicating the dominant direction as an angle defined from the positive x-axis, ranging from $-\pi$ to π. The image gradient can be estimated through convolving the image with partial derivatives, g_x and g_y, of a gaussian function, g, with variance σ, see equations 5.1 to 5.3 and figure 5.2. The spread of g_x on the y-axis and g_y on the x-axis provides a low pass effect.

$$g(x,y) = \frac{1}{\sigma\sqrt{(2\pi)}} \cdot exp\,(-\frac{x^2+y^2}{2\sigma^2}) \tag{5.1}$$

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = -\frac{x}{\sigma^2}\, g(x,y) \tag{5.2}$$

$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = -\frac{y}{\sigma^2}\, g(x,y) \tag{5.3}$$

One disadvantage of estimating the gradients with gaussian functions is unreliable results near the image border. However the face is restricted to be centered in the image and is therefore not a problem.

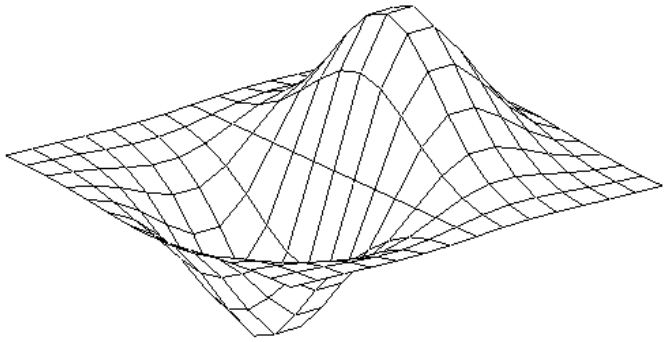

Figure 5.2: Schematic plot of $g_x(x,y)$.

With $*$ denoting convolution, the derivative along the x-axis can be put

$$im_x(x,y) = (im * g_x)(x,y) \tag{5.4}$$

and correspondingly

$$im_y(x,y) = (im * g_y)(x,y) \tag{5.5}$$

for the test image $im(x,y)$.

To get the image gradient in complex form the following is calculated:

$$im_{grad}(x,y) = im_x + i \cdot im_y = c(x,y) \cdot exp(i\,\theta) \tag{5.6}$$

with $c(x,y) = abs(im_{grad})$, $\theta = arg(im_{grad})$ and i is the imaginary unit.

And for the final local orientation representation the angle of im_{grad} is mapped to the double angle orientation:

$$z(x,y) = c(x,y) \cdot exp(i\,2\theta) \tag{5.7}$$

The result can be seen in figure 5.6. The plot function color codes the gradient with each color representing one orientation. The magnitude of z, $c(x,y)$, can be used as a measure of signal certainty. Figure 5.3 explains how the local orientation corresponds to the double angle.

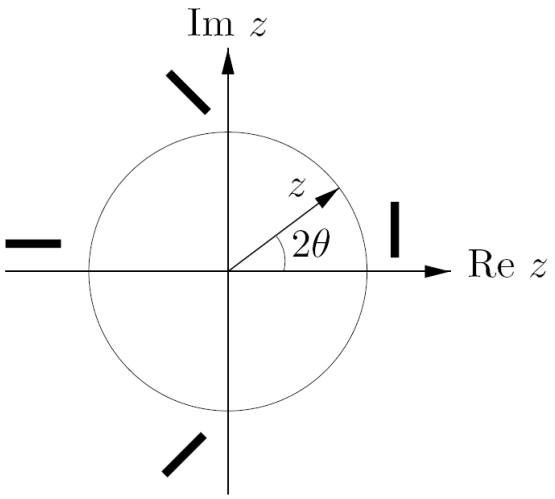

Figure 5.3: Double angle representation of local orientation. Image from [13]

There are at least three advantages with double angle representation:

- There are no discontinuities in the double angle representation. In single angle representation there is a leap between $-\pi$ and π, though they in fact correspond to the same orientation. This is avoided here and an angle θ has the same representation as $\theta + \pi$, namely $exp(i\,2\theta)$.
- This implies that two orientations with maximum angular difference ($\pi/2$) is represented as two opposite vectors.
- The double angle representation also makes averaging possible, which is useful when processing and merging a vector field.

5.3 2:nd Order Rotational Symmetries

A function can be called a rotational symmetry if the orientational angle, $arg(z)$, only depends on θ. This is a rather wide definition and specifically the n:th order rotational symmetry is defined as

$$z = c(x, y) \cdot exp(i\ (n\theta + \alpha)) \tag{5.8}$$

with $\alpha \in [-\pi, \pi]$, indicating signal phase.

By varying n we get different families of rotational symmetries, with different members depending on α. For example the 1:st order symmetries consists of parabolic structures and the 2:nd order includes circular and star patterns, depending on phase. For examples of 2:nd order symmetries see figure 5.4.

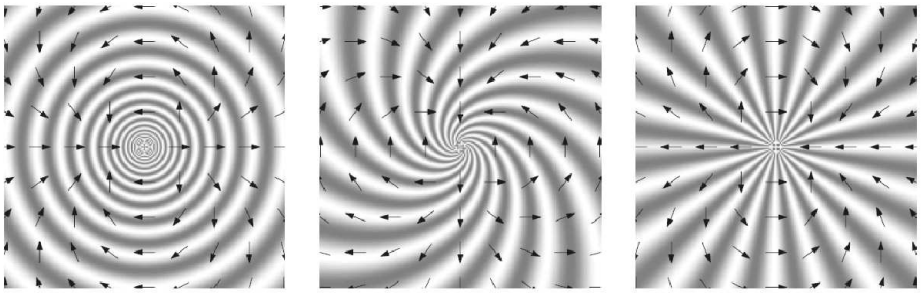

Figure 5.4: Examples of image patterns and corresponding local orientation in double angle representation. The argument of z from left to right: 2θ, $2\theta + \pi/2$ and $2\theta + \pi$. Image from [12].

Convolving with a function of same angular variation as z makes it possible to detect these rotational symmetries. If a circular pattern is searched the local orientation image (still in double angle representation) is convolved with the template

$$b = a(r) \cdot exp(i\ 2\theta) \tag{5.9}$$

where $a(r)$ is a radial magnitude function, describing and limiting the spatial extent of the filter, see figure 5.5. The result of the convolution is calculated according to equation 5.10 and can be seen in figure 5.7.

$$s(x, y) = (z * b)(x, y) = \sum_{\chi} \sum_{\psi} z_k(x, y)\ b(\chi - x, \psi - y) \tag{5.10}$$

$s(x, y)$ can also be written in complex representation as $|s| \cdot exp(i\ \varphi)$. The argument of s, $\varphi \in [0, 2\pi[$, describes what member, in the second order rotational symmetry family, is locally predominant. How to interpret the phase of s is described in figure 5.7. The two green areas indicate circular patterns and form the result of the coarse eye finder.

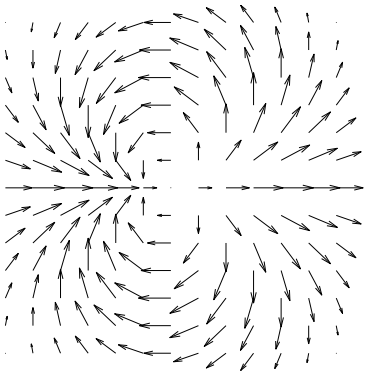

Figure 5.5: The circular symmetric template, b.

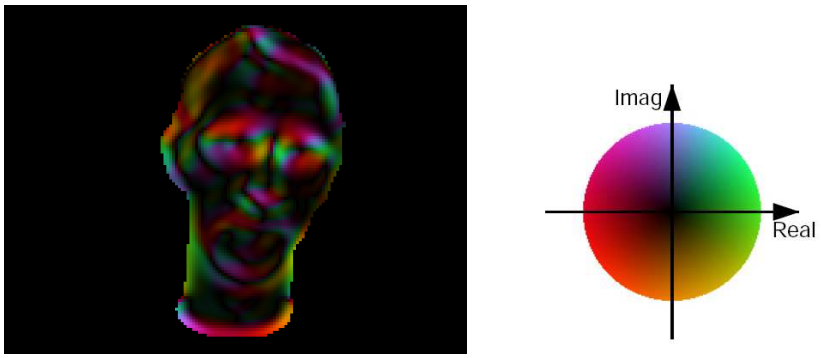

Figure 5.6: Image showing z with the face mask applied, and the color representation of complex numbers. Right image from [14].

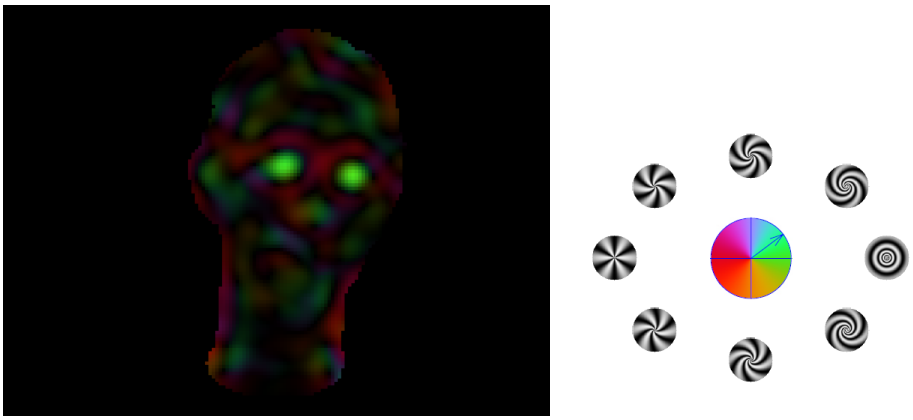

Figure 5.7: The result, s, of convolving z with b, with the face mask applied. Right image shows how the phase of s should be interpreted, from [14].

