Active Appearance Models for Gaze Estimation

Masters Thesis

by

Paul Ivan

August 21, 2007

Supervisors:
dr. S. Bhulai (VU)
drs. H. van Kuilenburg (VicarVision)

Vrije Universiteit Amsterdam
Faculty of Sciences
Business Mathematics & Informatics
De Boelelaan 1081a
1081 HV Amsterdam

Abstract

This thesis describes the implementation of and the experimentation with a gaze estimation system based on an approach called the Active Appearance Model (AAM). Active Appearance Models are generative models of a certain visual phenomenon. They can be used to model specific patterns of variability in shape and grey-level appearance, which in turn can be used directly in image interpretation.

Our goal for developing a gaze estimation system is a low-cost user friendly system. Therefore, we wish to design a system that can be used with normal off-the-shelf equipment, like a normal resolution webcam without infrared capabilities and infrared lighting.

Our method uses separate modules for subtracting both the head pose and the eye angles from an image depicting a face. The head pose is necessary information, since we want to allow the user to move its head.

We basically tested two different methods for calculating/estimating the gaze angle. The first one being a direct calculation method based on important positions of parts of the human eye, and a geometric gaze estimation model. The second method uses a Neural Network classifier to estimate the gaze direction based on a representation of the eyes of the user (we tested several representations). This second method proved to be most promising, where the best performing representation was a combination of the landmark vector and the appearance vector of our AAM (of an eye).

We showed that is possible to obtain quite accurate results from our gaze estimation system, although these results are not as accurate as the results of other systems using a infrared camera and/or higher resolution images. Depending on the field of application (where only moderately accurate results are required), our method can be a cheap alternative to other more expensive systems.

Acknowledgements

This thesis was written as the final stage of the master Business Mathematics and Informatics (BMI) at the Vrije Universiteit, Amsterdam. All the research for this project was done during my internship at VicarVision in Amsterdam. Therefore I want to thank Marten den Uyl for giving me the opportunity to work at VicarVision and giving me the time and chance to use their resources and learn from their extensive knowledge of computer vision techniques. Second I want to thank Hans van Kuilenburg for being my main supervisor and for his daily support and constant input during the time of my internship. Finally, my thanks goes out to all the other colleagues for providing a great place to work.

From the Vrije Universiteit I want to thank dr. Sandjai Bhulai for his enthusiastic look at the subject of my thesis and his intensive and quick comments during the writing-process. I would also like want to thank dr. Wojtek Kowalczyk for being the second reader at the Vrije Univeriteit.

And last but not least I want to thank my family for giving me the opportunity to start the master BMI after finishing my previous study. These extra years of studying really helped me in getting to know the interesting parts of Mathematics and Information Technology.

Amsterdam, August 21, 2007,

Paul Ivan

Contents

Chapter 1

Introduction

The eyes are one of the most important senses people have for gathering information. Our eyes constantly provide us with input for further processing. This input can range from learning and entertaining information, like text and images, to every day life perception of the world around us. Normally, we do not appreciate how great an effort our eyes put into our perception processes, and what vast amounts of information they process; we are used to regarding our visual system as "transparent", i.e., that we can concentrate our conscious processes on the concepts that surround us, leaving the intake and basic processing of optical information to our eyes and the visual system.

1.1 What Is Gaze Estimation?

Our eyes are constantly moving and focusing to retrieve the most valuable information. In this light, the eyes are mainly an input organ, but they can also function as an output organ, outputting their gaze direction and appearance. People are quite good at predicting the gaze of another person. From the orientation of a person's head and the angle of the eyes, we can easily give an estimate of the gaze direction of another person.

The gaze direction is defined as a person's current line of sight or point of focus. The focus point is defined as the intersection of the line of sight with the surface of the object (such as the screen) being viewed. Being able to derive the gaze direction automatically could be of great value to the interaction between humans and computers. Knowing the direction of a user's gaze may help a computer to interpret a user's request and possibly enable a computer to ascertain some cognitive states of the user, such as confusion or fatigue.

Several fields of application could benefit of such a gaze estimation system. An example of an useful application of gaze estimation is in the aid for severely disabled people. Here the need for means of communication is acute. Using one's eyes instead of one's arms for computer interaction could overcome the restrictions these people face. A totally different example lies in the field of marketing. By recording the way people use their eyes to examine certain products, for instance, a website, valuable information can be extracted about the best way to design or market such a product.

1.2 VicarVision

The research and development discussed in this thesis have taken place at a company called VicarVision. VicarVision uses the term vicarious perception to illustrate the underlying purpose of their technology, which is to automatically observe the world and translate what is perceived into humanly understandable terms. They create vicarious perception technologies to facilitate analysis and interpretation of image and video data. Their products support or replace human observers in efficient and accurate routine processing of large amounts of visual information.

VicarVision provides efficient and easy to use tools ranging from biometric applications to automated video analysis software. Application areas include: law enforcement, security, e-commerce, enterprise solutions, and of course, image content providing. With the technology, many businesses, organizations and consumers can achieve effective and easy management of visual content and realize straightforward interaction with large image databases.

Currently, VicarVision has a face recognition / emotional expression recognition software application (called The FaceReader). For the future they would like to enhance this application with gaze estimation capabilities. This thesis was written as an initial step towards adding such capabilities.

1.3 Related Work

Many different methods have been developed for tracking the gaze direction of a subject ranging from intrusive techniques, like sensors attached to the face and eye, restrictive video systems requiring a fixed head location, head mounted video systems, to non-intrusive video based techniques. This thesis will focus on non-intrusive techniques because these methods hold the greatest promise for a widely acceptable eye-gaze tracking interface. For a recent overview of alternative methods for eye-gaze tracking, see the review in [14], by Morimoto and Mimica.

The non-intrusive techniques can basically be separated in whether or not infrared lighting and an infrared camera is used. Infrared light along the optical axis results in an easily detectable bright iris. The pupil reflects almost all received infrared light back to the camera, producing a bright pupil effect. In [15], Ohno et al. present a gaze tracking system using a single camera and on-axis infrared light emitters. The gaze position is computed given the two estimated pupil centers utilizing an eyeball model.

In most infrared gaze tracking systems the relative positions of the reflections from parts of the eye are used to estimate the user's point of focus on a planar surface (e.g., a PC monitor). Several variations to interpolate the gaze from known calibration points have been reported in the literature, like the use of artificial neural networks ([1] and [10]). The use of infrared techniques is widely applied in current commercial gaze trackers. However, infrared video-based systems require high resolution images of the eye to accurately estimate the point of gaze, which explains the use of expensive hardware, such as a zoom-capable camera mounted below the screen. Furthermore, this special hardware can cause discomfort and can restrict the user's movements.

Another approach that has been developed recently detects the head pose separately and uses this information to estimate the gaze direction in 3D. This method has several advantages compared to the infrared technique. Besides cheap hardware requirements (a pair of normal cameras and a PC), tracking is not only restricted to points on a planar object. Since the gaze is tracked in a 3D world, we can also intersect the gaze with other objects of interest as well, provided that those objects are properly registered in the 3D world (i.e., the locations are accurately known). Because of this, the system can be easily modified for interaction purposes. Matsumoto et al. ([13]) used stereo cameras to track the head pose in 3D. To measure the gaze direction, the location of the eyeball center is calculated from the head pose and the pupil center is extracted from the stereo images. The vector that connects the eyeball center and the cornea center is the estimate of the gaze direction.

Ishikawa et al. ([9]) proposed the use of Active Appearance Models (AAM) as a part of their eye tracking system. A 3D AAM is fitted to the users face and tracked by using only a single camera. From the 3D AAM they subtract the eye corners and the head pose. The second step of their system uses black disk template matching and ellipse fitting to estimate the location of the pupil. Similar steps as in [13] are done to measure the 3D gaze vector. Another camera is used to view the scene and by asking the user to look at several points in the world, the relative gaze orientation with respect to the projection of these points in the view-camera image can be interpolated.

Hansen et al. ([7]) used a single low-cost camera and an Active Appearance Model of an eye to subtract information about the eye corners and pupil location from an image. To increase the resolution of the eye in the image they placed a camera close to the eye. Although this is a good method of increasing the accuracy, it restricts user movement since the view angle of the camera is smaller for objects close to the camera.

1.4 Goals

This thesis describes the implementation of and the experimentation with a gaze estimation system based on an approach called the Active Appearance Model (AAM). Active Appearance Models are generative models of a certain visual phenomenon. They can be used to model specific patterns of variability in shape and grey-level appearance, which in turn can be used directly in image interpretation.

Previous research projects have indicated that the AAM provides a good generalization to various lighting / pose conditions and incorporated in the right framework, it becomes a very powerful method of subtracting useful information from the object of interest.

Our goal for developing a gaze estimation system is a low-cost user friendly system. Therefore, we wish to design a system that can be used with normal off-the-shelf equipment, like a normal resolution webcam without infrared capabilities and infrared lighting. These requirements pose challenging constraints on the accuracy of the system and thus we do not expect the system to be as accurate as its counterparts with more advanced equipment setups.

The output of the system should be the point of gaze on a computer screen in front of the user. Therefore, we expect the user to sit in front of the screen at a normal distance (40cm-100cm), and the view direction of the camera to be perpendicular to the screen. In summary, we can now state the following problem definition for our research:

How accurate can a gaze estimation system, mainly based on Active Appearance Models, be using only a single non-infrared webcam?

The previous problem definition is subject to the following constraints.

- The system uses only static images as input data (no video sequences are needed) and these images require no special preliminary treatment (such as manually placing markers, manual re-positioning, or scaling), neither does it require specialized hardware such as infrared or 3D cameras.

- The system can handle images obtained in a large variety of settings without retraining the model. No laboratory conditions are required where people have to sit straight and look into the camera under perfect lighting conditions with a diffuse background.

- The system operates in real time.

Chapter 2 gives a global overview of the gaze estimation system we implemented and the (technical) details of the AAM. As the AAM only describes how to build an eye model from an eye image, a method for gaze estimation or calculation still has to be decided on.

Since we want our system to be non-intrusive and thus wish to allow head movement, we need a method for estimating the head pose of the subject. The method we used for head pose estimation is discussed in Chapter 3. Chapter 4 describes how we have trained an AAM for subtracting the necessary information from the user's eyes. Chapter 5 discusses the methods used to estimate or calculate the gaze direction (which includes both the head pose and the eye angles). In Chapter 6 will we show the quantitative results and the comparative evaluation of the different methods we proposed for the implementation of our gaze estimation system. Finally, a general conclusion of our findings along with some topics for further research will be given in Chapter 7.

Chapter 2

Implementation of the Gaze Estimation System

In this chapter a global overview of the different approaches is given as well as a description of the techniques that form the base of our system.

2.1 Global System Description

Globally, the system can be divided in three different parts. The head pose estimation, the eye angle estimation, and the gaze classification. These three parts can be further subdivided in a couple of different modules. Figure 2.1 shows a flowchart of the different parts and modules. We can define the following modules:

Head Pose

1. **Face Finding**. An input image is scanned for face-like patterns using a template matching method. No further information will be given on the exact implementation details of this part due to software copyright restrictions. In [21] a method very similar to the one used for this project is described. The face finding module estimates the most likely position of a face and does this with reasonable accuracy. But alternatively, any other technique for estimating the position of the face could be used, as long as it results in a uniform prediction of the location of a face.

2. **Face Representation**. We normalize the pixels of a cropped area of the face image found by the face finding module, and use these pixels as the representation of the found face. As a second approach we train

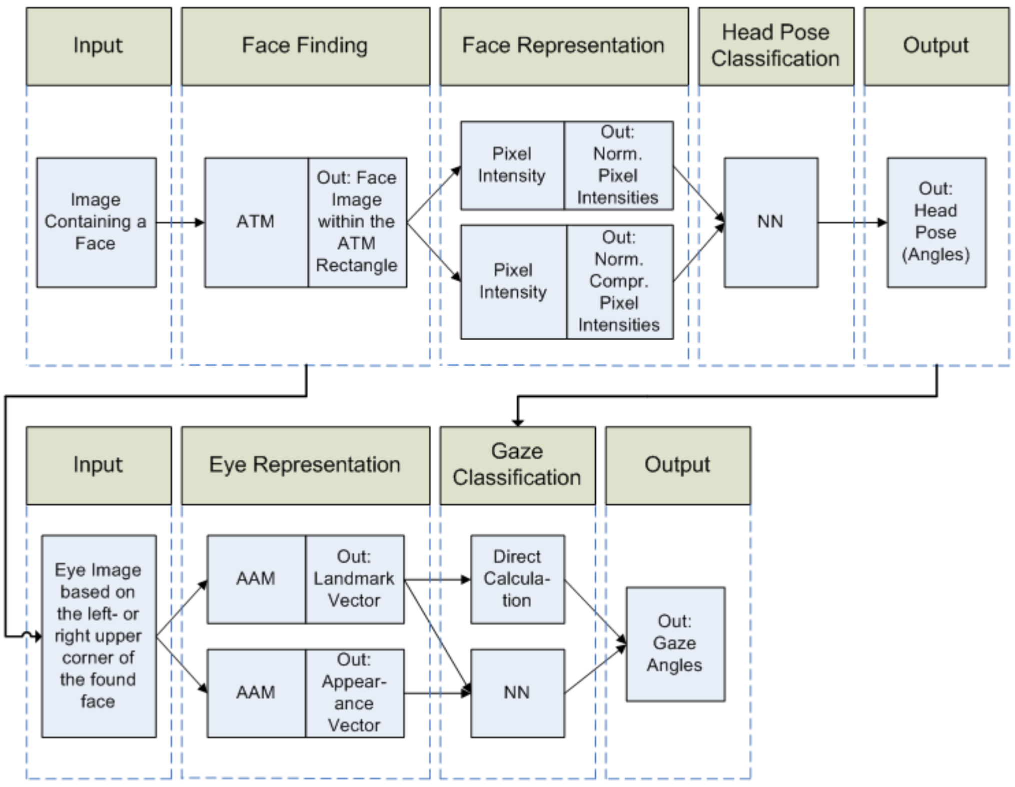

Figure 2.1: Scheme of the different modules

an auto-associative neural network to obtain a compressed vector (see Section 3.2.2 for a further explanation) of the normalized pixels to decrease the dimensionality of the face representation.

3. **Head Pose Classification**. We use a classifier to estimate the head pose from our representation of the face patch.

Eye Angles

1. **Eye Finding**. From the face-finding module we know the position of a face in an image. Using general characteristics of a face we can define two rectangular areas for the location of the right and the left eye. These estimates are used by the AAM as a starting position for the model search procedure and do not have to be totally accurate.

2. **Eye Representation**. We use an Active Appearance Model to create

a model of the detected eye(s). From this AAM we can subtract two different representations of the eyes in an image. Either the landmark vector (defining key positions of an eye) or the appearance vector is used (see the following section for an explanation of AAMs and these vectors).

3. **Eye Angle Classification**. Depending on the representation of the eye we can directly calculate the gaze direction, or use a classifier to estimate the gaze direction.

The most important underlying techniques used in our gaze estimation system are Active Appearance Models. Since this technique is quite complex it will be discussed in the next section

2.2 Active Appearance Models

The Active Appearance Model, as described by Cootes, Taylor, and Edwards (see, [3] and [4]) requires a combination of statistical shape and texture models to form a combined appearance model. The combined appearance model is then trained with a set of example images to form a statistical model of the variation in these example images. After training the statistical model, it can then be used to generate new appearances (consisting of a shape and a texture) resembling the appearances in the example images or new images that can be interpreted using the Appearance Search Algorithm. This chapter will describe these models in detail, mostly following the work of [3], [4], and [23].