6

DETECTION AND TRACKING OF FACIAL FEATURES

The theory behind the detect-and-track algorithm and how it has been carried out will be described here. The same test image will also in this section illustrate the algorithm.

6.1 Color Space Transformation to Find the Face

To find these facial features it would ease to first locate the face. Since it is a rather specific situation, some simplifications have been made. For instance the algorithm only searches for one face.

6.1.1 Re-sampling and Low Pass Filtering

To get a faster algorithm and to reduce noise the image is low pass filtered and downsampled. Faster because an image with fewer pixels is less computationally demanding and a less noisy image beacause this leads to a more homogeneous face image. See figure 6.1.

Figure 6.1: Left: Original image, resolution 480x640 pixels, in color. Right: The lowpass filtered and downsampled image of the user, resolution 120x160 pixels, in color.

6.1.2 Color Spaces – RGB to YCrCb

A color space is a definition of how we can describe the different wavelengths of the electromagnetic spectrum, especially those visible to the human eye. The most common way to render color images is to use the RGB-space, which is a three-dimensional color space consisting of the additive primaries red, green and blue. This can be derived from the photoreceptors on the retina, see chapter 4.

The web camera provides RGB-coded images. In MATLAB a color image is represented as an 3D-array of three separate 2D-images, one image for each primary. In MATLAB a digital image is equivalent to a matrix with each post corresponding to one pixel.

Many different color spaces are mentioned for face recognition. The one that was found most suited and easy to use was the YCrCb-space, which is used in video systems. It consists of one luminent component Y, representing brightness, and two chroma components, Cr and Cb, representing color. For mathematical expressions see equations 6.1 - 6.3.

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{6.1}$$

$$Cr = 0.713 \cdot (R - Y) \tag{6.2}$$

$$Cb = 0.564 \cdot (B - Y) \tag{6.3}$$

Where R, G and B are the three elements in the RGB-space. [5], [10], [15] & [22].

6.1.3 Morphological Operations

When thresholding or creating binary images in purpose of segmenting an image the result is not always the desired. Noise or irregularities in the image may disturb the result. Some post processing is often needed in the form of morphological operations.

Structure elements (small binary kernels) operates on the binary image with basic operations changing its shape.

Dilation Adds pixels to the object boundaries. Can be performed as convolution with a structure element followed by thresholding.

Erosion Removes pixels to the object boundaries or alternatively adds pixels to the background boundaries. Performed similarly.

These basic operations can be expanded into more complex operations such as filling binary holes, removing objects and skeletonization. [7]

6.1.4 Face Finder Implementation

The final face finder only uses the Cr component. An iterative thresholding of the Cr image gives a few possible skin areas. A binary mask is created, with ones indicating possible skin areas and zeros indicating background. The largest connected region is set as the face. This face mask is then a morphologically filled, meaning

that if there are any enclosed holes in the face area, these are also set to ones. See figures 6.2 to 6.4.

Figure 6.2: Left: The lowpass filtered and downsampled image. Right: A Cr-image of the user.

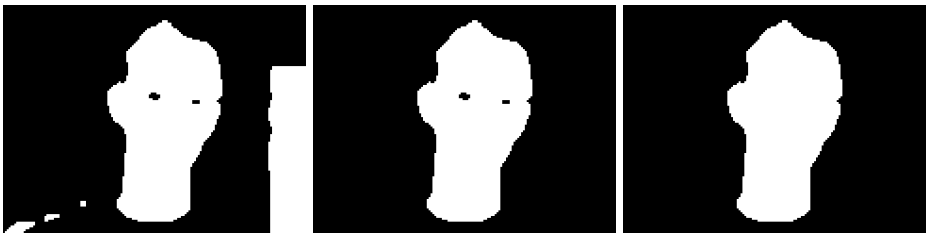

Figure 6.3: Sequence in the face finding algorithm: (1) the result of iterative thresholding of the Cr-image, (2) the largest connected region and (3) a morphologically filled image of (2).

Figure 6.4: Image showing the result from the face finding algorithm.

6.2 Coarse to Fine Eye Detection

This segmented image of the face will be the starting point for the eye finder algorithm, which is done in two steps. First the downsampled image gives a region of interest and then, in the original sized image, a more precise search for the center of iris is done.

6.2.1 Rotational Symmetries

In the face image the eyes are two blurry circular-shaped discs. Hence, the theory of rotational symmetries, described in chapter 5, is used. The algorithm can be summarized to the following:

- estimation of the image gradient
- gradient in double angle representation
- convolution with template filter
- find two areas with maximum values, satisfying geometrical terms

These two eye candidates are morphologically dilated into a binary eye mask, see figure 6.5, because this only gives a coarse eye location, not sufficient for gaze tracking. The eye mask is resampled to the original size. Approximate values of iris positions are given from the lowest pixel values, in the local averaged eye image. Then each iris center is found separately with rotational symmetries, assuming the iris being circular. See figure 6.6. The method here follows the same pattern as the list of items above.

The reason for implementing this symmetry based eye finder in two steps, coarse to fine, is to get a more robust result.

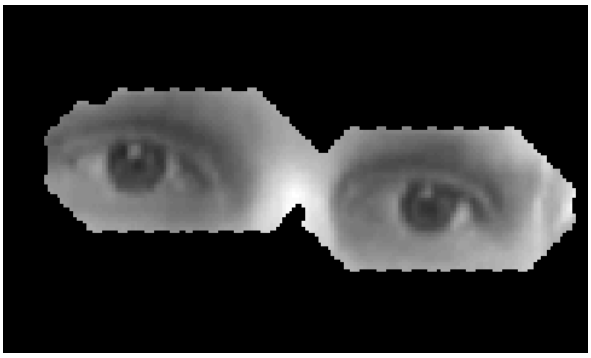

Figure 6.5: Zoomed image of the users eyes showing the result of the coarse eye finder.

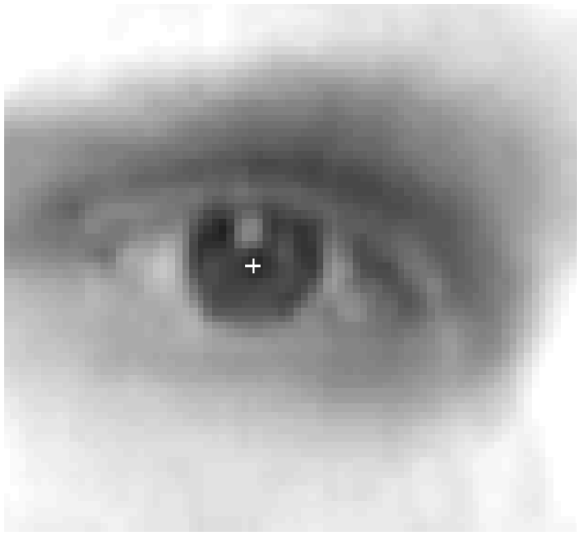

Figure 6.6: Zoomed image of the users right eye showing the result of the iris detection with a cross.

6.3 Detecting Corner of Eye

The detection of the outer corners of the eyes is a quite simple pixel based algorithm. A line between the two located iris centers is calculated. Along the prolongation of this line, an edge is searched. The edge represents the part where the sclera meets the eyelid.

The eye line is averaged with the filter $[1\ 1\ 1]/3$, to reduce noise. The edge is found by searching for the maximum pixel difference (going from lighter to darker pixel values) on the averaged line.

An attempt was done to improve the algorithm with the theory of rotational symmetries, looking for parabolic structures. However, the image quality was not sufficient to enhance the result.

Also a correlation technique was tried using a template corresponding to the COE-

area. This did not work either, mostly because of the large variety of COE appearance.

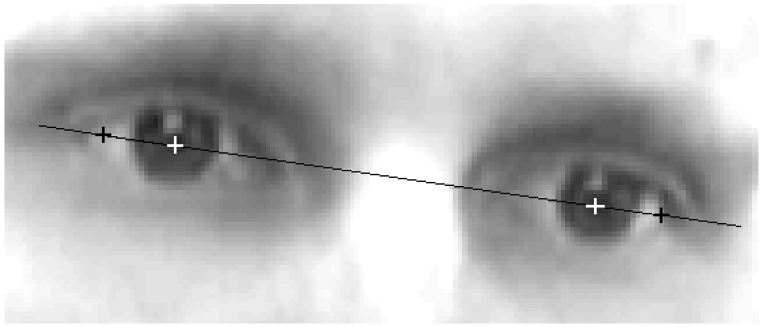

Figure 6.7: Zoomed image of the users eye showing the result of the COE-finder.

6.4 Detecting Nostrils

This algorithm searches for the widest gap between the two nostrils. It starts from the line between the eyes. On the center of this line a perpendicular line points out the approximate location of the nose. A loop finds the widest gap perpendicular to this new line based on pixel value variations. See figure 6.8.

Initially the tip of the nose was searched for, but as described earlier its low frequency appearance makes it difficult. Even though a few methods was attempted, among them estimating the angle of the inner nostrils, none of them improved the result.

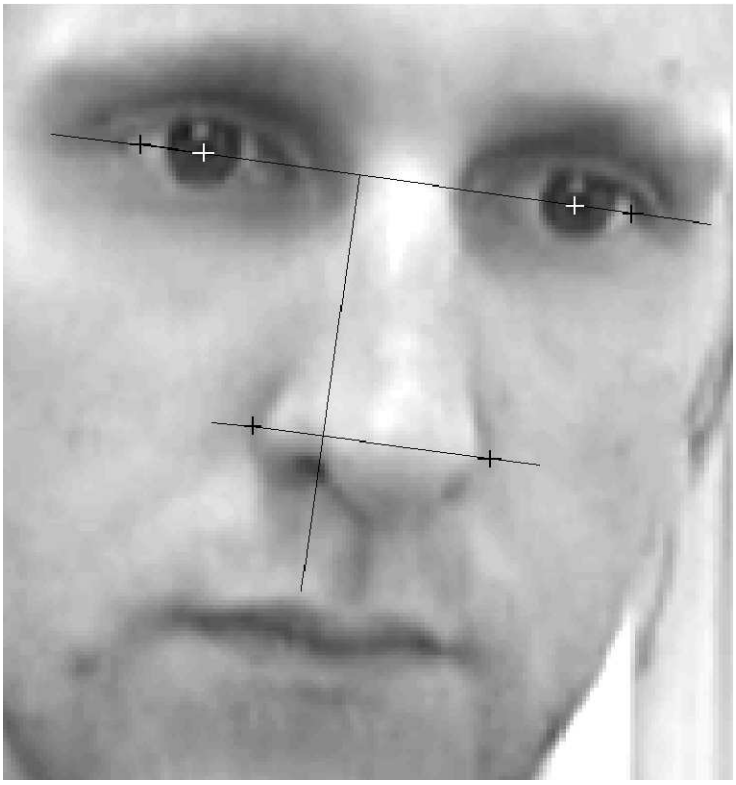

Figure 6.8: Zoomed image showing facial features detected.

6.5 Geometrical Features

The detection of the remaining features are shortly described in the following list. However, first the COS must be defined.

The sclera is divided by the iris in two regions and is estimated to the 10 % of the pixels that has the highest intensity, in the image of the eye, $im_{eye}(x, y)$, see figure 6.9. The coordinates of the COS, (COS_x, COS_y), is then estimated as the center of mass with subpixel accuracy.

$$COS_x = \frac{\sum_{\forall\ (x,y)} im_{eye}(x, y) \cdot x}{\sum_{\forall\ (x,y)} im_{eye}(x, y)} \tag{6.4}$$

$$COS_y = \frac{\sum_{\forall\ (x,y)} im_{eye}(x, y) \cdot y}{\sum_{\forall\ (x,y)} im_{eye}(x, y)} \tag{6.5}$$

Head Angle The head angle is easy to calculate, starting from the COI-coordinates

$$\beta = \tan^{-1}(COI_{diffy}/COI_{diffx}) \tag{6.6}$$

Where COI_{diffx} and COI_{diffy} are the pixel difference in coordinates of the COIs relative to the x- and y-axis respectively.

Eye Angles These two angles (for each eye) are in a similar way calculated from the coordinates of the COIs and the COSs.

COI-COS Measure The projection of b_i (the distance between COI and COS) on the eyeline is $d_i = b_i \cdot \cos(\gamma_i \pm \beta)$.

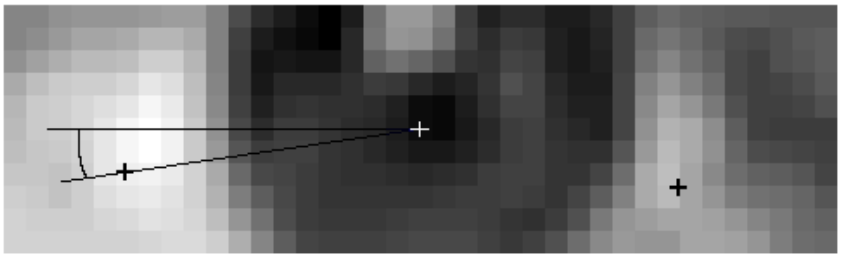

Figure 6.9: Image indicating the COS (black crosses) of the left eye in the test image. White cross is the COI. The acute angle defined by the black lines is γ_{L1}.

7

NEURAL NETWORK LEARNING

This chapter will simply answer the following question: How can the computer learn what you look at? On the way the theory of neural networks and back-propagation in multilayer perceptrons will be presented.

The learning process can be implemented in many ways and this chapter will only focus on the relevant theory for this thesis. The theory in this chapter is based on [9] & [4].

7.1 Introduction

The neural network part of this work implies that the system initially is not ready to use but must be trained. The training is done by updating parameters which improves the system behavior. Specifically in supervised learning a teacher monitors the learning process and provides the key, in the form of input-output training examples. Typical applications for neural network are classification and pattern recognition.

The neural network has a single model it strives to imitate, a model that obviously works and has been developed by itself.

7.1.1 Imitating the Brain

In the human brain there are billions of nerve cells (neurons). Together with the spinal cord they constitute the central nervous system, the majority of the nervous system. The other part is the peripheral nervous system which includes the remaining body, such as limbs and organs. In the neurons different stimuli propagate and are processed into the right response, e.g. a muscular contraction. The neurons consist of dendrites, a cell body, an axon and synaptic terminals, see figure 7.1. The dendrites pass signals from previous neurons (their synapses) and if the weighted sum of these signals are higher than a certain threshold the signal propagates through the axon. To better understand the function of these parts see for instance [24].

So, when creating a neural network the idea is to simulate this appearance and behavior. A model of the neuron, the perceptron, will further be described in section 7.2.1.

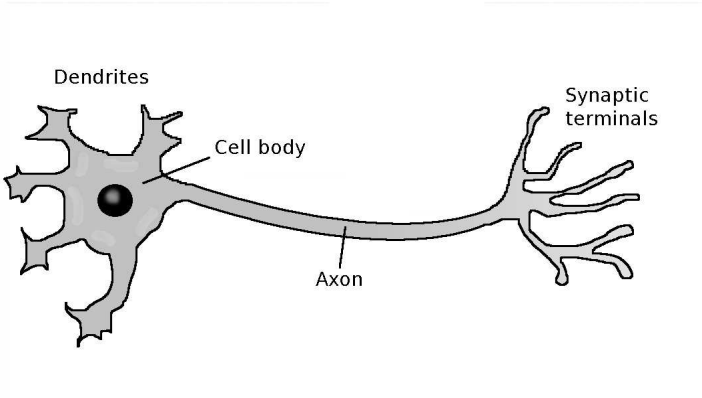

Figure 7.1: Schematic view of a neuron.

7.2 Multilayer Perceptrons

7.2.1 Perceptron

The perceptron is the building block in the neural net and has a number of inputs, $\mathbf{x} \in x_1, x_2, \ldots, x_K$, and one output, y_j, where j indexes the associated perceptron.

Similarly as in the neuron the input signals are weighted and affect the outcome of the perceptron. The synaptic weights are the free parameters of the perceptron and will be updated until the perceptron behaves as desired.

If the sum of these weighted signals exceeds a certain threshold (determined by the activation function) there is an output signal. The perceptron is mainly built up by three basic parts, see figure 7.3, which can be derived from the neuron.

Dendrites Each dendrite has a synaptic weight, w_{jk}. Index j refers, as before, to the current perceptron and index k indicates the associated input. The input signal x_k is multiplied with the weight w_{jk}.

Junction Here the weighted signals are added up.

$$s_j = \sum_{k=1}^{K} w_{jk} x_k + b_j = t_j + b_j \tag{7.1}$$

The external weight b_j has the ability of making affine transformation of t_j possible. The bias can be seen as a fixed input $x_0 = +1$ with the synaptic weight $w_{j0} = b_j$.

$$s_j = \sum_{k=0}^{K} w_{jk} x_k \tag{7.2}$$

Activation Function To limit the output signal and to introduce nonlinearity, an activation function, $\varphi(s_j)$, is used. It is typically a step function, limiting y_j to

the unit interval $[0, 1]$. It will later be necessary with a continuous and differentiable activation function. Therefore a sigmoid function will be used, namely the hyperbolic tangent function, see figure 7.2.

$$\varphi(s_j) = \tanh(s_j) \tag{7.3}$$

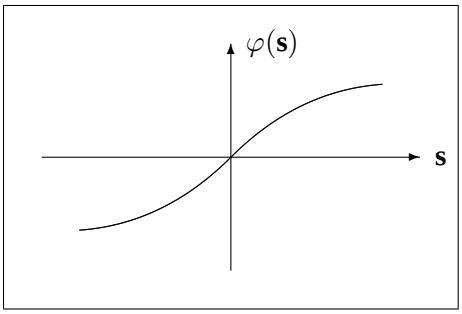

Figure 7.2: Sigmoid activation function, $\varphi(\mathbf{s}) = \tanh(\mathbf{s})$

One limitation in a single perceptron is that it only solves linearly separable problems. For instance when classifying two different classes they must be separated from each other with a decision surface that lies in a hyperplane. To avoid this limitation the perceptrons can be put consecutively in layers and form a net. Which lead us to the next section.

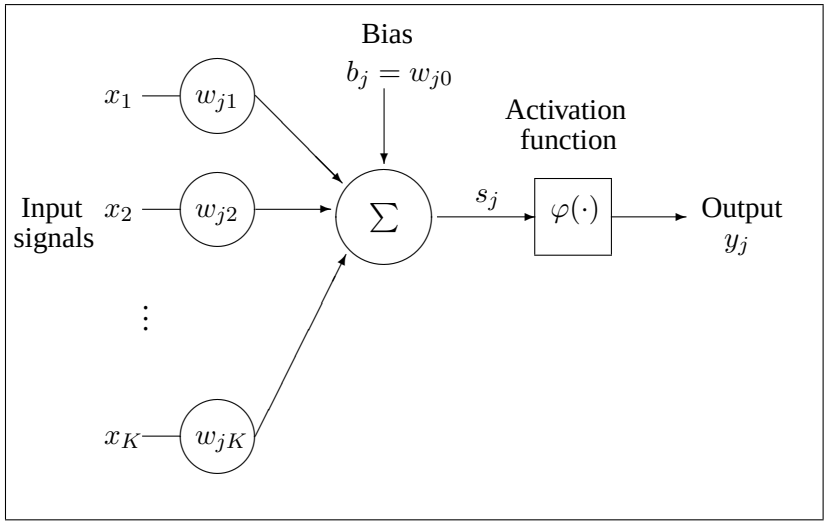

Figure 7.3: Schematic image of the perceptron.