2.2.1 Statistical Shape Model

The AAMs require a training set of labeled images (in our case images of eyes), where key landmark points are marked on each example object. The statistical shape model is used to represent objects in images. A shape is described by a set of n points (landmark points). The goal of the statistical shape model is to derive a model which allows us to both analyze new shapes and to synthesize shapes similar to those in the training set. The training set is generated by manual annotation of a set of training images.

With such a set of landmark points we can generate a statistical model of shape variation. The landmark points on a single object describe the shape of that object. We align all s sets of landmark points into a common coordinate frame and represent each by a vector $\mathbf{x}$. We would like to have a parametrized model M of the form $\mathbf{x} = M(\mathbf{b})$, where $\mathbf{b}$ is a vector of the parameters of the model. To be able to derive such a model we first reduce the dimensionality of the data from $2n$ values to a more manageable size. This is done by applying Principal Component Analysis (PCA). PCA is used to extract the main features of the data, by seeking the direction in the feature space which accounts for the largest amount of variance in the data set with possible correlations between variables. This direction (the first principal component) becomes the first axis of the new feature space. This process is repeated to derive the second principal component, and so on until either all variance is explained in the new feature space or the total explained variance is above a certain threshold (l). This approach is as follows:

1. Compute the mean of the data,

$$\bar{\mathbf{x}} = \frac{1}{s}\sum_{i=1}^{s}\mathbf{x}_i. \tag{2.1}$$

2. Compute the sample covariance of the data[1],

$$S = \frac{1}{s-1}\sum_{i=1}^{s}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \tag{2.2}$$

3. Compute the eigenvectors ϕ_i and the corresponding eigenvalues $\lambda_{s,i}$ of S (sorted such that $\lambda_{s,i} \geq \lambda_{s,i+1}$).

Then, if $\mathbf{P}_s$ contains the l eigenvectors corresponding to the largest eigenvalues, then we can approximate any shape vector $\mathbf{x}$ of the training set, using:

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{P}_s\mathbf{b}_s, \tag{2.3}$$

where $\mathbf{P}_s = (\phi_1|\phi_2|\ldots|\phi_l)$ is an orthogonal matrix (thus $\mathbf{P}_s^T = \mathbf{P}_s^{-1}$ and $\mathbf{P}_s^T\mathbf{P}_s = \mathbf{I}_l$) and $\mathbf{b}_s$ is an l-dimensional vector given by

$$\mathbf{b}_s = \mathbf{P}_s^T(\mathbf{x} - \bar{\mathbf{x}}). \tag{2.4}$$

Now we have the parametrized form, in which the vector $\mathbf{b}_s$ defines the set of parameters of the model. By the use of Principal Component Analysis we have reduced the number of parameters from s to l with $l < s$. Depending on l this can be a significant reduction in dimensionality. By varying the

[1]Note that, since $\mathbf{x}_i$ is a vector, this matrix can be seen as the covariance matrix between the individual (scalar) elements x_j of the vector $\mathbf{x}_i$.

elements of $\mathbf{b}_s$ we can vary the shape. The variance of the i^{th} parameter b_i across the training set is given by $\lambda_{s,i}$. By applying limits of $\pm 3\sqrt{\lambda_{s,i}}$ to the parameters of $\mathbf{b}_s$, we ensure that the shape generated is similar to those in the original training set. The number of parameters in $\mathbf{b}_s$ is defined as the number of modes of variation of the shape model.

2.2.2 Statistical Texture Model

To build a statistical model of the texture (grey-levels of the pixels), we warp each example image so that its control points match the mean shape (using a triangulation algorithm). We then sample the grey-level information $\mathbf{g}_{im}$ from the shape normalized image over the region covered by the mean shape.

To minimize the effect of global lighting, the shape-free patches should be photometrically aligned, or in other words, the shape-free patches should be normalized. This is done by minimizing the sum of squared distances E_g between each texture vector and the mean of the aligned vectors $\bar{\mathbf{g}}$, using offsetting (changing brightness) and scaling (changing the contrast) of the entire shape-free patch:

$$E_g = \sum_{i=1}^{s} |\mathbf{g}^i - \bar{\mathbf{g}}|^2, \tag{2.5}$$

where s is the number of shape vectors and texture vectors and thus the number of images in the training set.

E_g is minimized using the transformation $\mathbf{g}^i = (\mathbf{g}_{im} - \beta\mathbf{1})/\alpha$, where α is the scaling factor and β is the offset.

$$\alpha = \mathbf{g}_{im} \cdot \bar{\mathbf{g}}, \qquad \beta = (\mathbf{g}_{im} \cdot \mathbf{1})/n, \tag{2.6}$$

where n is the number of elements in the vector.

Obtaining the mean of the normalized data is a recursive process, as the normalization is defined in terms of the mean. This can be solved by an iterative algorithm. Use one of the examples as the first estimate of the mean, align the others to it (using 2.6) and re-estimate the mean, calculate E_g and keep iterating between the two until E_g has converged (does not get smaller anymore).

The next step is to apply PCA to the normalized data, in a similar manner as with the shape models. This results in:

$$\mathbf{g} \approx \bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g, \tag{2.7}$$

in which $\mathbf{P}_g$ contains the k eigenvectors corresponding to the largest eigenvalues $\lambda_{g,i}$ and $\mathbf{b}_g$ are the grey-level parameters of the model. The number of parameters are called the number of texture modes.

The elements b_i of $\mathbf{b}_g$ are again bounded by:

$$-3\sqrt{\lambda_{g,i}} \leq b_i \leq 3\sqrt{\lambda_{g,i}}. \tag{2.8}$$

If we represent the normalization parameters α and β in a vector $\mathbf{u} = (\alpha - 1, \beta)^T$, we represent $\mathbf{u}$ as $\mathbf{u} = (u_1, u_2)^T$, and $\mathbf{g} = (\mathbf{g}_{im} - \beta\mathbf{1})/\alpha$, we can state that the transformation from $\mathbf{g}$ to $\mathbf{g}_{im}$ is the following:

$$T_u(\mathbf{g}) = (\mathbf{1} + u_1)\mathbf{g} + u_2\mathbf{1}. \tag{2.9}$$

Now we can generate the texture in the image in the following manner:

$$\mathbf{g}_{im} \approx T_u(\bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g) = (\mathbf{1} + u_1)(\bar{\mathbf{g}} + \mathbf{P}_g\mathbf{b}_g) + u_2\mathbf{1}. \tag{2.10}$$

2.2.3 The Combined Appearance Model

The shape and texture of any example can thus be summarized by the vectors $\mathbf{b}_s$ and $\mathbf{b}_g$. Since there may be correlations between the shape and grey-level variations, we apply a further PCA to the data as follows. For each example we generate the concatenated vector

$$\mathbf{b}_{sg} = \begin{pmatrix} \mathbf{W}_s\mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s\mathbf{P}_s^T(\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T(\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix}, \tag{2.11}$$

where $\mathbf{W}_s$ is a diagonal matrix of weights for each shape parameter, allowing for the difference in units between the shape and grey models. Since $\mathbf{b}_s$ has units of distance and $\mathbf{b}_g$ has units of intensity, they cannot be compared directly. To make $\mathbf{b}_s$ and $\mathbf{b}_g$ commensurate, the effect of varying $\mathbf{b}_g$ on the sample $\mathbf{g}$ must be estimated. This can be done by systematically displacing each element of $\mathbf{b}_s$ from its optimum value on each training example and calculating the corresponding difference in pixel intensities.

A simpler alternative is to set $\mathbf{W}_s = r\mathbf{I}$ where r^2 is the ratio of the total intensity variation to the shape variation (in normalized frames). Note that we already calculated the intensity variation and the shape variation in the form of the eigenvalues $\lambda_{s,i}$ and $\lambda_{g,i}$, of the covariation matrix of the shape vectors and the intensity vectors. Thus:

$$\mathbf{W}_s = \frac{\lambda_g^+}{\lambda_s^+}, \tag{2.12}$$

with,

$$\lambda_g^+ = \sum_{i=1}^{k} (\lambda_{g,i}), \qquad \lambda_s^+ = \sum_{i=1}^{l} (\lambda_{s,i}). \tag{2.13}$$

where, $\lambda_{s,i}$ are the l eigenvalues of the covariance matrix of the shape vector and $\lambda_{s,i}$ are the k eigenvalues of the covariance matrix of the texture vector.

We apply PCA on these vectors, giving a further model:

$$\mathbf{b} = \mathbf{P}_c \mathbf{c}, \tag{2.14}$$

where $\mathbf{Q}$ are the eigenvectors and $\mathbf{c}$ is a vector of appearance parameters controlling both the shape and grey-levels of the model. Since the shape and grey-model parameters have zero mean, $\mathbf{c}$ does too. Now if we define $\mathbf{P}_c$ as:

$$\mathbf{P}_c = \begin{pmatrix} \mathbf{P}_{cs} \\ \mathbf{P}_{cg} \end{pmatrix}. \tag{2.15}$$

We see that the linear nature of the model allows us to express the shape and grey-levels directly as functions of $\mathbf{c}$.

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s^{-1} \mathbf{P}_{cs} \mathbf{c}, \qquad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{P}_{cg} \mathbf{c}. \tag{2.16}$$

An example image can be synthesized for a given $\mathbf{c}$ by generating the shape-free grey-level image from the vector $\mathbf{g}$ and warping it using the control points described by $\mathbf{x}$.

2.2.4 The Active Appearance Search Algorithm

Until now we have discussed the training phase of the appearance model. In this section the Active Appearance Search Algorithm will be discussed. This algorithm allows us to find the parameters of the model, which generate a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation[2].

Interpretation of a previously unseen image is seen as an optimization problem in which the difference between this new image and the model (synthesized) image is minimized.

$$\delta \mathbf{I} = \mathbf{I}_i - \mathbf{I}_m, \tag{2.17}$$

[2]To find a reasonable starting position, often a separate module/application is used, which has a fast way of finding an estimate of the position of a face in an image ([23, p.9])

where, $\mathbf{I}_i$ is the vector of grey-level values in the image and $\mathbf{I}_m$ is the vector of grey-level values for the current model parameters. We wish to minimize $\Delta = |\delta\mathbf{I}|^2$, by varying the model parameters $\mathbf{c}$. This appears to be a difficult high-dimensional optimization problem, but in [3] Cootes et al. pose that the optimal parameter update can be estimated from $\delta\mathbf{I}$. The spatial pattern in $\delta\mathbf{I}$ encodes information about how the model parameters should be changed in order to achieve a better fit. There are basically two parts to the problem:

1. Learning the relationship between $\delta\mathbf{I}$ and the error in the model parameters $\delta\mathbf{c}$,
2. Using this knowledge in an iterative algorithm for minimizing Δ.

The appearance model has one compact parameter vector $\mathbf{c}$, which controls the shape and the texture (in the model frame) according to:

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Q}_s\mathbf{c}, \qquad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{Q}_g\mathbf{c}, \tag{2.18}$$

where

$$\mathbf{Q}_s = \mathbf{P}_s\mathbf{W}_s^{-1}\mathbf{P}_{cs}, \qquad \mathbf{Q}_g = \mathbf{P}_g\mathbf{P}_{cg}, \tag{2.19}$$

and where $\bar{\mathbf{x}}$ is the mean shape, and $\bar{\mathbf{g}}$ is the mean texture in a mean shaped patch.

A shape in the image frame, X, can be generated by applying a suitable transformation to the point $x : X = S_t(x)$. Valid transformations are, scaling (s), an in-plane rotation (θ), and a translation (l_x, l_y). If for linearity we represent the scaling and rotation as (s_x, s_y) where $s_x = (s\cos\theta - 1)$ and $s_y = s\sin\theta$, then the pose parameter vector $\mathbf{t} = (s_x, s_y, l_x, l_y)^T$ is zero for the identity transformation and $S_{t+\delta t(x)} \approx S_t(S_{\delta t}(x))$. Now, in homogeneous co-ordinates, $\mathbf{t}$ corresponds to the transformation matrix:

$$\mathbf{S}_t = \begin{pmatrix} 1+s_x & -s_y & l_x \\ s_y & 1+s_x & l_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.20}$$

For the AAM we must represent small changes in pose using a vector $\delta\mathbf{t}$. This is to allow us to predict small pose changes using a linear regression model of the form $\delta\mathbf{t} = \mathbf{Rg}$. For linearity, the zero vector should indicate no change, and the pose change should be approximately linear in the vector parameters. This is satisfied by the above parameterisation. The AAM algorithm requires us to find the pose parameters $\mathbf{t}'$ of the transformation obtained by first applying the small change given by $\delta\mathbf{t}$, then the pose transform is given by $\mathbf{t}$. Thus, find $\mathbf{t}'$ so that $S_{t'}(x) = S_t(S_{\delta t}(x))$. Now it can be shown that for small changes, $S_{\delta t_1}(S_{\delta t_2}(x)) \approx S_{(\delta t_1 + \delta t_2)}(x)$, see Appendix D of [4].

From the appearance model parameters $\mathbf{c}$ and shape transformation parameters $\mathbf{t}$, we get the position of the model points in the image frame X. This gives the shape of the image patch to be represented by the model. During the matching phase we sample the pixels in this region of the image, $\mathbf{g}_{image}$, and project into the texture model frame, $\mathbf{g}_s = T_u^{-1}(\mathbf{g}_{image})$, with T_u from 2.9. The current model texture is given by $\mathbf{g}_m = \bar{\mathbf{g}} + \mathbf{Q}_g\mathbf{c}$. Then, the current difference between model and image in the normalized texture frame is then:

$$r(\mathbf{p}) = \mathbf{g}_s - \mathbf{g}_m, \tag{2.21}$$

where $\mathbf{p}$ are the parameters of the model, $\mathbf{p}^T = (\mathbf{c}^T|\mathbf{t}^T|\mathbf{u}^T)$. A scalar measure of difference is the sum of squares of elements of r, $E(\mathbf{p}) = r(\mathbf{p})^T r(\mathbf{p})$. A first order Taylor expansion of 2.21 gives,

$$r(\mathbf{p} + \delta\mathbf{p}) = r(\mathbf{p}) + \frac{\partial r}{\partial \mathbf{p}}\delta\mathbf{p}, \tag{2.22}$$

where the ij^{th} element of matrix $\frac{\partial r}{\partial p}$ is $\frac{dr_i}{dp_j}$.

Suppose during matching our current residual is r. We wish to choose $\delta\mathbf{p}$ so as to minimize $|r(\mathbf{p} + \delta\mathbf{p})|^2$. By equating 2.22 to zero we obtain the RMS (root mean squared) solution.

$$\delta\mathbf{p} = -\mathbf{R}r(\mathbf{p}), \text{ where } \mathbf{R} = \left(\frac{\partial r}{\partial \mathbf{p}}^T \frac{\partial r}{\partial \mathbf{p}}\right)^{-1} \frac{\partial r}{\partial \mathbf{p}}^T. \tag{2.23}$$

Normally it would be necessary to recalculate $\frac{\partial r}{\partial \mathbf{p}}$ at every step, which is an expensive operation. However, we assume that since it is being computed in a normalized reference frame, it can be considered approximately fixed. We can thus estimate it once from our training set. We estimate $\frac{\partial r}{\partial \mathbf{p}}$ by numeric differentiation, systematically displacing each parameter from the known optimal value on typical images and computing an average over the training set. Residuals at displacements of differing magnitudes are measured (typically up to 0.5 standard deviations of each parameter) and combined with a Gaussian kernel to smooth them. We then precompute $\mathbf{R}$ and use it in all subsequent searches with the model.