7.2.2 Two Layer Perceptron

These perceptrons can be put together and constitute nodes in a neural network. Figure 7.4 shows an example of a fully connected two layer network. It is called fully connected because all the nodes in one layer are connected to all the nodes in the previous layer. The signals propagates parallelly through the network layer by layer.

Since a single layer perceptron only is capable of solving linearly separable problems, a hidden layer is added to make the network learn more complex tasks. Hence, the purpose of the hidden layer is to make the problem linearly separable.

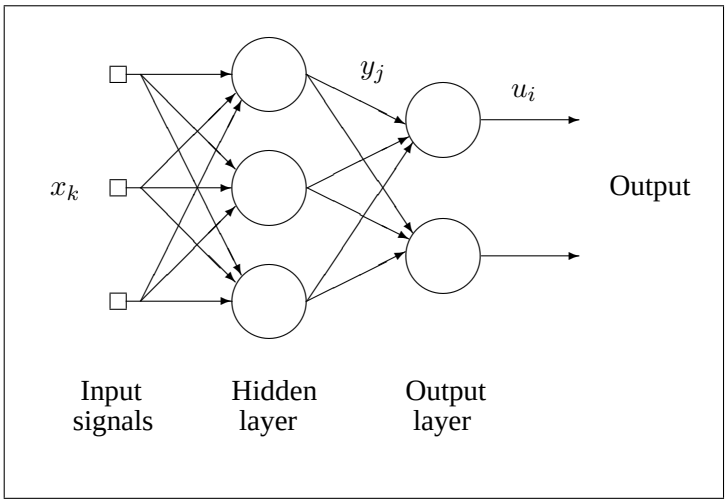

Figure 7.4: Schematic image of a two layer neural network with three input signals, three hidden neurons and two output neurons.

This second layer also brings a second set of synaptic weights, v_{ij}, and output signals u_i, see figure 7.4. This calls for some new definitions:

$$u_i = \sum_{j=0}^{J} v_{ij}\, y_j \tag{7.4}$$

where

$$y_j = \varphi(s_j) \tag{7.5}$$

and

$$s_j = \sum_{k=0}^{K} w_{jk}\, x_k \tag{7.6}$$

with J denoting the number of hidden neurons and v_{i0} is the bias of the second layer. Note that there is no activation function on the output layer, the reason for that is described in section 8.1.

The example in figure 7.4 is used to clarify the size of the synaptic weights. **W** should have twelve elements, four (as in the number of input signals plus bias) for each hidden neuron. Similarly **V** should have eight elements, four (as in the number of hidden neurons plus bias) for each output neuron.

7.3 Error Back-propagation Algorithm

Error back-propagation learning is the chosen method to update the free parameters of the neural net. It is based on an error-correction technique.

7.3.1 Learning by Error-Correction

Consider one perceptron, having a set of input-output training samples with input signals $\mathbf{x}_n$ and the desired responses d_i. The synaptic weights are initially randomized, ranging both positive and negative values. The output u_i gives rise to an error signal $e_i = d_i - u_i$. The task is now to minimize e_i by updating the synaptic weights.

To indicate the time step in the iterative process of updating the weights, the index m, denoting discrete time, is added to the signals.

$$e_i(m) = d_i(m) - u_i(m) \tag{7.7}$$

To get d_i closer to u_i we need to minimize a cost function based on the error signal, $e_i(m)$. Therefore the measure of instantaneous energy, based on the error signal, will be used.

$$\mathcal{E}_i(\mathbf{W}) = \frac{1}{2}e_i^2(m) \tag{7.8}$$

W represents the weights. The least-mean-square (LMS) algorithm will find the parameters (weights) that minimizes $\mathcal{E}_i(\mathbf{W})$. In the weight space we move in the direction of the steepest descent and the updating of the weights follows:

$$\mathbf{W}(m+1) = \mathbf{W}(m) + \Delta\mathbf{W}(m) \tag{7.9}$$

Where the last term is given by the delta rule:

$$\Delta\mathbf{W} = -\eta\frac{\partial\mathcal{E}_i(\mathbf{W})}{\partial\mathbf{W}} \tag{7.10}$$

η indicates the learning-rate parameter, which is a positive constant setting the pace in the learning process. The minus sign indicates the decrease of $\mathcal{E}_i(\mathbf{W})$ along the gradient descent in weight space. $\mathcal{E}(\mathbf{W})$ will in the future still depend on **W**, but the notation will be reduced to $\mathcal{E}$ from now on.

7.3.2 Back-propagation

The back-propagation algorithm is a supervised learning technique based on error correction and works by passing through the network in two ways.

Forward Pass The input signal propagates through the layers of the network producing the output signals. During this process the synaptic weights are fixed.

Backward Pass Here the error signal propagates backwards in the neural net, updating the synaptic weights. This in order to make the outcome closer to the desired.

Starting from the error-correction method a definition of the error signal energy is made:

$$\mathcal{E}(n) = \frac{1}{2}\sum_{i=1}^{I} e_i^2(n) \tag{7.11}$$

where I is the number of neurons in the output layer and n indexes the current training sample. If a set of N training samples is at hand, the average squared error energy is calculated according to

$$\mathcal{E}_{av} = \frac{1}{N}\sum_{n=1}^{N}\mathcal{E}(n) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{i=1}^{I} e_i^2(n) \tag{7.12}$$

The weights will be updated with the target of minimizing $\mathcal{E}_{av}$ according to:

$$w_{jk}(m+1) = w_{jk}(m) + \Delta w_{jk}(m) \tag{7.13}$$

and

$$v_{ij}(m+1) = v_{ij}(m) + \Delta w_{ij}(m) \tag{7.14}$$

where $\Delta w_{jk}(m)$ depends on the learning process. In the case of back-propagation the delta rule is defined as

$$\Delta w_{jk}(m) = -\eta\frac{\partial \mathcal{E}_{av}(m)}{\partial w_{jk}(m)} \tag{7.15}$$

and

$$\Delta v_{ij}(m) = -\eta\frac{\partial \mathcal{E}_{av}(m)}{\partial v_{ij}(m)} \tag{7.16}$$

This derivative is the reason for the necessity of a differentiable activation function.

7.3.3 Derivation of Delta Rule

To derive the delta rule the target of minimization is defined as:

$$\mathcal{E}_{av} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{i=1}^{I}(d_i(n) - u_i(n))^2 \tag{7.17}$$

The derivation will be done for the output and hidden layer respectively. Ergo, the concept is to use LMS to minimize $\mathcal{E}_{av}$ with respect to the synaptic weights.

Output Layer The chain rule and equation 7.4 gives that the derivative of 7.17 with respect to **V** can be written as:

$$\frac{\partial \mathcal{E}_{av}}{\partial v_{ij}} = \frac{\partial \mathcal{E}_{av}}{\partial u_i}\frac{\partial u_i}{\partial v_{ij}} = \frac{1}{N}\sum_{n=1}^{N}(u_i(n) - d_i(n))y_j(n) \tag{7.18}$$

If the indices are removed, the sum can be represented with matrices. The gradient is:

$$\Delta \mathbf{V} = \frac{\partial \mathcal{E}_{av}}{\partial \mathbf{V}} = (\mathbf{U} - \mathbf{D})\mathbf{Y}^T \tag{7.19}$$

The shape and content of these matrices will be further discussed in section 8.1.

In the search for optimum, a step is taken in the negative direction of the gradient. The size of the step is determined by η. Hence the updating of the output weights follows:

$$\mathbf{V}(m+1) = \mathbf{V}(m) - \eta \Delta \mathbf{V} = \mathbf{V}(m) - \eta(\mathbf{U} - \mathbf{D})\mathbf{Y}^T \tag{7.20}$$

Hidden Layer The updating of **W** is derived similarly:

$$\frac{\partial \mathcal{E}_{av}}{\partial w_{jk}} = \frac{\partial \mathcal{E}_{av}}{\partial u_i} \frac{\partial u_i}{\partial y_j} \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{jk}} \tag{7.21}$$

With equations 7.2, 7.3 and 7.4 the above is calculated to:

$$\frac{\partial \mathcal{E}_{av}}{\partial w_{jk}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} (u_i(n) - d_i(n)) v_{ij} \varphi'(s_j(n)) x_k(n) \tag{7.22}$$

Also this equation is rewritten without the indices on matrix form:

$$\Delta \mathbf{W} = (\mathbf{V}^T(\mathbf{U} - \mathbf{D}) .* \varphi'(\mathbf{S}))\mathbf{X}^T \tag{7.23}$$

where $.*$ denotes elementwise multiplication. Thus the update of the hidden layer weights is:

$$\mathbf{W}(m+1) = \mathbf{W}(m) - \eta \Delta \mathbf{W} = \mathbf{W}(m) - \eta(\mathbf{V}^T(\mathbf{U} - \mathbf{D}) .* \varphi'(\mathbf{S}))\mathbf{X}^T \tag{7.24}$$

7.3.4 Parameters

There are several parameters that can be adjusted affecting the outcome of the neural net. Some of them are mentioned here. All the indexes described by the lower case letters (i, j, k, m and n) have the maximum value denoted by corresponding Latin capital letters.

J, Number of Hidden Layers With larger J the net can handle more complex situations, with the downside of being more computationally demanding.

M, Number of Epochs An epoch consists of one sequence of forward and backward pass through the net and an update of the weights. M determines the number of epoch iterations. It is necessary to reach a certain level, for instance until the error signal passes a certain limit. But, with too many epochs there is a risk of overfitting, which means that the neural network adjusts too much to the training samples loosing information of the general case.

η, Learning Rate Determines the length of the step taken each epoch towards the optimum value. With increasing η there is a risk that the learning process hurries to much and misses the optimum. But with a too low value there is a risk that we never reach it.

N, Number of Training Examples The number of unique training sets, consisting of input-output signals. More training samples often lead to a more general training set.

Fixed Parameters The number of input and output signals are variables depending on the problem. In this case the input signals are determined by the number of facial features that should be tracked, six facial features with a (x, y)-pair each and the geometrical features gives $K = 19$. The output signals are limited to represent a single screen coordinate (x, y), $I = 2$.

The next chapter will describe the use of neural networks in this thesis and how the implementation is done.

8

IMPLEMENTATION

The implementation has been relatively straightforward, starting from the described theories, though a few areas need some further discussion. This chapter will fill these holes and explain the missing parts.

8.1 Matrices in MATLAB

The reason for using matrix form in section 7.3.3 is because this is how the implementation is done in MATLAB. A description of the shape and appearence of all the relevant signals and parameters is needed.

Input Signals The coordinates of the facial features are collected with the image analysis tools described earlier. These coordinates are put in a column vector:

$$\mathbf{x}_n = \begin{pmatrix} L\_COI_x \\ R\_COI_x \\ L\_COI_y \\ R\_COI_y \\ L\_COE_x \\ R\_COE_x \\ L\_COE_y \\ R\_COE_y \\ L\_NOSE_x \\ R\_NOSE_x \\ L\_NOSE_y \\ R\_NOSE_y \\ \beta \\ \gamma_{L1} \\ \gamma_{L2} \\ \gamma_{R1} \\ \gamma_{R2} \\ diff_L \\ diff_R \end{pmatrix}_n , \mathbf{X} = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \\ \vdots & \vdots & & \vdots \end{pmatrix} \tag{8.1}$$

For instance L_COE_y denotes the y-coordinate of the outer corner of the left eye. $\mathbf{x}_n$ is the input signal to the neural net. With a set of N input signals they are put together in $\mathbf{X}$ according to 8.1 and with bias:

$$\mathbf{X}_{bias} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \\ \vdots & \vdots & & \vdots \end{pmatrix} \tag{8.2}$$

Weight Matrices The sizes of these matrices depend on the number of input signals and the structure of the neural net and have the following appearance.

$$\mathbf{W} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1K} \\ w_{20} & w_{21} & & w_{2K} \\ \vdots & & \ddots & \vdots \\ w_{J0} & w_{J1} & \cdots & w_{JK} \end{pmatrix} \tag{8.3}$$

and

$$\mathbf{V} = \begin{pmatrix} v_{10} & v_{11} & \cdots & v_{1J} \\ v_{20} & v_{21} & & v_{2J} \\ \vdots & & \ddots & \vdots \\ v_{I0} & v_{I1} & \cdots & v_{IJ} \end{pmatrix} \tag{8.4}$$

We can specify even more since the fixed parameters I and K equals 2 and 19 respectively. The first column in $\mathbf{W}$ and $\mathbf{V}$ relates to the introduction of bias.

Output Signals To every input signal $\mathbf{x}_n$ there is a corresponding key signal $(d_{xn}\ d_{yn})^T$. They are placed in matrix $\mathbf{D}$.

$$\mathbf{D} = \begin{pmatrix} d_{x1} & d_{x2} & \cdots & d_{xN} \\ d_{y1} & d_{y2} & \cdots & d_{yN} \end{pmatrix} \tag{8.5}$$

The output from the hidden layer is $\mathbf{Y} = \varphi(\mathbf{S})$, where $\mathbf{S} = \mathbf{W} \cdot \mathbf{X}_{bias}$. Both $\mathbf{Y}$ and $\mathbf{S}$ has the size $J\mathrm{x}N$.

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & & s_{2N} \\ \vdots & & \ddots & \vdots \\ s_{J1} & s_{J2} & \cdots & s_{JN} \end{pmatrix} \tag{8.6}$$

Each element in $\mathbf{Y}$ is

$$y_{jn} = \varphi(s_{jn}) \tag{8.7}$$

Adding the bias on the output layer, **U** equals

$$\mathbf{U} = \begin{pmatrix} u_{x1} & u_{x2} & & u_{xN} \\ u_{y1} & u_{y2} & \cdots & u_{yN} \end{pmatrix} = \mathbf{V} \cdot \mathbf{Y}_{bias} =$$

$$= \begin{pmatrix} v_{10} & v_{11} & \cdots & v_{1J} \\ v_{20} & v_{21} & & v_{2J} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & \cdots & 1 \\ y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & & y_{2N} \\ \vdots & & \ddots & \vdots \\ y_{J1} & y_{J2} & \cdots & y_{JN} \end{pmatrix} \tag{8.8}$$

Since $\varphi(\mathbf{S}) = \tanh(\mathbf{S})$, the derivative is

$$\varphi'(\mathbf{S}) = \frac{\partial}{\partial \mathbf{S}} \tanh(\mathbf{S}) = 1 - \tanh^2(\mathbf{S}) = 1 - \mathbf{Y}.\hat{}2 \tag{8.9}$$

where .^2 denotes raising elementwise.

Training Mode The training can be done in two modes: sequential or batch. In sequential mode the updating of the weights is done after each training example. Consequently batch mode updates the weights after presentation of all training examples. The chosen mode is batch training, mostly because it is easier to implement and faster.

Activation Function If the neural net shall distinguish the input into different classes, a possible setup is to have just as many output neurons as classes, with each neuron indexing a specific class. An option is then to have an activation function on the output layer telling whether or not the input belongs to its class.

However this is not the case here so the output signals can linearly represent any of the coordinates on the screen.

Coordinates The facial features are found with various image analysis techniques, as described earlier. They are next transformed from the MATLAB coordinate system (ξ_1, ξ_2)[1] to a normalized coordinate system, ranging both positive and negative values, $(x, y) \in [-1, 1]$. The key, **D**, is transformed similarly.

8.2 Training the Neural Net

The training is done with a set of training samples. A training sample consists of extracted features from an image of the user and an associated screen point, described with two coordinates. Test pads are created showing these coordinates that will be gazed at, an example can be seen in figure 8.1.

Snapshots of the user is taken when looking at the screen points. Totally 500 images of the user, gazing at approximately 90 different screen points, are used. The sequences are varied in head motion; when the user only gazes with the eyes, (not

[1]The MATLAB coordinate system indexes with positive values from $(1, 1)$ up to corresponding image size (ξ_{N_1}, ξ_{N_2}).

moving the head) and when looking with relatively fixed eyeball positions, just moving the head.

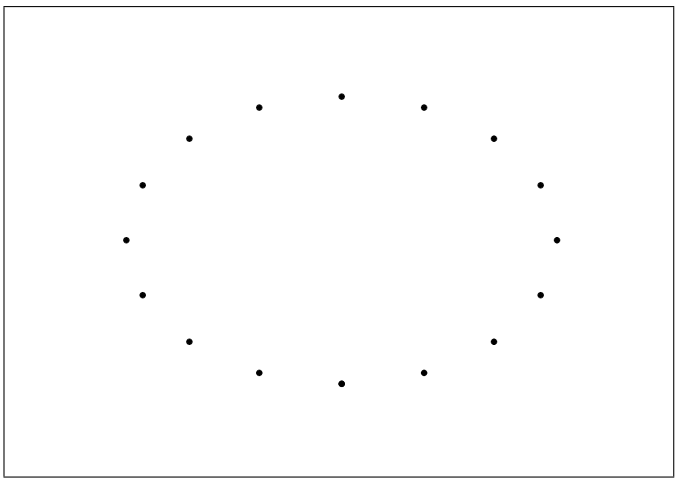

Figure 8.1: An example of a test pad, including 16 gaze points. Black stars indicates screen coordinates gazed at.

8.3 . . . and Testing it

The testing is done with a different set of face images then that the net has been trained with. Otherwise it follows the same structure.

First, a short explanation on how the result images should be interpreted. The white stars still indicates points gazed at and the small white circles, with varying connected lines, shows a simulated result from the gaze tracker.

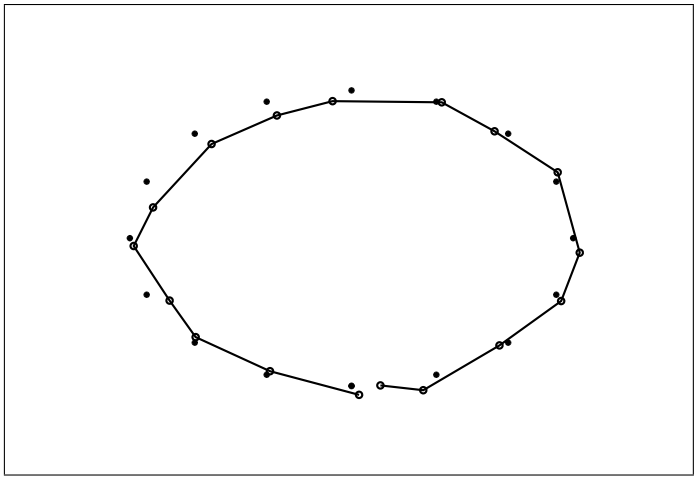

Figure 8.2: Stars indicates points gazed at and the solid line with circles shows the result from the gaze tracker. Note that the image does not show actual results.

8.3.1 Accuracy

To know how well the system works, a measure of the accuracy is needed. The distance between the user and the screen, L, is for the collected image sequences approximately 0.5 meter. $\bar{\rho}$ describes the average error in the gaze tracking results for the present image sequence. An average deviation, ς, is then calculated for this sequence.

$$\varsigma = \frac{180}{\pi} \tan^{-1}(\frac{\bar{\rho}}{L}) \tag{8.10}$$

ρ is translated from pixels to the metric system. ς has a few uncertainties and is therefore only stated with one significant digit, integer degrees.