Now, if we computed the matrix $\mathbf{R}$, we can construct an iterative method for solving the optimization problem. Given a current estimate of model parameters $\mathbf{c}$, the pose $\mathbf{t}$, the texture transformation $\mathbf{u}$, and the image sample at the current estimate $\mathbf{g}^{im}$, one step of the iterative matching procedure is as follows:

1. Project the texture sample into the texture model frame using $\mathbf{g}_s = T_u^{-1}(\mathbf{g}_{image})$.
2. Evaluate the error vector, $r(\mathbf{p}) = \mathbf{g}_s - \mathbf{g}_m$, and the current error, $E = |r(\mathbf{p})|^2$.
3. Compute the predicted displacements, $\delta\mathbf{p} = -\mathbf{R}r(\mathbf{p})$.
4. Update the model parameters $\hat{\mathbf{p}} \rightarrow \mathbf{p} + k\delta\mathbf{p}$, where initially $k = 1$.
5. Calculate the new points, $\widehat{X}$ and the model frame texture $\hat{\mathbf{g}}_m$.
6. Sample the image at the new points to obtain $\hat{\mathbf{g}}_{im}$.
7. Calculate a new error vector, $r(\hat{\mathbf{p}}) = T_{\hat{\mathbf{u}}}^{-1}(\hat{\mathbf{g}}_{im}) - \hat{\mathbf{g}}_m$.
8. If $|r(\hat{\mathbf{p}})|^2 < E$, then accept the new estimate (record $\mathbf{p} = \hat{\mathbf{p}}$), otherwise try at $k = 0.5, k = 0.25$, etc.
9. Repeat this procedure until no improvement is made to the error $|r(\mathbf{p})|^2$, and convergence is declared.

Chapter 3

Head Pose Estimation

To be able to estimate or calculate the gaze direction and the point of gaze on the screen it is necessary to develop a head pose estimation/calculation system. Even a small change in the head pose of a person can change the point of gaze drastically (while the eye angles seemingly appear the same, because of the low resolution). We thus need an estimation of the head pose of a person, since we want to allow the user to move its head during a session. This chapter describes the method we used for estimating the head pose of a person.

The head pose estimation system used can be separated in the following 3 steps.

1. Face Finding. We use the face-finding module to accurately find the position of a face in an image. This results in a bounding box around the key features of a face (see Section 3.1).

2. Face Representation. Depending on the approach for estimating the head pose we need a representation of the found face. We tested two different representation methods. Either using the normalized pixels of the found face, or using a compressed form of the normalized pixels.

3. Head Pose Classification. We use a normal feed forward neural network to estimate the head pose using the selected face representation (see Section 3.3).

These steps will be discussed in detail below.

3.1 Face Finding

We use the face-finding module to accurately find the position of a face in an image. This results in a bounding box around the key features of a face and an in-plane rotation value ϕ_z. The face-finding module estimates the most likely position of a face using a deformable template matching method, and it does this with reasonable accuracy. The module is capable of finding a face by scaling, translating, and rotating its (deformable) template over the image. At every location (within certain bounds) a difference vector between the template and pixels of the image is calculated and stored. After scanning the whole image a classifier is used to classify the difference vectors as faces or non-faces. The difference vector with the highest score is selected as the one matching the position of the face in the image.

From the transformations of the deformable template at that position we get the in-plane rotation value ϕ_z. From the resulting bounding-box and the in-plane rotation value we can rotate the face image back to zero in-plane rotation. Therefore, in the next stage, we can focus the head pose classification on classifying the pitch ϕ_x and the yaw ϕ_y. Note that our method of defining the head pose follows the direction the face is turning. With an angle of ϕ_x we mean a horizontal rotation, with ϕ_y we mean a vertical rotation and with ϕ_z we mean an in-plane rotation.

In summary, the face-finding module supplies us with a bounding box around the key-feature of a face with zero in-plane rotation as in Figure 3.1.

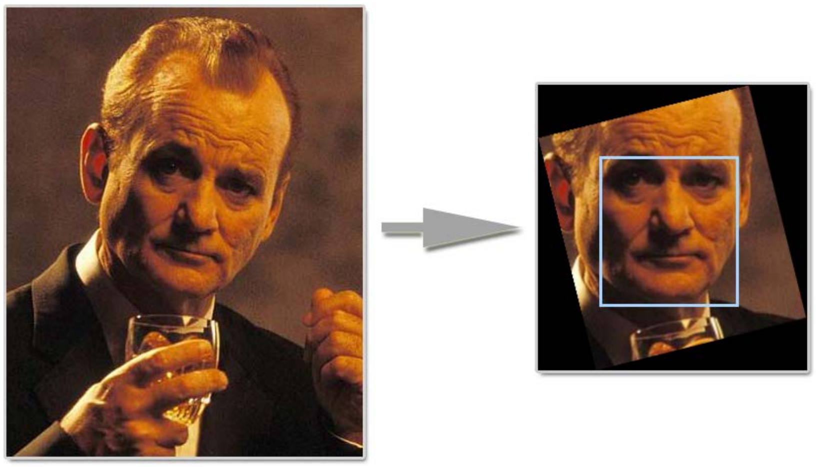

Figure 3.1: The face-finding module accurately estimates the position of the key-feature of a face in an image. From the resulting in-plane rotation parameter, we can rotate the face back to zero in-plane rotation.

3.2 Face Representation

As described earlier we will test two different methods for obtaining a face representation. The first is normalizing the grey-level intensity values of the found face, the second is using an auto-associative neural network to obtain a compressed vector of the normalized grey-level intensity values. From the face-finding module we get the position of the face in an image, this defines a face patch. Since the face module results in a rectangular face patch (and a face is not rectangular) we crop a smaller part from this face patch to remove background information influencing the normalization and classification process.

Experiments have shown that the best results can be achieved by using only a small vertical stroke (from the face patch) around the nose. This could be explained by the fact that that area of the face patch contains the most valuable information concerning the head pose and including a larger area would add noise to the input space. We thus crop the centered 44% of the width and 82% of the height (starting from the top) from the rectangular face patch, as can be seen in the left image in Figure 3.2.

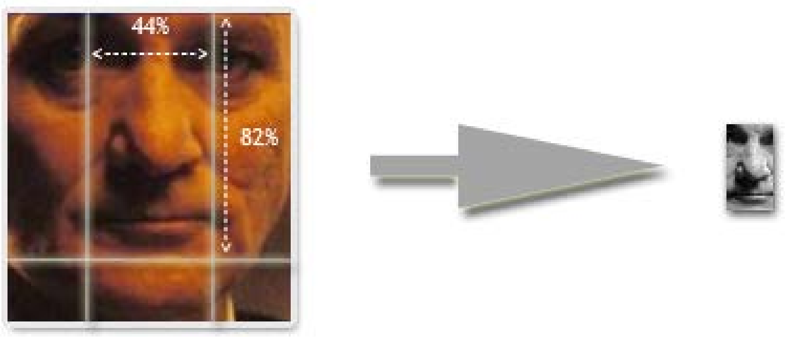

Figure 3.2: In our representation of the face patch we crop the face patch to a smaller size and scaled it to 22x41 pixels.

In the next step we scale the cropped image to a size of 22x41 pixels, as in the right image in Figure 3.2. Now we sample the grey-level intensity values of the cropped and scaled image to form a grey-level vector:

$$\mathbf{p} = (p_1, p_2, p_3, \ldots, p_n), \tag{3.1}$$

where p_i is the grey-level intensity of pixel i and n is the total number of pixels in the image (in our case $n = 22 \cdot 41 = 902$).

3.2.1 Pixel Representation

Now we normalize the pixel vector by applying the following transformation:

$$\hat{p_i} = \frac{p_i - p_{min}}{p_{max} - p_{min}}, \tag{3.2}$$

where p_{min} and p_{max} are, respectively, the minimum and the maximum pixel intensity over the whole cropped face patch and $\hat{p_i}$ is the normalized pixel value. From the $\hat{p_i}$'s we form the normalized pixel vector:

$$\hat{\mathbf{p}} = (\hat{p_1}, \hat{p_2}, \hat{p_3}, \ldots, \hat{p_n}). \tag{3.3}$$

The vector $\hat{\mathbf{p}}$ is now our normalized pixel vector with each value between 0 and 1. This pixel vector is our representation of the face in the image.

3.2.2 Compressed Pixel Representation

In the previous subsection we discussed a simple method for representing the pixels of a face image. A problem with this simple representation is that the dimensionality of such a number of pixels (902) can be very large, making the classification process of these pixels hard and probably showing poor generalization properties.

Therefore, as a second method of representing the pixels in a face image we use a method to reduce the dimensionality of the face patches using the learning abilities of Artificial Neural Networks. If we have an image with N_p pixels, we construct a feed forward neural network with N_p input neurons, N_p output neurons and as many hidden neurons as we wish the reduced dimensionality to be. We then train the network to estimate the same output as the input using the back propagation algorithm. Image compression with neural networks is further discussed in [11] and [5]. After training, the network will learn to make a compact representation of an image in its hidden layer. We can subtract this compact representation in the form of the activation values of its hidden layer and this (compressed) vector can be used as an input vector for classification.

Table 3.1 shows the parameters of the compression network we trained. From the trained network we can subtract compressed images to see if the compressed images still hold information concerning the head pose. Figure 3.3 shows the compressed version of several normalized (and cropped) images. We can see that the compressed images look a lot like the original images. Although they are more blurry, they still contain the head pose information.

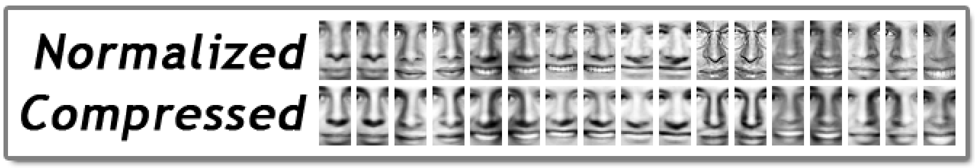

Figure 3.3: Several normalized face patches and their compressed version.

	Compression Network Pixels
network type	Feed forward
training method	Back propagation
network input neurons	902
network hidden neurons	50
network output neurons	902
size of training set	1290
size of stop set	645
learning speed	0.001
training epochs	2000

Table 3.1: Compression Neural Network Parameters

3.3 Head Pose Classification

From the previous section we have a representation of the face depicted on a face image in the form of a normalized pixel intensity-vector or a compressed normalized pixel intensity-vector. We now need to specify a classification method to estimate the head pose. Since our input space consists of vectors of real numbers, that are not likely to be linearly separable, a number of classification algorithms drop out. But there are still a number of possibilities left. A few well-known and promising classifiers are Support Vector Machines (SVMs) (see [24]), k-nearest neighbor classifications, or Artificial Neural Networks (ANNs).

For this thesis we have chosen to use ANNs. ANNs have often been used for classification tasks and pattern recognition and are widely accepted as suitable for those tasks. Other benefits of ANNs are the (relatively) easy way in which they can be implemented and used. For further reading on neural networks and their applicability, see [18], [17], and [2]. Note that for this project we have not conducted a comparative study between ANNs and other classification algorithms. However, we do not expect the differences to be very large, most likely they are in the range of a few (fractions of) percents ([16]), which can be considered insignificant within the scope of this project.

There are many known types of ANNs, but the most obvious choice is probably a simple feed forward neural network (FFNN) with a hidden layer. Such a FFNN can be trained using the backpropagation algorithm. We want to create a network with $N = n$ input neurons (where n stands for the number of pixels in the cropped face patch, see Section 3.2), a hidden layer consisting of as many neurons as needed to correctly classify, but not too many, as we want to retain generalization properties and avoid overfitting. Finally, we need 2 output neurons, since we want to estimate the head pose in the x and y direction.

To be able to train this network we need a large database of head pose annotated images. The collection and gathering of such a database is described in the following section.

3.4 Head Pose Data Collection

Because of the need for a large database of head pose annotated images, and the absence of such a (freely available) database, we need a method to calculate the head pose ourselves. To do this, we used a method described in [6], by Gee and Cipolla. The authors describe a facial model from which the normal vector to a face can be calculated.

This facial model is based on the ratios of four world lengths, L_f, L_n, L_e, L_m. The image quantities (l_f, l_n, l_e and, l_m) corresponding to these ratios are shown in Figure 3.4. The outer corners of the eyes and mouth define a plane, which is called the facial plane. L_f and L_m are measured along the symmetry axis of this plane, while L_e is measured along the direction of symmetry correspondence. L_n, the distance between the nose tip and the nose base, is measured along the normal to the facial plane. The model is based on the distances between relatively stable features: we do not expect the distances to change very much for different facial expressions.

Gee & Cipolla present two methods for estimating the direction of the facial normal. We only use the first method (the 3D-method), since this method proved to work better for near-frontal views of a face. Since our goal is to estimate the point of gaze on a computer screen, we can assume that we are dealing with near-frontal views of a face. This method requires the model ratios $R_m \equiv L_m/L_f$ and $R_n \equiv L_n/L_f$.

Consider a single image of a face in general pose. Assume a camera centered-coordinate system, with the x and the y axes aligned along the horizontal and vertical direction in the image, and the z-axis along the normal

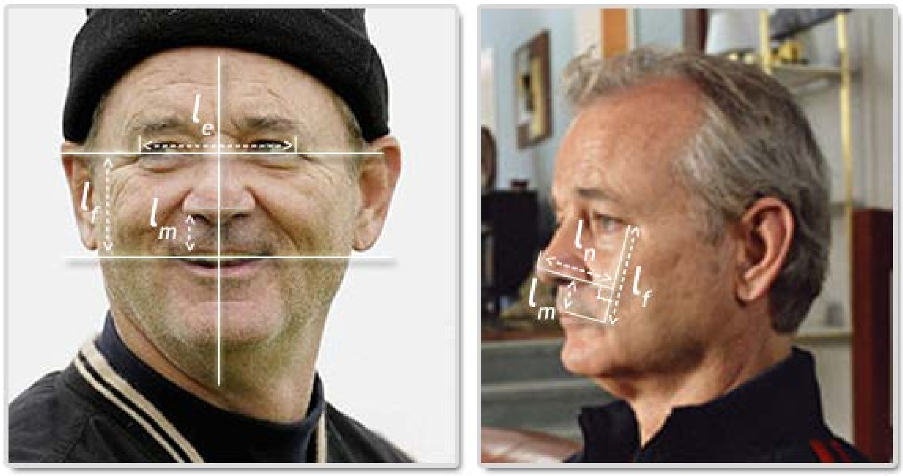

Figure 3.4: Image quantities, l_f, l_n, l_e, l_m.

to the image plane. Assume also that the far corners of the eyes and mouth, and the tip of the nose, have been located in the image. Now it is possible to locate (in the image) the symmetry axis of the facial plane, by finding the midpoints of the eye and mouth points, and connecting those midpoints. Furthermore, using the model ratio R_m, the nose base can be located along this line, since length ratios along the symmetry axis are preserved. Joining the nose base and the nose tip gives immediately the projection of the facial normal in the image, and hence the facial plane's tilt direction (the distance from the camera to the facial plane increases most rapidly in this direction). The tilt is quantified using the angle τ between the imaged normal and the x-axis.