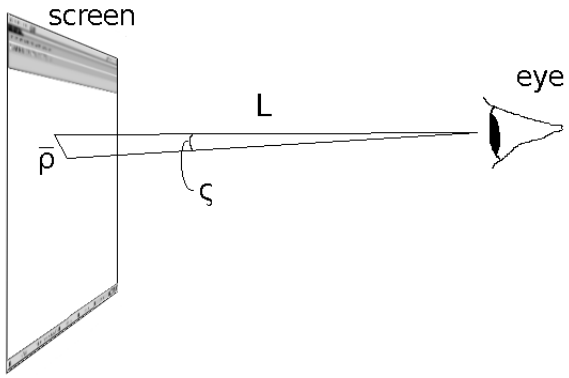

Figure 8.3: Schematic image showing how the angular deviation is estimated.

9

RESULT AND DISCUSSION

This chapter will discuss the results from the implementation of the gaze tracker. It will basically follow the same framing as the thesis so far. First the tracking, then the learning part and finally the gaze tracking in general.

9.1 Facial Feature Detection

Here is a short description of how well the different functions have worked.

9.1.1 Face Finder

It can cope with rather complex backgrounds, but since it is color-based, a skin-color-like background like various shades of red will impair the face finder. As mentioned earlier a few different color spaces was tried. All of them were quite easy to implement, but the Cr-component in the YCrCb-space was found to be most reliable.

Besides the 420 test images of the author, the face finder has been tried on five other Caucasian faces. It has worked sufficiently on all these faces. The face finder has only been tested on Caucasian skin and cannot be reliably used on other skin types. See examples in figure 9.1.

9.1.2 Eye Detection

The detection of COI has also worked satisfyingly. However it would be interesting to know the exactness of the result. How often is it correct, or how many pixels off is it from the true COI?

This is hard to answer, since this evaluation has to be done visually. 20 randomly chosen result images have been looked at, all of these are correct to the extent that the COIs are situated in the iris. The exactness is difficult to establish, but the error varies from zero to three pixels. The diameter of the iris is approximately 20 pixels.

Examples of eye and iris detection results can be seen in figures 9.2 and 9.3.

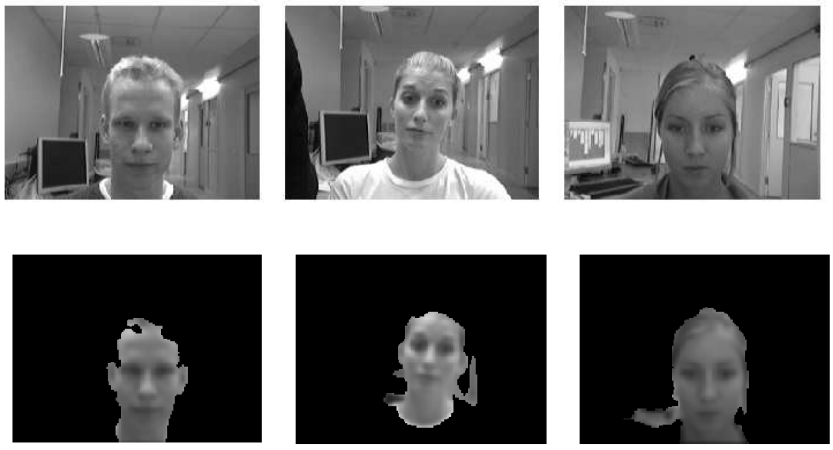

Figure 9.1: Examples of face finder results, in color.

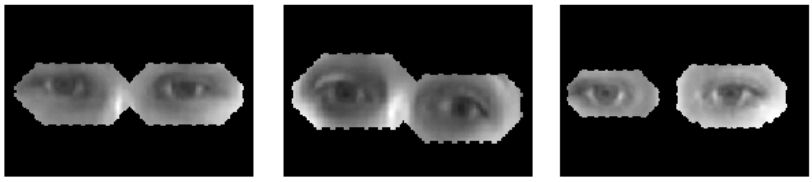

Figure 9.2: Examples of eye detection results.

9.1.3 Detecting Corner of Eye

Only searching for the COE along the line between the eyes naturally restricts the result. But since no other method has worked better this is the chosen one.

The major restriction in finding the COEs is the resolution and/or image quality. In some images it is difficult to visually determine the COE. In other there are disturbing effects from strong light sources, e.g. the sun.

The result is not completely satisfying, it acts sometimes as a rather stochastic COE-finder. However there is only one degree of freedom which limits the extent of the error. See figure 9.3

9.1.4 In Search for Nostrils

Since this algorithm is based on pixel values and shadows in the face it is more sensitive to the lighting conditions.

Despite of that, it seems to work. Maybe it is not the same point on the nose that is

found in all the different head alignments, but the results seems to be reproducible.

In figure 9.3 we can see that the algorithm has failed on the center image, the reason is probably that the user is too far away from the camera combined with the lighting reflections on the cheeks.

Figure 9.3: Examples of results from facial feature detection. All images are zoomed equally.

9.1.5 The Geometrical Features

The head angle β is often quite small, but is accurate. The detection of the COS is impaired if the eyelid shows a bright reflection of light, which will affect the eye angles and the feature *diff*.

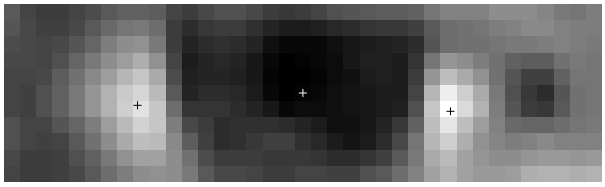

Figure 9.4: Example of results from COS detection.

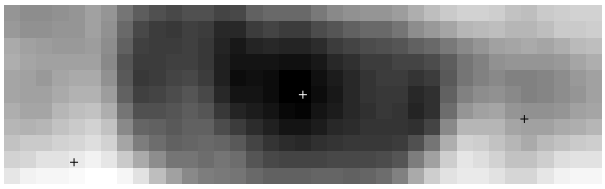

Figure 9.5: Example of result from dysfunctional COS detection, impaired by light reflection on lower eyelid.

9.2 Back-Propagation

As described earlier there are several parameters when handling with a back-propagation net.

9.2.1 Learning Parameters

The basics has already been described, so this section will comment on how the parameters have been adjusted on this problem. Generally there are not one single solution, and the different parameters depend on each other, so similar results can be achieved with a number of setups.

Five perceptrons have been used in the hidden layer, which seems to be sufficient. Give or take a few perceptrons does not afffect the outcome much, provided adjustment of the other parameters.

The number of epochs, M, and η are linked, so to avoid the issues with too large η it is put to a small value (0.02) and M has been adapted accordingly. For results corresponding to an exact value see the following section, but it is in the region of tens of thousands.

A large number of training samples is very importpant, since it is such a complex problem. As an example; increasing the number of training samples, where head position varies laterally, will stabilize the result in that dimension.

9.3 Gaze Tracking

The result is not as good as the existing systems on the market, but it is clear that it works. Some images displaying the result and comments on what affects the outcome will be shown.

The first figure (9.6) illustrates with the solid line the result from the gaze tracker. The order in which the user has looked at the points is evident. The result is in average better vertical than horizontal, so the features describes in a more precise way gaze vertically.

Figure 9.7 shows how the result improves with increasing epochs, M. With to large M the result becomes worse, due to overfitting. this is however not the case in this example.

To aid the training of the net information was introduced that could improve the result horizontally. Therefore the input signals $COI_x - NOSE_x$ were added, which should contain information about head alignment horizontally. Further improvements were made with $COE_x - NOSE_x$. See figure 9.8. Note that no real new information is added here, already existing information is only clarified to the net.

Moving the head in the test sequence, still gazing at the same screen coordinate, shows little effect on the outcome. This means that the system, to some extent, has learned translation invariance.

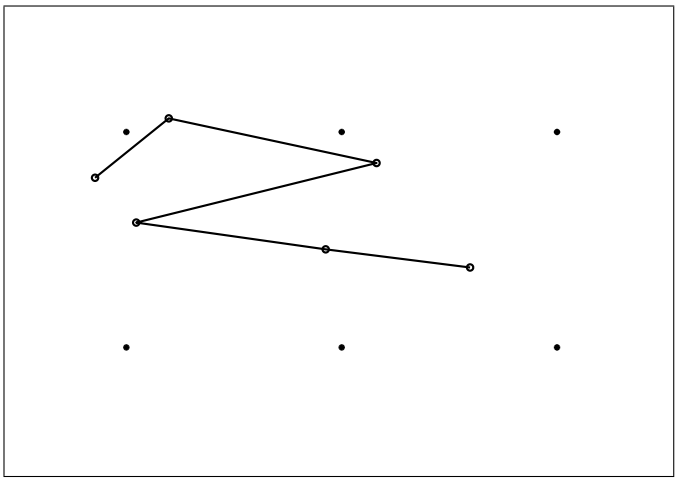

Figure 9.6: Stars indicates points gazed at and the solid line with circles shows the result from the gaze tracker.

One downside on this gaze tracker might be that so many of the input signals are based on the eye. Since there are no restrictions on head alignment there should perhaps be more features describing it.

To see how the nostril detection limits the result, a comparison was done with manually estimating the tip of the nose. This was not done on the entire set of images but on a sufficient amount to see that the result was improved. Hence, a better nose finder is desirable.

9.3.1 Accuracy

The average angular deviation, ς, varies to some extent, depending on the used test sequence. This implies that the result is not that robust and contains some noise. So to try to enhance the result two or more input signals from similar sequences was averaged. It gave, in some cases, slightly better results.

ς varies from two to four degrees depending on test sequence. This must be compared to the built-in limitation, in gaze tracking, of one degree.

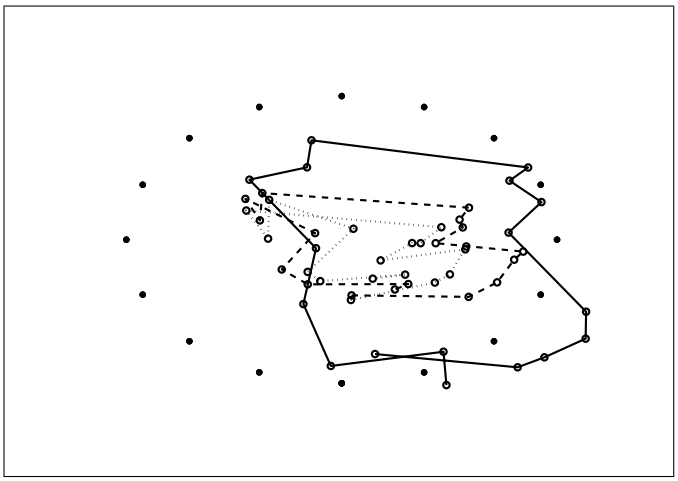

Figure 9.7: Result improvement with increasing epochs. Dotted: $M = 1000$, dashed: $M = 10000$ and solid: $M = 50000$.

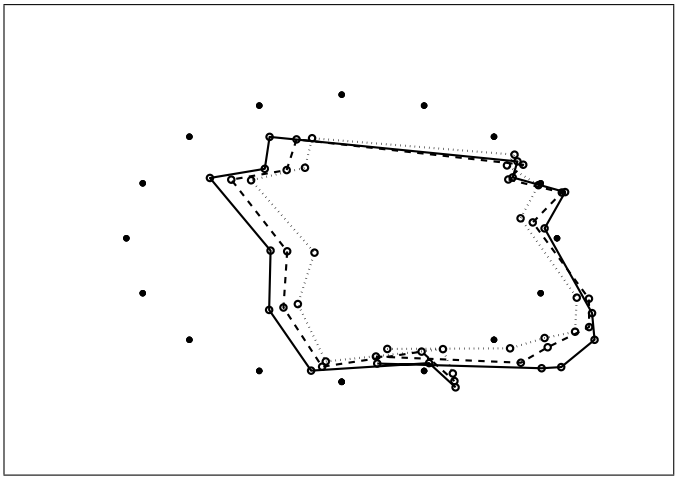

Figure 9.8: Gaze tracking with gradually adding of input signals, improving the result along the x-axis. Dotted: 19 input signals, dashed: $COI_x - NOSE_x$ added and solid: also $COE_x - NOSE_x$.

9.4 The Features Effect on the Outcome

It would be interesting to know how separate features affect the outcome and which features are most important in revealing the gaze.

9.4.1 Correlation

First two definitions have to be made, the coordinates in the image of the user will be referred to as (x_u, y_u) and the coordinates on the computer screen as (x_s, y_s). A reminder to the reader of the appearence of the input signal is probably also necessary.

$$\mathbf{x}_n = \begin{pmatrix} L\_COI_x \\ R\_COI_x \\ L\_COI_y \\ R\_COI_y \\ L\_COE_x \\ R\_COE_x \\ L\_COE_y \\ R\_COE_y \\ L\_NOSE_x \\ R\_NOSE_x \\ L\_NOSE_y \\ R\_NOSE_y \\ \beta \\ \gamma_{L1} \\ \gamma_{L2} \\ \gamma_{R1} \\ \gamma_{R2} \\ \mathit{diff}_L \\ \mathit{diff}_R \end{pmatrix}_n = \begin{pmatrix} x_1 \\ x_2 \\ \\ \vdots \\ \\ x_K \end{pmatrix}_n \tag{9.1}$$

The correlation coefficients in the correlation matrix $\mathbf{C}_X$, are

$$c_{ij} = \frac{cov(\mathit{feature}_i, \mathit{feature}_j)}{\sqrt{V_{ii}V_{jj}}} \tag{9.2}$$

where $\mathit{feature}_i$ is a row vector in $\mathbf{X}$ for each of the inputs and V_{ii} are elements in the covariance matrix

$$\mathbf{cov}_X = (\mathbf{X} - \mu) \cdot (\mathbf{X} - \mu)^T \tag{9.3}$$

μ denotes the mean value of $\mathbf{X}$.

$\mathbf{C}_X$ shows strong correlation between those features described by coordinates on the x_u- and y_u-axis respectively. See figure 9.9. For instance between x_1 and x_2 or x_3 and x_4. There is also a correlation between the inner and outer eye angles respectively.

Something less obvious is the negative correlation between the x_u and y_u coordinates, described by the inputs x_1 to x_{12}. One explanation to this can be that the different coordinate systems used are not exactly aligned, which might cause a skew result.

These correlation coefficients will be used to validate the salience – the importance of each input on the output.

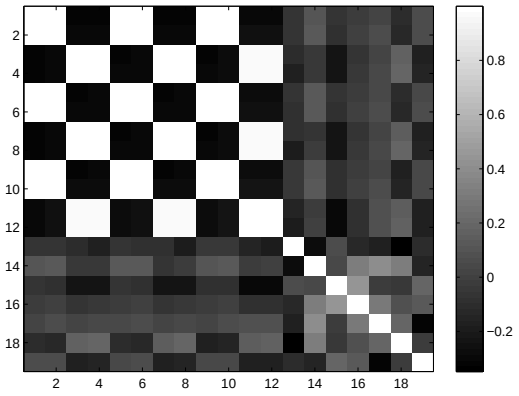

Figure 9.9: The correlation coefficient matrix, $\mathbf{C}_X$, of the input signals.

9.4.2 Principal Component Analysis

An average input signal, $\mathbf{x}_{ave}$, is calculated from all the training examples in $\mathbf{X}$.

$$\mathbf{x}_{ave} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{9.4}$$

To study the directions (in the 19-dimensional space of inputs) with maximum variance of $\mathbf{X}$, PCA (Principal Component Analysis) is performed. The eigenvectors to the covariance matrix, $\mathbf{cov}_X$, are calculated. The two eigenvectors, $\bar{\mathbf{e}}_1$ and $\bar{\mathbf{e}}_2$, with the largest associated eigenvalues indicates the directions with maximum variance. In the 19-dimensional space of inputs a movement along $\bar{\mathbf{e}}_1$ and $\bar{\mathbf{e}}_2$, starting from $\mathbf{x}_{ave}$, is performed, to see how this transportation affect the gaze.

$$\mathbf{x}_{in(1,2)} = \mathbf{x}_{ave} + \tau \cdot \bar{\mathbf{e}}_{(1,2)} \tag{9.5}$$

τ varies in the interval $[-1, 1]$. How the inputs $\mathbf{x}_{in1}$ and $\mathbf{x}_{in2}$ affect the outcome is illustrated in figure 9.10. Studying $\bar{\mathbf{e}}_1$ and $\bar{\mathbf{e}}_2$, see figure 9.11, gives a hint to what affects the outcome in the two directions in figure 9.10. The features described by x_u-coordinates (x_1, x_2, x_5, x_6, x_9 and x_{10}) has a major effect for the direction along the x_s-axis. The y_u-coordinates of the nostrils and the outer eye angles (x_{14} and x_{17}) seems important for the direction dominated by the y_s-axis.

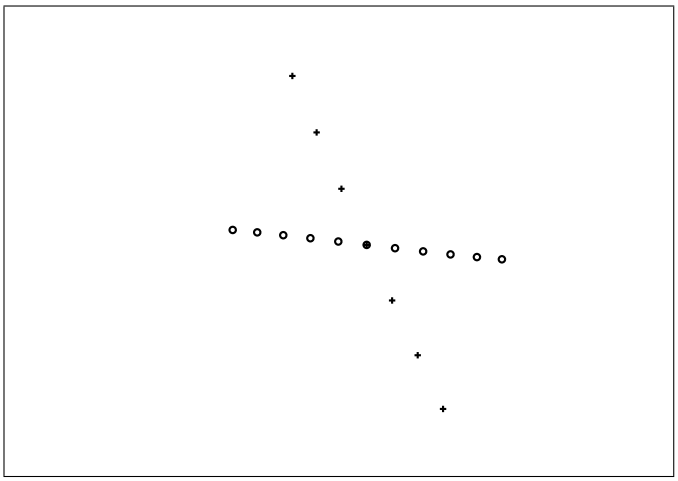

Figure 9.10: Illustration of how the two eigenvectors, $\bar{\mathbf{e}}_1$ and $\bar{\mathbf{e}}_2$, affect the gaze tracking result.

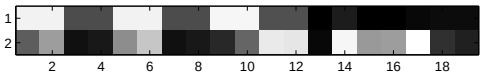

Figure 9.11: Illustration of the two eigenvectors affecting gaze horizontally and vertically. The absolute value is shown, ranging $[0, 1]$, black corresponds to zero.

9.4.3 Salience

The gradient of the output, **U**, from the neural net, is calculated with respect to the input. This makes it possible to see how variations in the input affect the output.

$$\frac{\partial u_i}{\partial x_k} = \sum_j \frac{\partial u_i}{\partial y_j} \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial x_k} = \sum_j v_{ij} \varphi'(\sum_k w_{jk} x_k) w_{jk} \tag{9.6}$$

These gradients are put in the matrix $\mathbf{U}_X$.