To fully determine the facial normal it is also necessary to find the *slant* σ (see Figure 3.5), the angle between the optical axis and the facial normal in 3D space. Then, in camera-centered coordinates, the facial normal $\mathbf{n}$ is given by

$$\mathbf{n} = [\sin\sigma\cos\tau, \sin\sigma\sin\tau, -\cos\sigma]. \tag{3.4}$$

The slant can be calculated using the model ratio R_m and two measurements in the image: the ratio $l_n : l_f$ and the angle θ. The theory for calculating the slant is presented in Appendix A of [6]. From the facial normal $\mathbf{n}$ we can calculate the head pose. If we denote the facial normal as:

$$\mathbf{n} = [n_x, n_y, n_z], \tag{3.5}$$

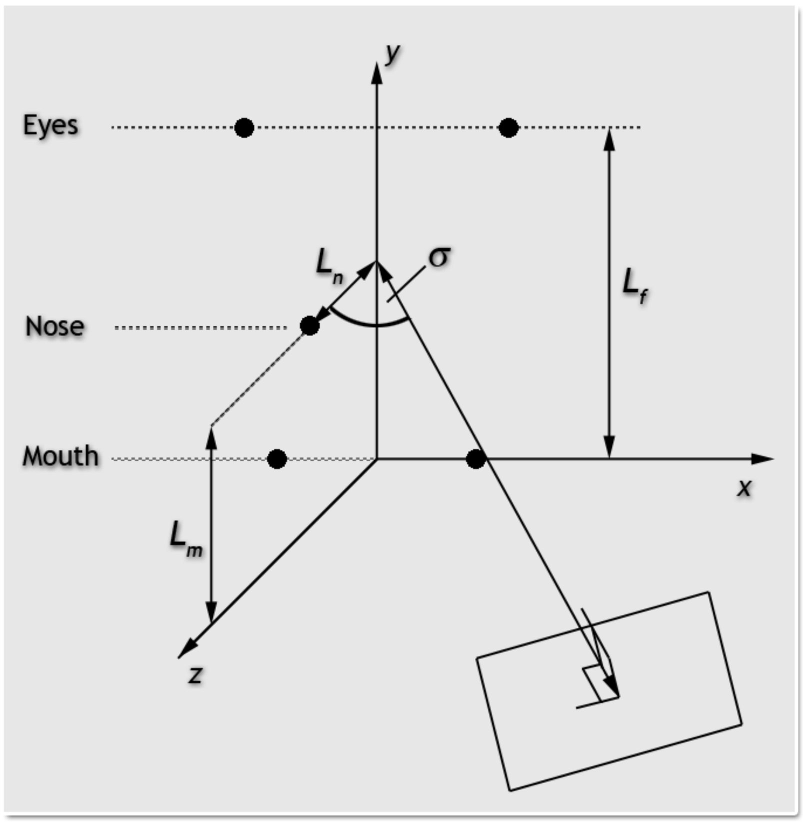

Figure 3.5: The slant (σ), the angle between the optical axis and the facial normal in 3D space.

then

$$\phi_x = \tan^{-1}\left(\frac{n_x}{n_z}\right), \qquad \phi_y = \tan^{-1}\left(\frac{n_y}{n_z}\right), \tag{3.6}$$

where (ϕ_x, ϕ_y) is the head pose. For Bill Murray in Figure 3.6, using model ratios $R_n = 0.6$ and $R_m = 0.4$, this results in a head pose of $\phi_x \approx -8.0°$ and $\phi_y \approx 2.5°$, which is in good agreement with human perception.

Since our classifier will be used to estimate the head pose in the in-plane corrected image, afterwards the normal vector needs to be rotated back to account for the corrected in-plane rotation.

Now, since we calculated the normal vector in the in-plane rotation corrected image, we need to rotate the normal vector back to obtain the real normal-vector. The real normal vector $\hat{\mathbf{n}}$ can be calculated with the help of

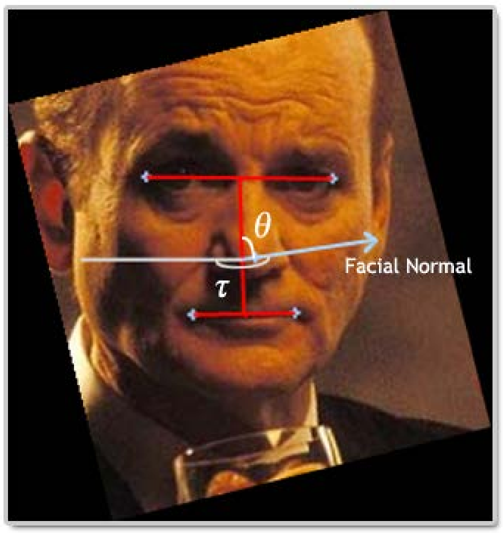

Figure 3.6: Estimating the facial normal

a rotation matrix and the in-plane rotation value ϕ_z.

$$\hat{\mathbf{n}} = \begin{pmatrix} \cos(-\phi_z) & -\sin(-\phi_z) & 0 \\ \sin(-\phi_z) & \cos(-\phi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}. \tag{3.7}$$

3.4.1 Training the Neural networks

To train the feed forward neural network for classifying the head pose, we annotated a set of 1061 images with the necessary landmark points for the method described above (the left and right outer eye corners, the nose, and the left and right outer mouth corners). The set of images is a randomly chosen collection of a larger set of images of celebrities. All the images were horizontally mirrored to increase the amount of training data. An important criterion for the images is that the face-finding module must be able to accurately find the face depicted in the image. From the landmarks we calculated the head pose and constrained the angles between -30.0 and 30.0 degrees. This led to a database of 1858 head pose annotated images with $-30.0 \leq \phi_x \leq 30.0$ and $-30.0 \leq \phi_y \leq 30.0$.

From the training data and the representation of a face we described above, we can now train a feed forward neural network using the back propagation algorithm. To start with, we would like to make a comparison between the two proposed representations of the found face. One possibility is to train a classifier on the normalized pixel vector, as described in Section 3.2.1. An alternative is described in Section 3.2.2, training a neural network which compresses raw image data using a compact hidden layer. The activation values of this hidden layer could be used to train a classification network, which has already been tested and optimized in a previous project, described in [12].

A small comparative experiment has been conducted between these alternatives. The compression network compresses a 902 pixel sampling into a hidden layer of 50 neurons. The settings used for training the neural networks, partly obtained by iterative optimization, are specified in Table 3.2.

	Normalized Pixels	Compressed Normalized Pixels
network type	Feed forward	Feed forward
training method	Back propagation	Back propagation
network input neurons	902	50
network hidden neurons	50	10
network output neurons	2	2
size of training sets	1238	1238
learning speed	0.05	0.05
training epochs	$\approx$ 800	$\approx$ 1000

Table 3.2: Configuration of the head pose network

3.5 Head Pose Estimation Test Results

To compare the alternatives for estimating the head pose we constructed a test set of 500 head pose annotated images, and compared the estimated head pose with the annotated head pose. Figure 3.7 shows how the absolute estimation errors are distributed and Table 3.3 summarizes the results.

	Normalized Pixels	Compressed Normalized Pixels
	X, Y	X, Y
Mean Error (deg)	3.90 , 4.92	4.31 , 5.32
St. Dev (deg)	3.98 , 4.42	4.01 , 4.84

Table 3.3: Results of the different head pose estimation methods

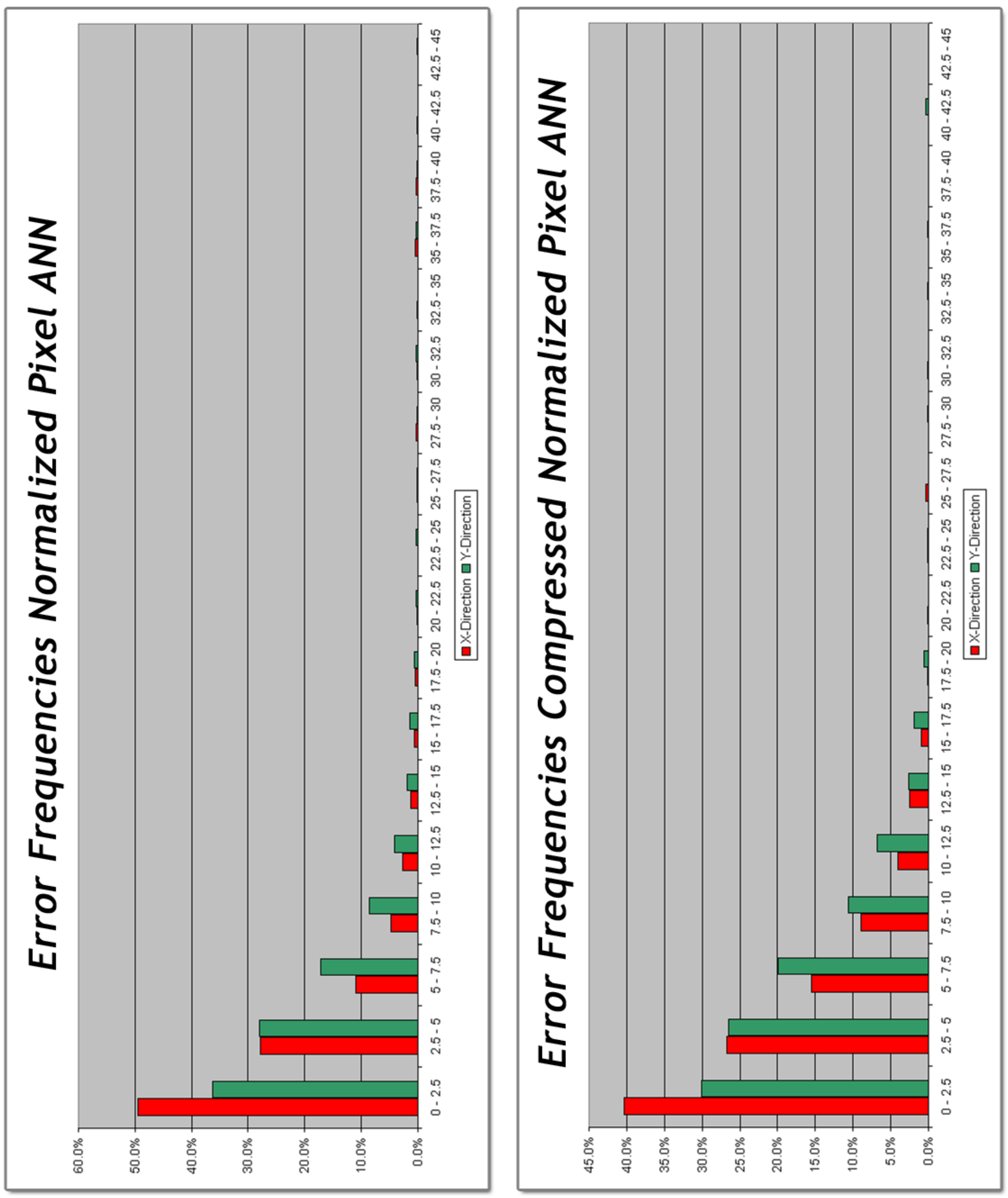

Figure 3.7: Error frequencies of the two head pose estimation ANNs

Clearly the results presented in Table 3.3 are not very good. The ANNs using the normalized and the compressed normalized pixels lead to quite large average errors with large standard deviations. There are a couple of reasons thinkable why the results are disappointing. At first, because of the lack of a correct head pose annotated database, we had to use an estimation method to construct such a database ourselves. Although the method discussed in

Section 3.4 can lead to accurate estimates of a person's head pose, small errors in the annotation (or for example a slightly different definition of the position of the nose in an image) can lead to quite different results. Therefore, it is possible that conflicting data exist in our training set, making it hard to train a network correctly.

Another reason could be that estimating the head pose from our chosen representations is not a classification that is easily learned by an ANN. If we look at Figure 3.7, we can see that although the average errors are quite high, about 77% of the data in the X-direction and almost 64% in the Y-direction has an error smaller than 5 degrees (for the normalized pixels ANN). Table 3.3 and Figure 3.7 show that the normalized pixels ANN performs better than the compressed normalized pixels ANN. Apparently, the ANN with the normalized pixel vector is capable of achieving some sort of compression in its 50 hidden layers, that lead to slightly more accurate results. Therefore, in the following experiments we choose to use the normalized pixel vector ANN as our methodology for estimating the head pose.

3.5.1 Appearance Representation

Before ending this section, we like to introduce another possible representation, only as a test to see if it performs better. This representation is the appearance vector of an AAM of a face as discussed in Section 2.2. We expect that this representation retains information about the head pose, because the AAM used was trained on a training set of face image with different head poses (as well as different expressions, lighting conditions, and ethnicities). This is motivated by the fact that the appearance modes 0 and 5 (see Section 2.2), clearly describe the head pose. Figure 3.8 shows the effect of varying mode 0 and mode 5 of the used AAM between bounds of $-3.0\sqrt{\lambda}$ and $3.0\sqrt{\lambda}$.

Again we constructed an ANN, but now used the appearance vector as input for estimating the head pose. The training set had to be reduced since we could only used the images that led to a good appearance fit. Although the AAM is trained on images with varying head poses, the range of head poses for the AAM to be able to achieve a good appearance fit is approximately between $[-20.0, 20.0]$ degrees. Table 3.4 shows the configuration of this network.

After training the network, we again tested it on the same test set used in Table 3.3. The results are summarized in Figure 3.9 and Table 3.4. This new network performs better in estimating the head pose in the X direction, however it performs practically the same as the network based on the normalized pixels in estimating the head pose in the Y direction.

The above tests showed us that our method for estimating the head pose

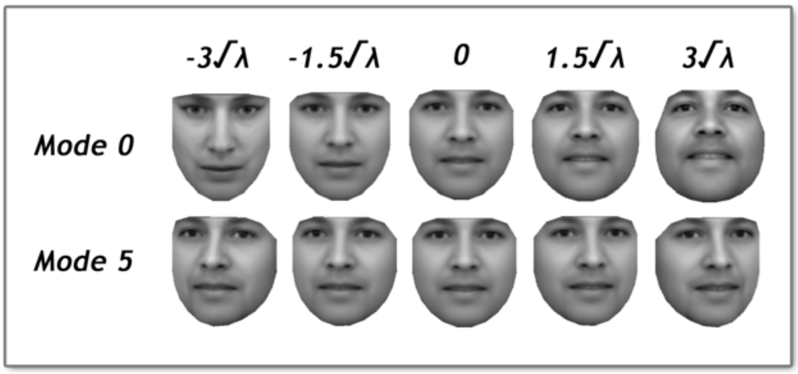

Figure 3.8: Appearance mode 0 and 5 of the AAM used to estimate the head pose

	Appearance Vector
network type	Feed forward
training method	Back propagation
network input neurons	106
network hidden neurons	10
network output neurons	2
size of training sets	625
learning speed	0.05
training epochs	$\approx$ 800

Table 3.4: Configuration of the head pose network based on the appearance vector of a face AAM

	Appearance Vector X, Y
Mean Error (deg)	2.99 , 4.59
St. Dev (deg)	2.83 , 4.30

Table 3.5: Results of the Appearance vector based head pose estimation

does not lead to very accurate results. In search of a better representation we also tested an ANN trained on an appearance vector to estimate the head pose. This raises the question why we did not introduce the appearance

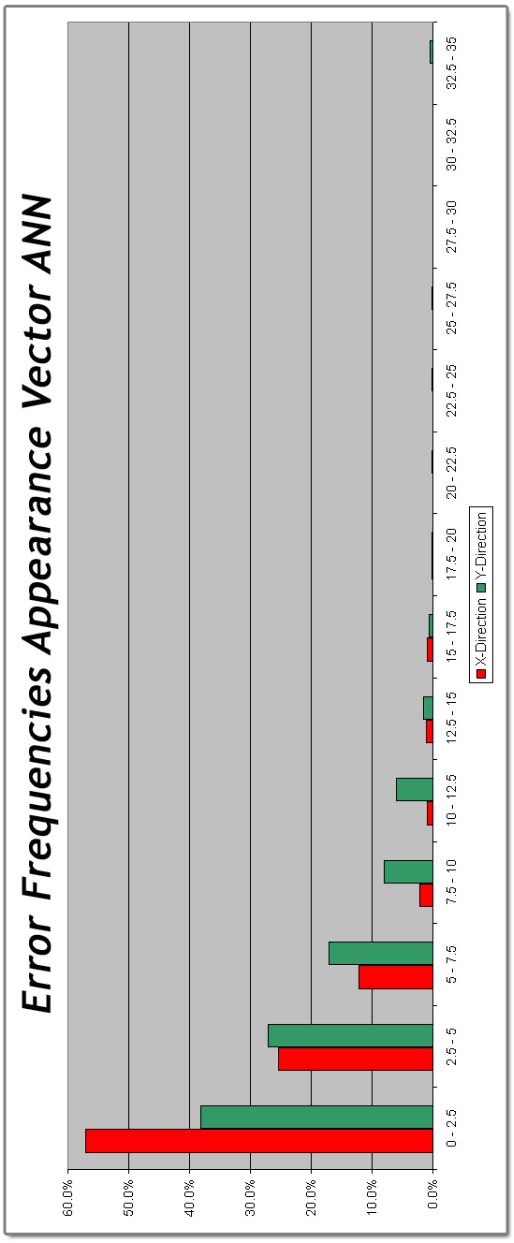

Figure 3.9: Error frequencies of the two head pose estimation ANNs

vector as an initial option for the representation of a face. If we expect the appearance vector to possibly hold more information about the head pose, this could have been a better methodology. The answer to this question is solely based on our goal of building a system that operates in real time. Including a full face appearance fit (with the our current implementation of the AAM and a normal PC) to our system would decrease the frame rate in such a manner that the system cannot operate in real time anymore.

Chapter 4

Eye Modeling

The next step in our gaze estimation system is the estimation of the eye angles. Therefore we need a representation of a person's eyes in an image. The system is supposed to work on normal resolution images taken from a webcam (with a resolution of 640x480 pixels). Since we use the full face of a person for our head pose classification and we expect a certain distance between the camera and the subject (40-100cm), the number of pixels remaining for the eyes is small (about 60x20, at a distance of 50cm). This means that varying the gaze direction of the eyes changes very little pixels in the image. And it is hard to use local feature detectors (like detecting the edges of the iris, or detecting the circle of the pupil) because of the low level of detail. Therefore we chose to use an Active Appearance Model as our representation of an eye.

4.1 Eye Finding

To generate constant and closely resembling fits, Active Appearance Models need a reasonable starting position. To find a reasonable starting position of the eyes in the image, we use the frame found by the face-finding module. Since the face-finding module from Section 3.1 rotates the face image back to zero in-plane rotation we can use the natural characteristics of the face. We record the initial estimate of the left eye as the left-upper third of the face frame and the position of the right eye as the right-upper third of the face frame. This can be seen in Figure 4.1.

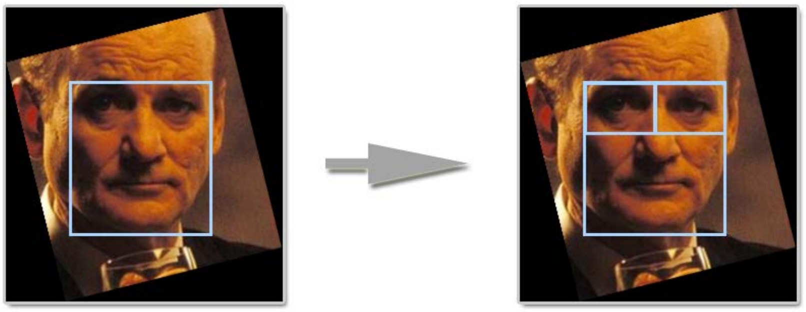

Figure 4.1: The initial positions of both eyes, using the bounding box found by the face-finding module and natural characteristics of the face.

4.2 Eye Representation

We use an Active Appearance model of an eye for our eye representation. For this purpose we tested several landmark annotation models, to find the most suitable landmark annotation model for our goal.

4.2.1 Landmark Annotation Model

Since we need to use the landmark positions predicted by an Active Appearance model, it makes sense to list the possible problems with certain landmark annotation models. When defining a landmark annotation model of an eye the following problems arise:

1. Possible occlusion of the iris by the eyelids.
2. Weak definition of the area surrounding the eyes.

Although an eye is a well-defined class of objects, an eye can have very different appearances for different persons. An important person-specific difference is the amount of the iris that is occluded by the eyelids. Since our goal is to make a multi-person system, the different levels of occlusion make it hard to define a general landmark annotation model for the iris. When building a multi-person AAMs it is important to reduce the introduction of unnecessary variation to a minimum. Because of the occlusion problem we tested leaving the iris out of the annotation model (only annotating the pupil and the eye corners). Although at first hand this seems a good solution, our tests showed that this leads to less accurate Appearance Models (at least for our goal). Namely, when we would leave the iris out of the annotation model, the position of the pupil would solely depend on the landmark position of the pupil. Since the pupil area often yields more than one pixel, this could lead to errors in the estimation of the pupil location.

In our landmark annotation model we used another approach to deal with occlusion of the iris. We systematically annotated the iris with 8 landmark points lying on a perfect ellipse, with the pupil as its center point. Even if the upper and/or lower landmark points of the ellipse turn out to lie on the eyelids, we annotated the iris with an elliptical shape. In the training phase the Active Appearance Model learns this relationship between the individual landmarks of the iris and therefore will always tend to form an ellipse of the landmark points of the iris. Using this method we thus solved the problem with a partly occluded iris and we did not introduce any unnecessary variation in the landmark positions. Because the iris is now part of our landmark annotation model, we can combine the estimation of the pupil location (from its landmark point) with the center of the ellipse, yielding a better (and more constant) estimate of the actual pupil location.

Another problem is the weak definition of the surrounding area of an eye. There are no clear edges in this area, so it is not trivial to define general landmark positions around the eye. We tested leaving the area around the eye out of the Appearance Model, but then another problem arises: the landmark points of the eye corners tend to move along with the pupil. This happens because the eye corners are not that well defined either, if not both sides of the eye corners (the inner eye and the outer eye) are included in the Appearance Model. The approach we used for solving this problem is to add a constrained area (proportional to the size and the center of the eye) to the Appearance Models of the eye. This means, we added four landmark points forming a box around the center of the eye at a scale proportional to the size of the eye. This methods enables the AAM to learn this constrained relationship between the bounding box and the other landmark points, making it easier to predict those locations. Besides this, it reduces the introduction of extra unnecessary variation in the landmark position due to annotation errors.

Figure 4.2 shows our landmark annotation model.

4.2.2 Training Set

The variance in the type of pictures and the type of eyes the AAM will be able to correctly handle after the training has completed is directly related to the images used for training the model. We must thus aim to use a highly diverse collection of images with a variety similar to the variety we want our model to be able to handle. Since our goal is to predict the gaze of a person on a screen in front of him/her we need to train the AAM on eye-images with different gaze directions (looking at different positions on a screen).

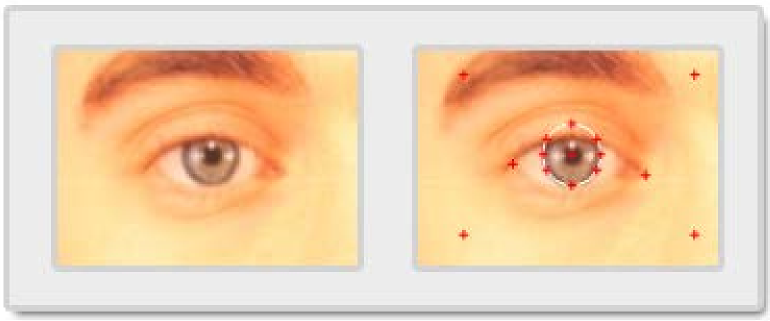

Figure 4.2: Partly occluded iris and our annotation model.

Depending on our method of gaze estimation the AAM should enable us to find the eye corners and the pupil within a face/eye image, or give a good fit on eye images of different persons and of different angles.

To build a generic multi-person Active Appearance Model of an eye, we first collected 9 images of 13 different people with 9 different eye angles (at a resolution of about 400x300 pixels per eye). Since the resolution of these 117 training images is much higher than the actual images the model will be fitted on, we added another collection of 80 eye-images to the training set. These 80 eye-image depict the eyes of different people with different lighting condition at approximately the same resolution as the images the system will be using (mostly looking straight at the camera). Finally, to include as much variation possible due to different gaze angles, we added a set of 103 images with a wide angle variation. In summary, we collected a training set of 300 eye-images with the aim of including as much variation possible due to:

1. Different gaze directions,
2. Different people,
3. Different lighting conditions.

Table 4.1 shows the distribution of the images in our training set. Note that the images were roughly sorted into these categories, and that these categories do not mean that all the images depicting an eye looking to the Center-Left all have the same gaze angle. From the table we can see that the images are quite equally distributed over the whole viewing space of a human eye (within the necessary bounds of looking at a screen), but that only the images depicting an eye looking straight at the camera are overrepresented. This is motivated by our goal of adding as much variation due to personal differences and the fact that in most publicly available images people tend to look at the camera.

Gaze Direction	Images		**Total**
	High Resolution	Low Resolution	
Left-Up	13	24	**37**
Up	13	12	**25**
Right-Up	13	21	**34**
Left-Center	13	21	**34**
Center	13	58	**71**
Right-Center	13	18	**31**
Left-Down	13	9	**22**
Down	13	10	**23**
Right-Down	13	10	**23**
Total	**117**	**183**	**300**

Table 4.1: Distribution of training images for the AAM

4.2.3 Right & Left Eye

Since we want to model both the eyes of the subject of the gaze estimation system, we horizontally mirrored all the images in the training set (the training set only consists of left eye images) to obtain right eye images. Although in real life the eyes of a person are often not perfectly alike (when mirrored), our research has shown that mirroring left eye images to train a right eye AAM has no (or hardly no) effect on the quality of the AAM. Besides the advantage of not having to gather a representative database of right eye images, this method makes sure that the left eye and the right eye are modeled in the same manner. Therefore, the results of the gaze estimation of both eyes separately can be perfectly combined into a gaze estimation of both eyes together.

4.3 Eye AAM Results

From our training set of annotated images we trained an AAM. Figure 4.3 shows the effect of varying the first three modes of appearance of the AAM. Clearly, we can see that the first two modes represent the gaze direction.

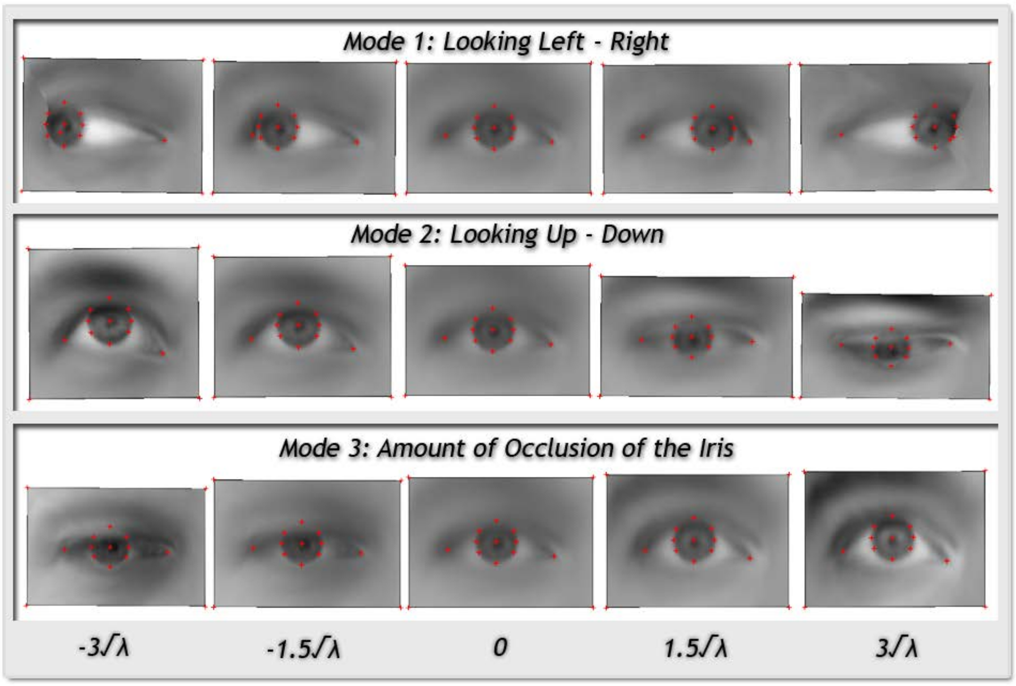
Mode 1: Looking Left - Right
Mode 2: Looking Up - Down
Mode 3: Amount of Occlusion of the Iris
-3√λ
-1.5√λ
0
1.5√λ
3√λ

Figure 4.3: The first three modes of appearance of the AAMs

Varying the first mode represents looking to the left or the right and varying the second mode represents looking up or down. By combining these two modes, the AAM is capable of modeling all necessary eye angles to predict the gaze of a person. The third mode represents a more person-specific characteristic, namely the amount of occlusion of the iris.

The AAM has 16 modes of appearance in total, where the first two modes represent the viewing direction and the rest of the modes represent the variation in the appearance of the eyes of different people.

Figure 4.4 shows the appearance fits of several eye images with different gaze angles. The images show that the AAM is capable of modeling the gaze variation in the different images. If we look at the model images (middle images) we can see that they look a lot like the original images. Although some person-specific details are lost (the images are a bit more blurry), the gaze direction is preserved. This statement is extra motivated if we look at the retrieved landmark positions. The landmark positions of the iris, the pupil, and the eye corners accurately match the positions of those key locations in the original image.

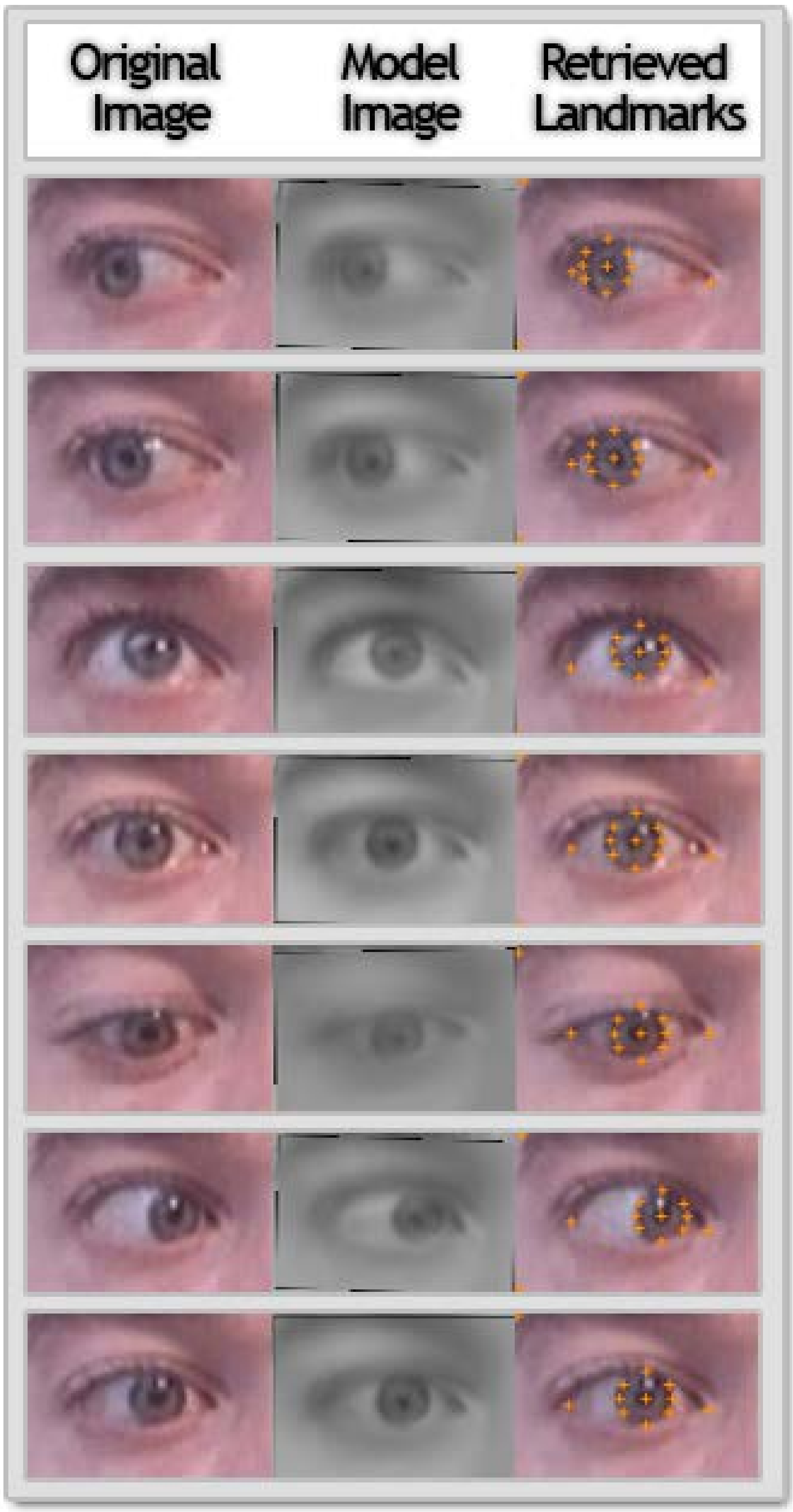

Figure 4.4: The appearance fits of several eye images with different gaze angles

Chapter 5

Gaze Estimation

Once the head pose is estimated and the representation of the eye is made, the gaze direction can be estimated. This will be done in the Gaze Classification module. Depending on the representation of the eye, we tested two methods for gaze classification.