Using the entire set of inputs, **X**, and trained weights, **W** and **V**, the result is visible in figure 9.12. This figure can be read as the different inputs (x-axis) importance, on the output (y-axis). This is rather difficult to interpret, and is not so useful, since the net has not been trained to detect variations in one single input. Instead groups of inputs vary equivalently, as shown in figure 9.9.

If the result is weighted with the matrix square root of $\mathbf{C}_X$, $\mathbf{C}_{sqrt}$

$$\mathbf{C}_{sqrt} \cdot \mathbf{C}_{sqrt} = \mathbf{C}_X$$

according to

$$\mathbf{U}_{CX} = \mathbf{C}_{sqrt} \cdot \mathbf{U}_X$$

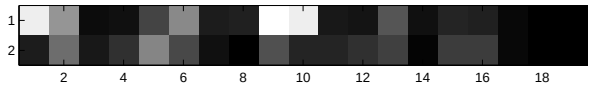

Figure 9.12: Illustration of $\mathbf{U}_X$, the partial derivative of **U** with respect to **X**. The absolute value is shown, ranging $[0, 1]$, black corresponds to zero.

the result should be more reliable regarding salience. This with regards to enable to study the effect of how combinations of inputs affect the gaze. See figure 9.13.

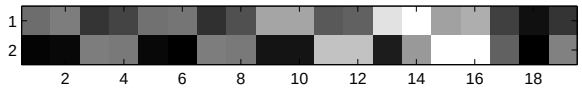

Figure 9.13: Illustration of $\mathbf{U}_{CX}$ the partial derivative of **U** with respect to **X**, weighted with the square root of the correlation coefficients. The absolute value is shown, ranging $[0, 1]$, black corresponds to zero.

This figure illustrates how the inputs described by coordinates on the x_u-axis (x_1, x_2, x_5, x_6, x_9 and x_{10}), are more important for the result on the x_s-axis. That is, affecting the gaze more horizontally than vertically. And vice versa for the inputs describing coordinates on the y_u-axis (x_3, x_4, x_7, x_8, x_{11} and x_{12}). This is visualized as the alternating pattern in the left part of the figure.

The *diff*-features, (x_{18} and x_{19}), have little salience, which might indicate that they are not reliable enough to provide any information for gaze tracking.

The head angle (x_{13}) and the eye angles (x_{14} to x_{17}) are varying to the extent of salience. The inner eye angles (x_{15} and x_{16}) seems to be more useful, maybe there is less disturbance of light reflections, in this shadowier part of the eye.

10

CONCLUSION AND FUTURE WORK

This last chapter will highlight the main conclusions and give proposals for future work.

10.1 Conclusion

Gaze tracking is a complex problem which can be solved in many ways. The most reliable and used systems seem to be those using IR-light. There are many applications where gaze tracking can be utilized, for instance in HCI and usability studies.

The ability to find facial features depends on the image quality. In many cases the result could be improved with higher quality. Of the features used, the COI is found with best result. The reason is probably that in this case a region of pixels is the base. This compared to the COE and nostrils, which are based on certain pixel values.

The neural net has many parameters. They are difficult to grasp and control, and there is always something that can be done to improve the result. Despite that, the result from this gaze tracker is not limited by the net. It is more likely the quality and composition of facial features that limits the outcome.

Most of the features affect the outcome, and it is maybe the quantity of inputs that enables gaze tracking.

An average angular deviation of 2-4 degrees is not that satisfactory, due to large variance. In exceptional cases the deviation amounts to 7-8 degrees, which gives a rather random appearence. Despite that, it works! It is obvious on the test sequences how the user has looked, and there is presumably some application where this accuracy is sufficient.

10.2 Future work

To improve the results, to an acceptable level, the image quality from the web camera must be enhanced. Maybe not that much higher resolution is needed, but instead better contrast and noise conditions. Better image quality would probably enhance

the COI, COE and COS detection and consequently raise the quality of 11 of the 13 facial features. Other gaze tracking systems has a high resolution image of the eye; that might be necessary.

If a better image quality was at hand, other methods to detect the COI and COE could be developed. Enhancement of the image quality could be possible with some preprocessing.

A feature describing head alignment more robustly would be desireable. Either a better nose finder, or some other feature, e.g. the eyebrow or estimating the head alignment with angles.

Regarding the real time application, there is still work to be done. The detection of all the features is possible with a framerate of approximate one frame per second. Sending the coordinates through the net is, in this case, negligible.

Bibliography

[1] Tobii Technology AB. Eye tracking applications, 2005-10-25. http://www.tobii.com.

[2] Shumeet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Advances in Neural Information Processing Systems (NIPS) 6*, number CMU-CS-94-102, January 1994.

[3] David Beymer and Myron Flickner. Eye gaze tracking using an active stereo head. In *Proceedings of the 2003 IEEE Computer Society on Computer Vision and Pattern Recognition (CVPR'03)*, San Jose, CA, USA, June 2003.

[4] Magnus Borga. Neuronnät och lärande system, TBMI26. Lecture notes available on: http://www.imt.liu.se/edu/courses/tbmi26/.

[5] Douglas Chai and Abdesselam Bouzerdoum. A bayesian approach to skin color classification in ycbcr color space. In *IEEE Region Ten Conference (TENCON'2000)*, Kualu Lumpur, Malaysia, September 2000.

[6] R. Z. Cherif, A. Naït-Ali, and M. O. Krebs. An adaptive calibration of an infrared light device used for gaze tracking. In *Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IMTC'02)*, Anchorage, AK, USA, May 2002.

[7] Per-Erik Danielsson, Olle Seger, Maria Magnusson Seger, and Ingemar Ragnemalm. *Bildanalys, TSBB52, 2004 – Kompendium*. Department of Electrical Engineering, Linköpings universitet, 2004.

[8] Gösta H. Granlund and Hans Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publisher, 1995. ISBN 0-7923-9530-1.

[9] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. ISBN 0-13-908385-5.

[10] Rein-Lien Hsu, Mohamed Abdel, and Anil K. Jain. Face detection in color images. Technical Report MSU-CSE-01-7, Department of Computer Science, Michigan State University, East Lansing, Michigan, March 2001.

[11] Robert J. K. Jacob. *Virtual Environments and Advanced Interface Design*, chapter Eye Tracking in Advanced Interface Design. Oxford University Press, New York, 1995.

[12] Björn Johansson. *Low Level Operations and Learning in Computer Vision*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, December 2004. Dissertation No. 912, ISBN 91-85295-93-0.

[13] Björn Johansson. Multiscale curvature detection in computer vision, 2001. Lic. Thesis, Linköpings universitet, SE-581 83 Linköping, Sweden.

[14] Björn Johansson. Rotational symmetries – a quick tutorial, 2004. Tutorial available on: http://www.cvl.isy.liu.se/Research/Filter/rotsym/tutorial_new/.

[15] Shinjiro Kawato and Jun Ohya. Automatic skin-color distribution extraction for face detection and tracking. In *International Conference on Signal Processing, Vol. 2*, Phoenix, Arizona, USA, 2000.

[16] Eun Yi Kim, Sin Kuk Kang, Keechul Jung, and Hang Joon Kim. Eye mouse: mouse implementation using eye tracking. In *International Conference on Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers*, Seoul, South Korea, January 2005.

[17] Hiromi Kobayashi. Evolution of human eye as a device for communication. *Journal of IEICE (The Journal of The Institute of Electronics, Information and Communication Engineers)*, 1999. Abstract available on: http://www.saga-jp.org/coe_abst/kobayashi.htm.

[18] Dinesh Kumar and Eric Poole. Classification of EOG for human computer interface. In *Proceedings of the Second Joint EMBS/BMES Conference*, Houston, TX, USA, October 2002.

[19] Department of Biomedical Engineering, Linköping University, Sweden, 2005-10-25. Image available at: https://www.imt.liu.se/Medical_Images/ppages/-ppage392.html.

[20] Nischal M. Piratla and Anura P. Jayasumana. A neural network based real-time gaze tracker. *J. Netw. Comput. Appl.*, 25(3):179–196, 2002.

[21] Joakim Rydell. Perception-based second generation image coding using variable resolution. Master's thesis, Department of Electrical Engineering, Linköping University, Sweden, 2003.

[22] Eli Saber and Murat Tekalp. Face detection and facial feature extraction using color, shape and symmetry-based cost functions. In *Proceedings of ICPR, '96*, Rochester, NY, USA, 1996.

[23] Sheng-Wen Shih and Jin Liu. A novel approach to 3-D gaze tracking using stereo cameras. In *IEEE Transactions on Systems, Man and Cybernetics, Part B*, Nantou, Taiwan, February 2004.

[24] Gerard J. Tortora and Sandra Reynolds Grabowski. *Principles of Anatomy and Physiology*. John Wiley & Sons, Inc., 10 edition, 2003. ISBN 0-471-41501-4.

[25] Jie Zhu and Jie Yang. Subpixel eye gaze tracking. In *Proceedings of FG '02*, May 2002.

A

MATLAB-scripts

Script	*Description*
eye_track	Main function of the eye tracking.
find_face	Color based face detection, results in a face mask.
find_eyes	Rotational symmetry based coarse eye finder, results in a eye mask.
find_iris	Gives an approximate position of the iris.
find_center_of_iris	Rotational symmetry based fine eye finder, results in iris coordinates.
find_eyeline	Gives the coordinates of the corner of the eyes and head angle, based on the line between the eyes.
find_eyeangles	Gives the geometrical features in the eye based on a zoomed image of the eye.
gradient_est	Estimates the image gradient with partial derivatives of a gaussian.
find_eyepair	Selects the eyes from a few eye candidates.
find_nostrils	Detects the outer nostrils.
coord_transf	Transforms the coordinates of the facial features and computer screen coordinates.
backprop	Learning and testing of the neural net.

Short description of the major MATLAB-scripts.