The first one being direct calculation of the gaze based on important positions of parts of the human eye. The second is using some sort of classifier (for example, a Neural Network) to estimate the gaze of a person based on the representation of the eye.

These two methods will be presented here.

5.1 Direct Calculation

In [9] a geometric gaze estimation model is described. This geometrical model assumes that the eyeball is spherical and that the inner and outer eye corners have been estimated. From this information, the algorithm can be described by two steps:

1. Estimate the center, and the radius of the eyeball in the image from the eye corners and the head pose.

2. Estimate the gaze direction from the pupil location, the center and the radius of the eyeball.

The first step requires the following anatomical constants:

- R_0: The radius of the eyeball when the scale of the eye is 1.

- (T_x, T_y): The offset in the image (when the face/eye is frontal and the scale is 1) between the mid-points of the two eye corners and the center of the eyeball.

- L: The depth of the center of the eyeball relative to the plane containing the eye corners.

For now let us assume that these constants are known. Figure 5.1 shows a top down view of the geometrical eye model.

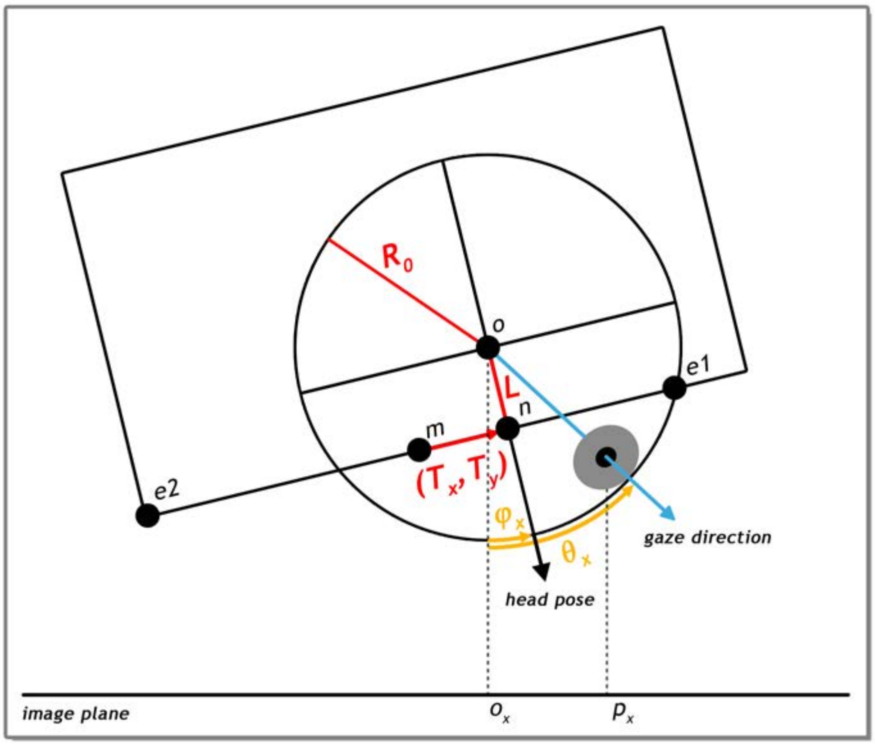

Figure 5.1: Top Down view of the geometrical Eye Model

The center and radius of the eyeball are computed using the following three steps.

1. The mid-points (m_x, m_y) between the inner eye corner $(e1_x, e1_y)$ and outer eye corner $(e2_x, e2_y)$ are computed by:

$$\begin{pmatrix} m_x \\ m_y \end{pmatrix} = \begin{pmatrix} \frac{e1_x + e2_x}{2} \\ \frac{e1_y + e2_y}{2} \end{pmatrix}. \tag{5.1}$$

2. The scale of the face/eye S is computed. There are several ways to do this, one way is to use the foreshorten-corrected distance between the eye corners:

$$S = \frac{\sqrt{(e1_x - e2_x)^2 + (e1_y - e2_y)^2}}{\cos \phi_x}. \tag{5.2}$$

The disadvantage of this approach is that it is very noise sensitive because it is the difference between two points that are very close to each other in the image. Instead, we use another method for estimating the scale. We estimate the scale by dividing the actual size of a person's head by the size of the head found in the image by the face-finding module. This estimate is more reliable because it is calculated over the whole face region.

3. The center of the eyeball (o_x, o_y) is then computed as the mid-point (m_x, m_y) plus two corrections:

$$\begin{pmatrix} o_x \\ o_y \end{pmatrix} = \begin{pmatrix} m_x \\ m_y \end{pmatrix} + S \begin{pmatrix} T_x \cos\phi_x \\ T_y \cos\phi_y \end{pmatrix} + S \cdot L \begin{pmatrix} \sin\phi_x \\ \sin\phi_y \end{pmatrix}. \quad (5.3)$$

The first correction is a foreshortened offset that compensates for the fact that the mid-points of the eye corners are not necessarily the eye center even for a frontal image. The second correction compensates for the fact that the eyeball center does not, in general, lie in the plane of the eye corners. Note that (ϕ_x, ϕ_y) is the head pose.

4. The radius of the eyeball in the image is computed as $R = S \cdot R_0$.

Now if the location of the eye corners, the location of the pupil, and the anatomical constants are known we can compute the gaze direction (θ_x, θ_y) as follows:

$$\begin{pmatrix} \sin\theta_x \\ \sin\theta_y \end{pmatrix} = \begin{pmatrix} \frac{p_x - o_x}{\sqrt{R^2 - (p_y - o_y)^2}} \\ \frac{p_y - o_y}{\sqrt{R^2 - (p_x - o_x)^2}} \end{pmatrix}. \quad (5.4)$$

The Anatomical Constants

In [9] a method for training the anatomical constants in an offline phase is described. Since this results in constant values for R_0, (T_x, T_y), and L it is questionable whether it is necessary to find these constants in a training procedure. Another method would be to take anatomical averages (which can be found in medical literature as can be seen in Figure 5.2) for the values of R_0 and L.

Figure 5.2 shows some parameters of the eye. As the parameters of the eye will vary between models, it is equally true that they will vary from person to person, depending on such factors as age, size, gender, ethnicity, health, etc. The values shown in Figure 5.2 are believed to represent valid data for a typical case involving an average, middle-aged person.

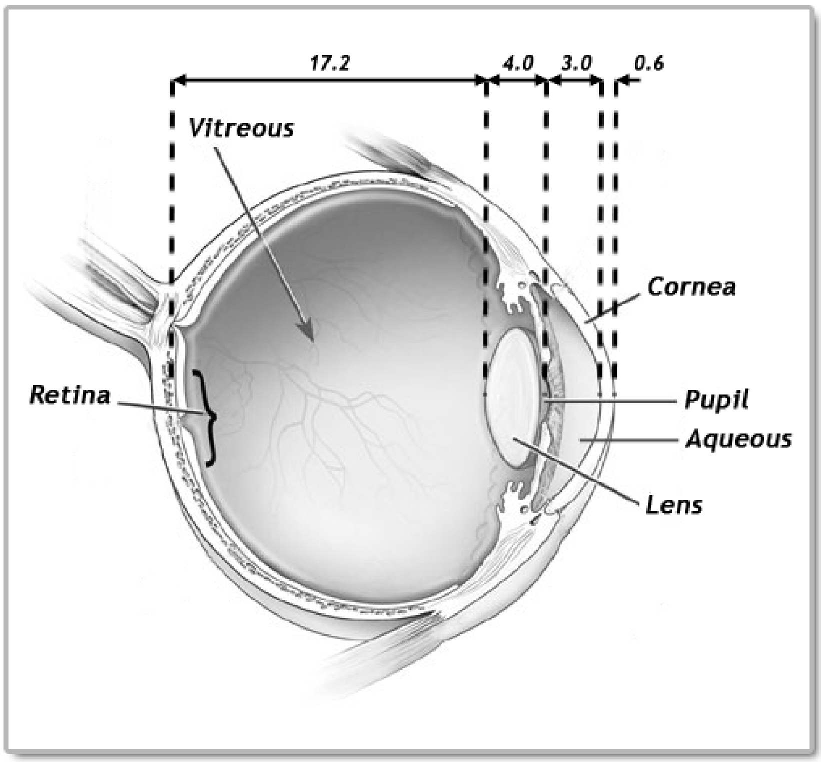

Figure 5.2: The Anatomical constants, a typical case involving an average, middle-aged person. All distances are in millimeters.

From Figure 5.2 we could fix the radius of an eyeball R_0 at 12.4mm, and L, the depth of the center of the eyeball relative to the plane containing the eye corners, at 4.5mm. Next, the offset (T_x, T_y) between the actual eye center in the (X, Y) plane and the difference between the eye corners could be computed/estimated by a simple initial calibration step. The following describes how to calibrate (T_x, T_y):

1. Ask the person using the system to sit upright, directly in front of the camera.

2. Ask the person to look at the lens of the camera, while holding his/her head parallel to the image plane, i.e., with $(\phi_x, \phi_y) = (0, 0)$.

3. Now we know that $o_x = p_x$ and that $(\phi_x, \phi_y) = (0, 0)$. Thus we can calculate (T_x, T_y) by:

$$T_x = \frac{p_x - m_x - SL\sin\phi_x}{S\cos\phi_x} = \frac{p_x - m_x}{S}, \tag{5.5}$$

$$T_y = \frac{p_y - m_y - SL\sin\phi_y}{S\cos\phi_y} = \frac{p_y - m_y}{S}. \tag{5.6}$$

We can conclude this part with a summary of the requirements for the eye tracking system in case of direct calculations of the gaze angles. To be able to compute the gaze directly from the information of the eye tracking system we need the following input variables:

- The position of the eye corners $(e_x^{inner}, e_y^{inner})$ and $(e_x^{outer}, e_y^{outer})$ and the position of the pupil (p_x, p_y).
- The head pose (ϕ_x, ϕ_y).
- The anatomical constants R_0, (T_x, T_y), and L.

The positions of the eye corners and the position of the pupil can be derived from the fitted landmark vector of the AAM. We calculate the position of the pupil as the average of the center of the iris and the pupil landmark.

5.2 Neural Network Classification

Another method of estimating the eye angles is with the use of a classification algorithm like Neural Networks. In [19], Stiefhagel et al. achieved accurate results, using a normalized eye image (its grey-level intensity values) as input for a neural network to predict the gaze direction of a person. A downside of their approach is that they used the neural network to predict the gaze position (on the screen) directly instead of the gaze angle of an eye. This restricts the movement of the user because it does not account for the head pose. Another downside of the approach is their representation of an eye. Since we use an AAM of an eye as the representation of our eye, we possibly have more generic information about the gaze angles than can be expected from the normalized eye images.

A model instance of the AAM is characterized by an appearance vector, containing information about both the shape and the texture of an eye. Assuming that when training the AAM, a reasonable amount of variation caused by different gaze directions was present in the original eye images, the AAM will be able to model the angle information, as we already saw in Section 4.2.2). Since the appearance vector is reasonably compact, it should intuitively be possible to use it directly as input for a classifier. In [23], H. van Kuilenburg already showed that using a classifier directly on the appearance vector[1] to extract useful information (embedded in the training set) can lead to an accurate classifier.

Another approach is using a classifier on the landmark vector to estimate the gaze angles. Since our landmark annotation model is defined at key positions of an eye, the landmark vector contains valuable information regarding the gaze angles (as also follows from Section 5.1). Therefore, using a classifier on the landmark vector could lead to a correct transformation from that vector to the gaze angles, without having to define an eye model as in Section 5.1.

Both these vectors potentially hold enough information to predict the gaze. The appearance vector is a very compact representation of the eye in the image and holds information about the texture and the shape (and thus the gaze direction) of the eye. On the other hand, the landmark vector only holds information about the shape of the eye, and because of our landmark annotation model, it mainly holds information about the position of the iris/pupil within the rest of the eye. Therefore, one could state intuitively that the landmark vector should contain enough information for our classification task and thus might generalize better over different people.

In the following section we will describe how we implemented these two methods.

5.2.1 ANN Eye Model

Before we will discuss the input for our classification algorithm, we have to define how the eye angles (θ_x, θ_y) are measured. We will do this by first defining a reference point from where our eye angles are measured. In our model we need the camera to be in front of the screen with its view direction perpendicular to the screen. Then, if we have an image I of size (I_x, I_y) and the projected location of the camera on the screen is at the point (w_x, w_y), we can calculate our reference point[2] (c_x, c_y) on the screen as follows:

$$\begin{pmatrix} c_x \\ c_y \end{pmatrix} = \begin{pmatrix} w_x \\ w_y \end{pmatrix} - \frac{1}{S} \begin{pmatrix} \frac{1}{2} \cdot I_x - e_x \\ \frac{1}{2} \cdot I_y - e_y \end{pmatrix}, \tag{5.7}$$

where the point (e_x, e_y) is the approximated eye center in the image. This approximated eye center is calculated from the face-finding module, as in Figure 5.3.

[1]In [23] an Artificial Neural Network was successfully trained on the appearance vector of a face AAM to classify the expressions of a person in a face image.

[2]The translation from the projected camera location, which is proportional to the translation of the eyes from the center in the image.

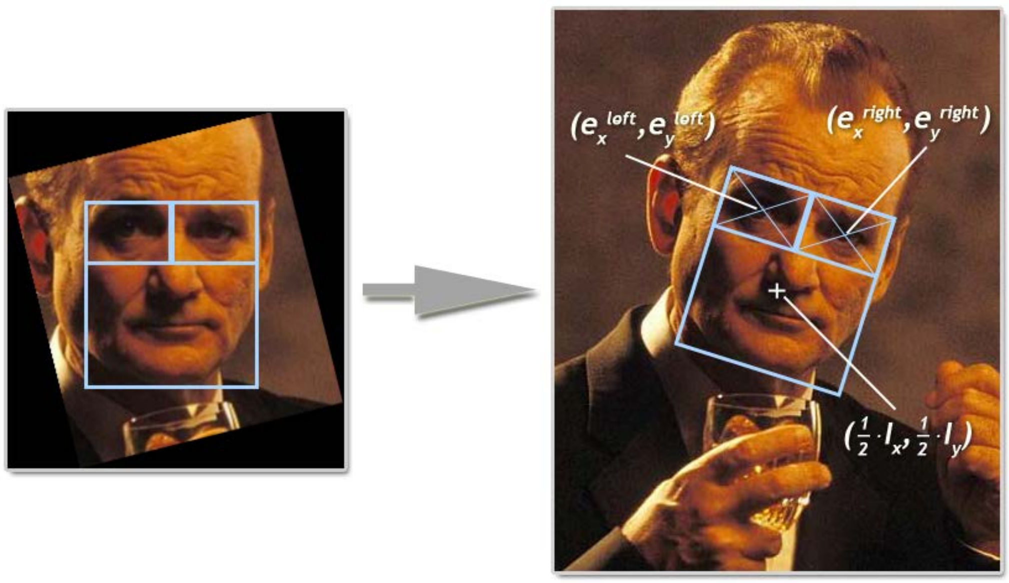
(e_x^{left}, e_y^{left})
$(e_x^{right}, e_y^{right})$
$(\frac{1}{2}\cdot I_x, \frac{1}{2}\cdot I_y)$

Figure 5.3: Translation from the centerpoint $(\frac{1}{2}I_x, \frac{1}{2}I_y)$ in the image.

The point (c_x, c_y) is now our reference point from where the angles of the eye are measured (Figure 5.3). The distance to the camera can be calculated with the help of the scale S. Since distance is proportional to the inverse of the scale, we can calculate the distance to the camera as follows:

$$d^c = D \cdot \frac{1}{S}, \tag{5.8}$$

where D is a constant depending on the focus of the lens of the camera in use. The distance from the camera to the screen d^s is a parameter of the model (entered by the user). Now if a person is looking at the point (P_x, P_y) at the screen, we can calculate the gaze angles[3] (θ_x, θ_y) as follows:

$$\begin{pmatrix} \theta_x \\ \theta_y \end{pmatrix} = \arctan \begin{pmatrix} \frac{P_x - c_x}{d^c + d^s} \\ \frac{P_y - c_y}{d^c + d^s} \end{pmatrix}. \tag{5.9}$$

And vice versa, if the eye angles are (θ_x, θ_y), we can calculate the position on the screen (the person is looking at) as follows:

$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = (d^c + d^s) \begin{pmatrix} \tan\theta_x \\ \tan\theta_y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}. \tag{5.10}$$

[3]Note that these gaze angles (in the X and Y direction) include the head pose.

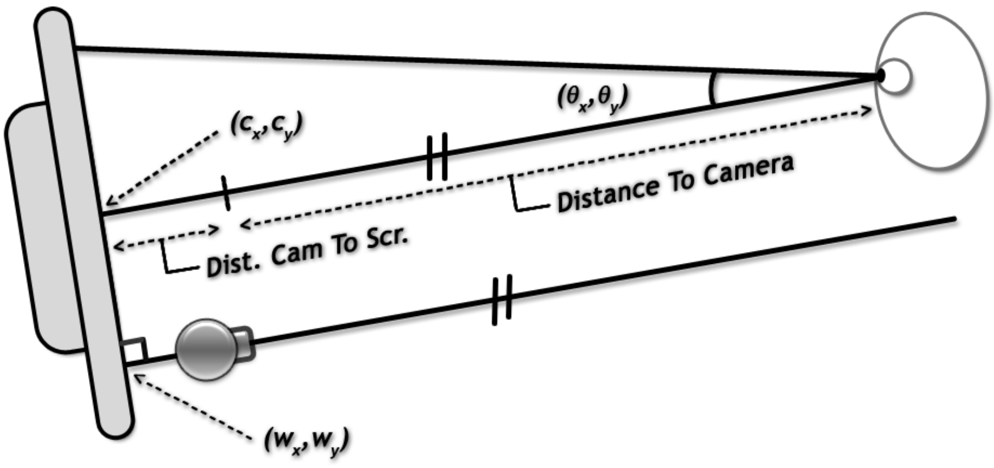

Figure 5.4: Side view of the gaze model used for estimating the gaze using a classifier.

Now, the above defines our gaze model for the classifier method. We want to build a classifier capable of predicting the gaze angles (θ_x, θ_y). From our representation of an eye we can subtract two important vectors for input for our classifier; the appearance vector and the landmark vector. Besides these vectors we added extra information to the input space by including the distance to the screen $(d^c + d^s)$, the head pose (ϕ_x, ϕ_x), the in-plane rotation ϕ_z, and the reference points (c_x, c_y) (which holds information about the translation from the camera center). Tabel 5.1 summarizes the three different ANNs we trained and tested.

Note that we combined the left and the right eye into single ANNs. We thus combined the appearance vector of the two eyes and the landmark vector of the two eyes, to predict the combined gaze angle of the two eyes. To combine both eyes into a single ANN, we used the average reference point of the two eyes.

5.3 Calibration

The defined models incorporate simplification from reality that can lead to estimation errors. A one time (per user) calibration is used to account for these errors. Our calibration method is derived from the calibration method discussed in [8]. The calibration consists of looking at five points on the screen and can be performed in 10-30 seconds. The calibration consists of

	Appearance Vector	Landmark Vector	Landmark + Appearance Vector
Distance	$(d^c + d^s)$	$(d^c + d^s)$	$(d^c + d^s)$
Head Pose	(ϕ_x, ϕ_x, ϕ_z)	(ϕ_x, ϕ_x, ϕ_z)	(ϕ_x, ϕ_x, ϕ_z)
Reference Point	(c_x, c_y)	(c_x, c_y)	(c_x, c_y)
network type	Feed forward	Feed forward	Feed forward
training method	Back propagation	Back propagation	Back propagation
network input neurons	38	66	98
network hidden neurons	15	15	15
network output neurons	2	2	2
size of training sets	914	914	914
learning speed	0.05	0.05	0.05
training epochs	≈ 4000	≈ 4000	≈ 4000

Table 5.1: Three ANNs for gaze classification.

computing the error in the estimated point of gaze (POG) when the user is looking at each of the five calibration points on the screen. At each of these points i the error (e^i_x, e^i_y) between the estimated position (n^i_x, n^i_y) and the calibration point (m^i_x, m^i_y) is calculated and recorded as:

$$\begin{pmatrix} e^i_x \\ e^i_y \end{pmatrix} = \begin{pmatrix} m^i_x - n^i_x \\ m^i_y - n^i_y \end{pmatrix}. \tag{5.11}$$

Future POG estimates are adjusted by applying the five correction factors e_i, each weighted inversely proportional to the distance the computed POG is from each of the original calibration POGs as follows.

$$d_i = ||p^e - n^i||, \tag{5.12}$$

$$w_i = \frac{1}{d_i \cdot \sum_{k=1}^{5} \frac{1}{d_k}}. \tag{5.13}$$

Then, the corrected POG p^c can be calculated as follows:

$$p^c = p^e + \sum_{k=1}^{5} w_i \cdot e_i. \tag{5.14}$$

This calibration method is capable of correcting pattern-like errors. It is based on the idea that the estimation/calculation error at a point on the screen will be consistent and that the errors at points close to that point can be derived from the error at that point. The experimental results discussed in the following chapter are all gathered after running this calibration routine.

Chapter 6

Experimental Results

In the previous chapters we discussed two methods of estimating the gaze of a person on a computer screen. The first one being direct calculation, using the positions of key features of the user's eyes, and a geometrical eye-model. The second method uses an ANN to classify the gaze direction of the user, based on a representation of the user's eyes. We defined three different representations of the user's eyes, namely the appearance vector, the landmark vector, and a combination of both (from now on denoted as the "combined vectors"). In this chapter a comparison will be made between these different methods, based on experimental results.

To be able to compare these methods, we comprised a test set of images of four different users looking at 88 points on a computer screen. These 88 points are equally distributed over a computer screen with a resolution of 1600x1200 pixels. With this test set we can now compare the performance of the different methods.

6.1 Test Results

For each of the 88 points and each of the users we recorded 6 images (of that user looking at that point). From these 6 images (per point, per user) we calculated the point of gaze (using one of the four proposed methods). The final estimated/calculated position of gaze is recorded as a running average over these 6 images, thus for each of the 88 points we now know the actual position and the estimated gaze position. Using this information we can compare the accuracy of the different methods.

Figure 6.1 displays scatter plots of the error in centimeters plotted against the X-axis and Y-axis. The (absolute) mean and standard deviation of these errors are summarized in Table 6.1. Table 6.1 also lists the (absolute) average angular error and standard deviation (in degrees). The angular error E_θ is calculated as follows:

$$E_\theta = \tan^{-1}\left(\frac{\sqrt{E_x^2 + E_y^2}}{d^c + d^s}\right), \tag{6.1}$$

where E_x and E_y are, respectively, the errors in the X and Y direction and $d^c + d^s$ is the distance from the user to the screen.

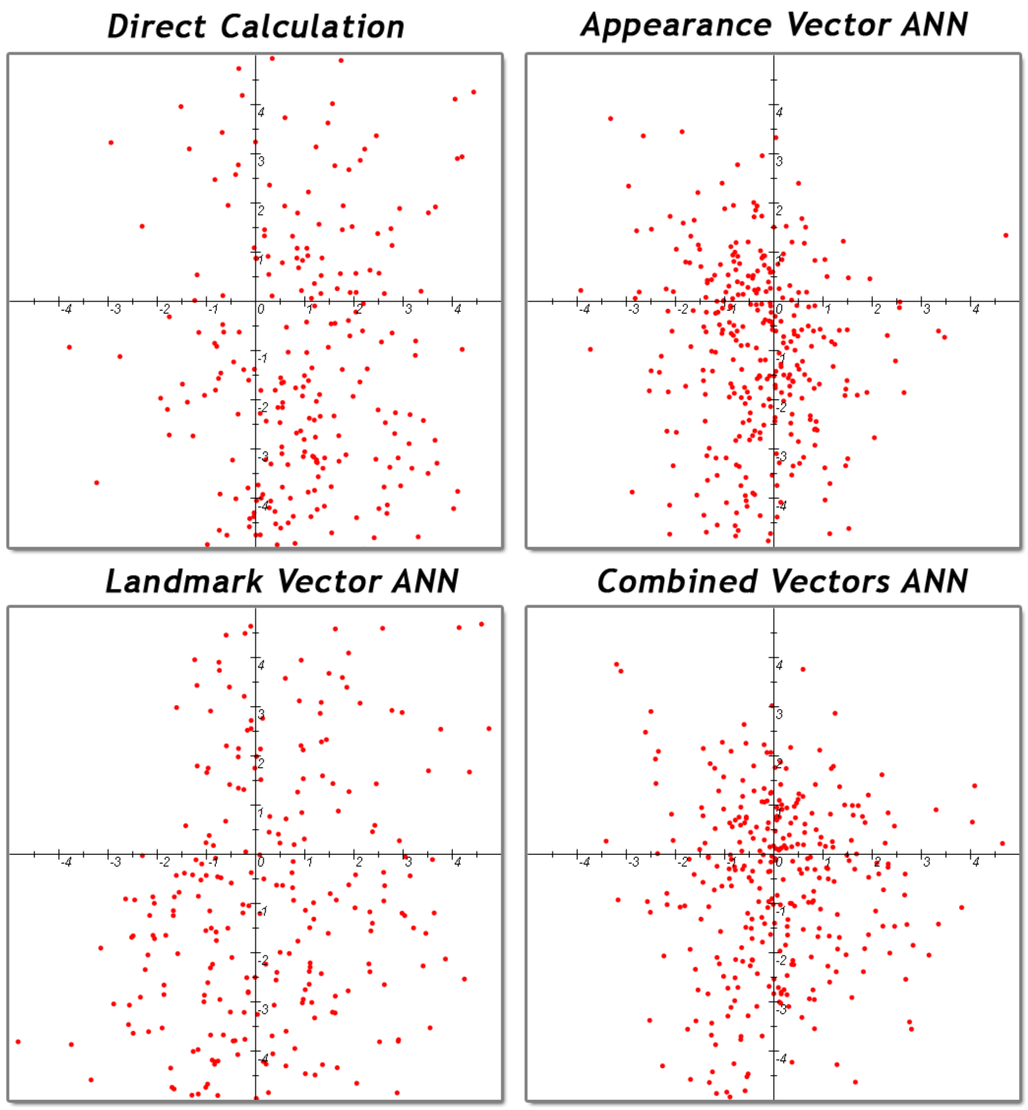

Figure 6.1: Scatterplots of the different methods plotted against the X-axis and the Y-axis.

	Error(cm) (X, Y)		Angular Error (deg)	
	Mean	St. Dev	Mean	St. Dev
Direct Calculation	1.60 , 3.98	1.52 , 2.87	3.92	2.23
Appearance Vector ANN	0.99 , 2.09	1.09 , 2.11	2.13	1.66
Landmark Vector ANN	1.83 , 3.78	1.82 , 2.98	3.81	2.48
Combined Vectors ANN	1.05 , 1.92	1.03 , 1.92	2.05	1.53

Table 6.1: Error distributions of the four proposed methods for gaze estimation.

Table 6.1 and Figure 6.1 clearly show that the direct calculation method performs worse than using a classifier to predict the gaze angle (mainly the ANNs based or partly based on the appearance vector). There are basically three reasons why in our setup this method does not perform good. The first one being that due to the low resolution of the images used, the predicted positions of the pupil and the eye-corners are not accurate enough for use in a geometrical eye model. Small shifts in the gaze direction of a user are not well reflected in the returned landmark positions of the appearance model.

The second reason is that, as discussed in Section 3.5, our methodology for estimating the head pose is not very accurate. Since in the direct calculation method the estimated head pose is used as a direct input, this can lead to irregular errors in the estimate of the gaze direction.

A third reason is that because of these (random) inaccuracies, our calibration method seems to fail. This can be seen in the scatter plot of the direct calculation method. The errors seem to lie more in the second quarter. This type of behavior should normally be tackled by the calibration method (since the calibration should align the errors around zero mean). The calibration method is capable of correcting pattern-like errors, but in the direct calculation method the initial estimates of the gaze direction at certain positions seem to differ at different times (or in other words, do not follow a pattern).

This same effect can be found in the results of the classifier based on the landmark vector. From the three proposed representations of the eyes the representation based on the landmark vector performs the worst, due to the same reasons as with the direct calculation method. The two other classifier methods, based completely or partly on the appearance vector perform a lot better. Apparently the appearance vector contains enough information for predicting the gaze direction of a user. Since the appearance vector contains not only information about the key positions of the eyes of the user but also texture information, small shifts in the gaze direction are better represented in the appearance vector than in the landmark vector.

Table 6.1 displays another noticeable effect of our proposed methods for gaze estimation. All the methods perform better on predicting the gaze direction in the X direction (horizontal) than in the Y direction (vertical).

This effect is probably due to the horizontal shape of the human eye. When looking to the left or the right the white area surrounding the iris changes, when looking up or down this has less effect. Therefore looking left or right is more easily picked up by the AAM and thus more accurately modelled.

6.2 Performance Evaluation

Both the appearance vector ANN and the combined vectors ANN perform quite well. With an average angular error of 2.05 degrees and a standard deviation of 1.66 degrees the combined vectors ANN clearly can estimate the gaze direction of a user up to a reasonable error, and so can the appearance vector ANN. They both perform almost equally well, although the results from Table 6.1 suggest that the combined vectors ANN performs slightly better and is slightly more robust. This effect can be simply explained by the fact that both methods partly use the same input and the Combined vectors ANN uses an extra input in the form of the landmark vector. Therefore, the landmark vector adds some extra accuracy to the network, since it is dealing with a bigger input space. Now we can conclude that the Combined vectors ANN performs best.

To visualize the performance of the Combined vectors method, Figure 6.2 and 6.3 show the results of two of the test users. The green points denote the actual position and the red points denote the estimated position. The first graph shows that although the estimates are almost never exactly correct, they are almost always very close. The graph also shows that at some points the estimate in the Y direction is worse than the estimate in the X direction. However, the second graph is less accurate than the first.

This raises questions concerning the generalization of the system. Since we use training methods based on datasets, the accuracy is completely dependent on how well those datasets reflect with the actual users of the system. Although this is of importance when building a fully functional system, we have restricted our test to showing the accuracy of our method. Thus, further tests concerning the generalization properties have not been taken, since they are beyond the scope of this thesis.

We can now compare our results with other known gaze estimation systems. Ishikawa et al. ([9]) report an average error of 3.2 degrees, using the same direct calculation method. Their system is evaluated in a car, with one camera pointed at the driver and one camera pointed at the looking direction of the user. Their system uses a 3D AAM of the user's face to subtract the head pose, this could be a reason why they reported a smaller average error. Tobii Technology AB ([22]) reports a maximum error of 0.5 degrees in front of a 17" monitor. Tobii develops a one camera commercial system using infrared lighting. Their setup is an out-of-the-box plug and play solution (with the camera and lighting embedded in the screen).

In [15], Ohno et al. report an accuracy in the X direction between 0.23 degrees to 0.7 degrees in view angle, and an accuracy in the Y direction between 0.3 degrees to 0.9 degrees. But in the worst case, it was larger than 1.0 degrees. They present a gaze tracking system using a single camera and on-axis infrared light emitters. The gaze position is computed given the two estimated pupil centers utilizing an eyeball model.

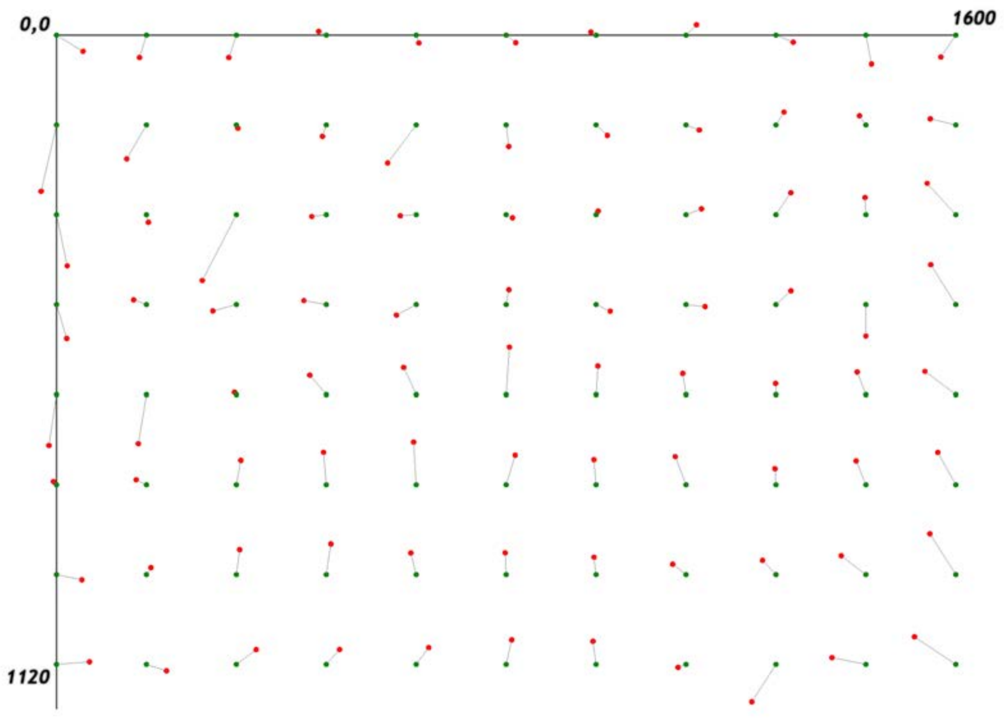

Figure 6.2: The results of one of the test users, using the combined vectors ANN

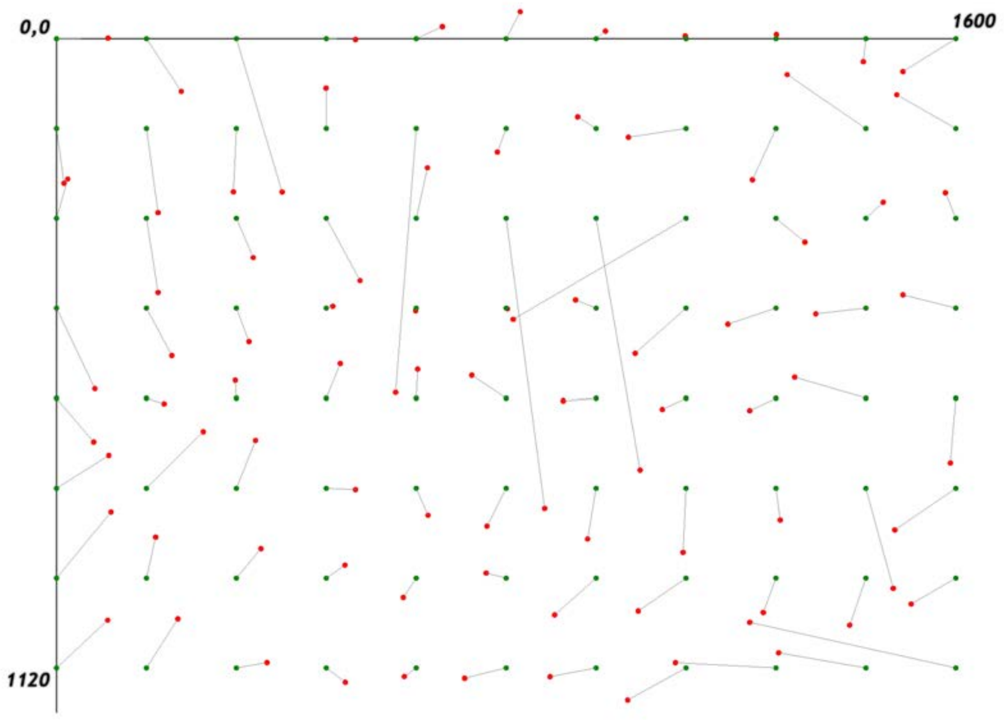

Figure 6.3: The results of another test users, using the combined vectors ANN

But, how accurate can we expect the gaze estimation to be? In fact, the gaze is not a single line in space. Our eyes do not only perceive the exact point of focus but also its surroundings. If we stare at a word in a text, then without moving our eyes, we can see a couple of words in the surroundings of that word. Thus a part of the error can be explained by the noise of how the human eye works. According to [20] this error is about 1.0 degree. Hence, a gaze estimation system can estimate the gaze of a person with a margin of error of plus or minus 1 degree of the visual angle.

Chapter 7

Conclusions

7.1 Conclusions

This thesis describes the implementation of and experimentation with a gaze estimation system, mainly based on an image processing/analysis technique called Active Appearance Models. The main goal is to develop a system that can estimate the point of gaze of a person on a computer screen in front of him/her, only using one normal off-the-shelf camera. In other words, without the use of multiple or infrared cameras and infrared lighting. The development of this system was subdivided in three stages.

Head pose estimation The first stage is to estimate the head pose of the user. We described two different methods of gathering a representation of a face in an image for use as the input for a classifier to classify the head pose. The first representation is a normalized pixel vector of a cropped face patch found by a face-finding module. The second representation uses a feed forward neural network to compress the normalized pixel vector to obtain a compressed normalized pixel vector.

For classifying the head pose based on these representations we chose to use an Artificial Neural Network, since these have proven their use in classification tasks and pattern recognition. To train such an ANN (a feed forward neural network in particular) we needed a head pose annotated database of images. Because of the lack of such a database, we used a method to calculate the head pose of a person in an image to construct such a database ourselves.

Our test results have shown that our method for estimating the head pose does not lead to very accurate results. There are a couple of reasons thinkable why the results are disappointing. The first reason is that because of the lack of a correct head pose annotated database, we had to use an estimation method to construct such a database. Although the method we used for constructing this database can lead to accurate estimates of a person's head pose, small errors in the annotation, due to for example a slightly different definition of the position of the nose in the image, can lead to quite different results. Therefore it is possible that conflicting data exists in our training set, making it hard to train a network correctly.

Another thinkable reason could be that estimating the head pose from our chosen representations is not a classification task that is easily learned by an ANN. Clearly, these inaccuracies will influence the accuracy of the final system.

Eye representation In the second stage we defined a way of representing the eyes of the user of the system. We chose to use an Active Appearance Model of an eye to get information about the eyes of the user. We chose this method because we are dealing with low resolution images (at least for the eyes). This means that varying the gaze direction of the eyes changes very few number of pixels in the image. Therefore it is hard to use local feature detectors (like detecting the edges of the iris, or detecting the circle of the pupil) because of the low level of detail. We defined a landmark annotation model and collected a database of eye images with different gaze angles to train the AAM.

We have shown that it is possible to train an AAM that can accurately model the eyes of the user of the system. By investigating the modes of appearance of the eye AAM, we can see that the different gaze angles are represented by the model. When fitted to a sequence of images depicting an eye, the resulting model instances closely resemble the original eye images and the retrieved landmark positions are accurate estimates of those positions in the original image.

Gaze estimation The last stage is to define a method for estimating the gaze from the estimated head pose and the representation of the eyes. We tested two methods for estimating the gaze, the first one being direct calculation of the gaze based on key features of the human eye. The second method is using an ANN to estimate the gaze of a person based on the representation of the eye. The second method was further subdivided into three different input methods for the ANN. An appearance vector ANN, a landmark vector ANN, and a combined vectors ANN.

We tested the final system on four different users looking at 88 different points on a computer screen in front of them. Our tests have shown that the combined vectors ANN method leads to the most accurate results. Although we have not tested the generalization properties of our system, we have shown that our method can lead to quite accurate results.

In comparison with other systems, we have a larger average angular error. However, the difference is only in the order of 1 to 1.5 degrees. Since we only use normal of the shelf equipment and the remaining resolution for the eyes in the image is low, these results are quite good.

Goals In the introduction we stated our goals for developing the gaze estimation system. We will look at these one by one and see to what extent they have been met in the developed system.

- The system uses only static images as input data (no video sequences are needed) and these images require no special preliminary treatment (such as manually placing markers or manual re-positioning or scaling), neither does it require specialized hardware such as infrared or 3D cameras and/or infrared lighting).

  Result: These goals have clearly been met. We use a simple webcam to automatically classify the gaze direction of the user of the system. No additional treatment is required and no specialized hardware is needed.

- The system can handle images obtained in a large variety of settings without retraining the model. No laboratory conditions are required where people have to sit straight and look into the camera under perfect lighting conditions with a diffuse background.

  Result: These goals have partly been met. First, although the system can handle images in a large variety of settings, the are situations where the lighting conditions are not good enough for the system. Second, our head pose estimation system does not lead to very accurate results, therefore the system performs better when the user sits straight in front of the camera. Third, we have not yet tested the generalization properties of the system. Although the system works for different users, the results can vary among different people.

- The system operates in real time.

  Result: On a 3.0 GHz Pentium D PC, a frame-rate of over 7 frames per second can be reached (where for each frame, the image is scanned for faces, the head pose is estimated, two eye models are created and these models are classified on the gaze direction). Using some extra fine-tuning and some extra processing power, 15 frames per second appears a realistic estimate of the attainable frame-rate today. Although not actually real-time this is sufficient for obtaining gaze estimation data.

Summarizing the above defined results, we can conclude that it is possible to obtain quite accurate results from a gaze estimation system using only a normal webcam and middle resolution images (640x480). However the results are not as accurate as the results of other systems using an infrared camera and/or higher resolution images. Therefore, the applicability of our system is dependent on the field is it is used in. If only moderately accurate results are required our method can be a cheap alternative to other more expensive systems.

7.2 Further research

In the previous sections we discussed our implementation of a gaze estimation system. We have shown that although it has opportunities for being a accurate system, there were some set-backs which influenced the accuracy. In this section we will discuss some possibilities for further research which could improve the accuracy of our system.

The first being our head pose estimator. Although it was not directly clear what the reason was for the inaccuracies, the solution for improving this system either lies in getter better data or using some other method to estimate the head pose. Another method would be using the method discussed in Section 3.4 in real time, using some method to find the position of the nose, and the mouth corners. An advantage of such a system would be that it is more clear what is causing the inaccuracies and individual settings could be used to achieve better results. The downside is that it would need another algorithm for finding the necessary points in a face, which is another constraint on the speed of the system.

Another point for further research is the generalization properties of the system. Although we have tested the system on several people, this is still no indication of how well the system performs on different people. Our best performing method (the appearance vectors ANN) is based on two training sets which completely influence the system's performance. The first training set is the AAM training set. Adding additional eye images to this training set could not only improve the overall fit of the AAM, it also could increase the generalization properties since the training set would have a larger variation embedded.

The second training set is the set of images used for training the ANN (images of people looking at certain points on the screen). Increasing the number of images in this training set could increase the learning abilities of the ANN and therefore lead to more accurate results.

Bibliography

[1] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Advances in Neural Information Processing Systems*, volume 6, pages 753–760. Morgan Kaufmann Publishers, Inc., 1994. 3

[2] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995. 21

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *5th European Conference on Computer Vision 1998*, volume 2, pages 484–498, 1998. 9, 14

[4] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester, Wolfson Image Analysis Unit, Imaging Science and Biomedical Engineering, United Kingdom, 1999. 9, 15

[5] R. Dony and S. Haykin. Neural network approaches to image compression. In *Proceedings. IEEE*, pages 83:288–303, 1995. 20

[6] A. H. Gee and R. Cipolla. Determining the gaze of faces in images. Technical Report CUED/F-INFENG/TR 174, Trumpington Street, Cambridge CB2 1PZ, England, 1994. 22, 23

[7] D. W. Hansen, J. P. Hansen, M. Nielsen, A. S. Johansen, and M. B. Stegmann. Eye typing using Markov and Active Appearance models. In *IEEE Workshop on Applications of Computer Vision - WACV*, pages 132–136, dec 2002. 3

[8] C. Hennessey, B. Noureddin, and P. Lawrence. A single camera eye-gaze tracking system with free head motion. In *ETRA '06: Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 87–94, New York, NY, USA, 2006. ACM Press. 48

[9] T. Ishikawa, S. Baker, I. Matthews, and T. Kanade. Passive driver gaze tracking with active appearance models. In *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, October 2004. 3, 41, 43, 55

[10] Q. Ji and Z. Zhu. Eye and gaze tracking for interactive graphic display. In *SMARTGRAPH '02: Proceedings of the 2nd international symposium on Smart graphics*, pages 79–85, New York, NY, USA, 2002. ACM Press. 3

[11] J. Jiang. Image compression with neural networks - a survey. In *Signal Processing: Image Communication*, pages 737–760, 1999. 20

[12] E. Lebert. Facial expression classification. Technical report, Vicar Vision BV, Amsterdam, the Netherlands, 1997. 26

[13] Yoshio Matsumoto and Alexander Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 499, Washington, DC, USA, 2000. IEEE Computer Society. 3

[14] C.H. Morimoto and M.R.M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1):4–24, 2005. 2

[15] T. Ohno, N. Mukawa, and A. Yoshikawa. Freegaze: a gaze tracking system for everyday gaze interaction. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 125–132, New York, NY, USA, 2002. ACM Press. 3, 55

[16] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997. 22

[17] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1996. 21

[18] B.D. Ripley. Pattern recognition via neural networks, 1996. 21

[19] R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. In *Proceedings of the Workshop on Perceptual User Interfaces (PUI'97)*, pages 98–100, Alberta, Canada, 1997. 45

[20] Eyetrack III study. http://poynterextra.org/EYETRACK2004/accuracy.htm. 56

[21] K.K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998. 7

[22] Tobii Technology. http://www.tobii.se. 55

[23] H. van Kuilenburg. Expressions exposed; model based methods for automatic analysis of face images. Master's thesis, Department of Philosophy, Utrecht University, 2005. 9, 13, 45, 46

[24] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer, November 1999. 21