Here's looking at you: Finding gaze direction of faces

Claudia Sameshima

MSc in Artificial Intelligence
Division of Informatics
University of Edinburgh
1999

Abstract

Gaze direction can be considered as a key factor in the process of social learning since is one of the most reliable indicators of the object that a person is considering or focusing on.

This project investigates the estimation of the fixation point, based on the information provided by both eyes, without using special equipment or being user-intrusive. The project assumes that the location of the eye is already estimated by a higher-order process, but not its gaze direction.

The steps followed are: iris location and position estimation, gaze direction calculation, eyes location in the scene, and fixation point projection in the screen. Two additional steps are the camera calibration and the user calibration.

The experiments carried out show that the iris location is quite good, but the location of other eye features to estimate the iris position, like the eye corners, is not a straightforward task. The fixation point calculation was not as good as expected due to the user calibration process.

Acknowledgements

To Dr. Bob Fisher for his guidance, his wisdom, and his great patience during this months.

To Dr. Gillian Hayes for proposing the initial idea of this project and accepting the new direction it took.

To Dr. Craig Robertson for his advice, his cheerful availability to help, and his edge detection code.

To Dr. Anthony Ashbrook for his invaluable time and his code for the camera calibration process.

To Arturo Espinosa-Romero for his concern and the images from his PhD project.

To Luis Miramontes for his priceless friendship through all these years, his advice, and the proof reading of some chapters.

To Ilias Boussios for accepting to participate in the experiments, although there was no opportunity to carry them out with his help.

To Jean Bunten for her unique way to look after our well-being.

To the lecturers and tutors for sharing their knowledge and time during the first terms of this course.

To my fellow students for being part of these wonderful period of my life.

To CONACYT for providing the means without which this goal would not be possible.

To Jesus, Incarnated Word, and Mary for being my guiding light.

To my parents for their unyielding faith, no matter how risky or implausible are my goals.

To my brother and sister for all the marvellous moments we share, despite the distance.

To my sister-in-law and my nephew for the happiness they bring to my brother's life.

To the memory of my late grandparents and aunt for being an example of tireless fight and for their wise and upright guidance.

To my aunts, uncles and cousins for the support they always give to my family.

To my friends in Mexico for holding the bonds between us, never minding the distance or time.

To my friends in Edinburgh for making me feel at home.

To the Mexican Society of Students in Edinburgh for being a niche of that wonderful country.

To Elia Fernandez, Carlos Garcia, Charles De Olden and Rafael Rivera for encouraging me to go after this objective.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Gaze direction can be considered as a key factor in the process of social learning since is one of the most reliable indicators of the object that a person is considering or focusing on. If the gaze direction and the fixation point can be accurately estimated, there are countless applications where these measurements can be used, like human-computer interfaces, communication tools and man-machine interaction.

In the case of man-machine interaction, the use of the fixation point can contribute to a great extent to make the task easier and closer to the way human beings communicate between them. For example, in the process of learning by imitation, if the learner has to determine the focus of attention of the teacher, one of the clues used by human beings is the direction in which the eyes are looking. If the objective is to mimic this kind of process, the agent should estimate the gaze direction of its human teacher.

The initial proposal of the project was focused on reimplementing the system made by Gee and Cipolla [Gee & Cipolla 95, Gee & Cipolla 94a, Gee & Cipolla 94b], where the gaze direction was estimated based on the face direction. The project considered some extensions of the system, like real-time performance and automatic feature detection. Nevertheless, when the project was analysed, other possibility emerged.

The gaze direction can be estimated using two different approaches: using the face direction, as in the system by Gee and Cipolla, or combining the information from both eyes, which is the new aim of the project.

This decision was made because the estimation of the gaze direction using the eyes direction may produce more accurate measurements. Other factor that was in favour of this approach was the fact that there is already a project [Stiefelhagen *et al.* 97] that overcame the limitations of the system by Gee and Cipolla. This system runs in real-time and automatically finds the face features.

When similar research already done was reviewed, it was found that in all the projects done except for [Heinzmann & Zelinsky 98], the fixation point estimation by eye direction was based on the information provided by one eye. Additionally, some projects required the user to wear intrusive equipment (*e.g.* head-mounted cameras) or used special equipment (*e.g.* LED sources, automatic-rotating cameras, tracking vision systems).

Therefore, this project investigates the estimation of the fixation point, based on the information provided by both eyes, without using special equipment or being user-intrusive.

An overview of the process to find the fixation point is outlined in Figure 1.1.

The project assumes that the location of the eye is already estimated by a higher-order process, but not its gaze direction. This decision was made because there is a great amount of research already done to detect face features. Chapter 2, Literature Review, contains a summary of the research done on face image processing.

Chapter 3, Iris Location and Position Estimation, details the next step in the process where the locations of the iris and the eye corners are calculated within the eye image, using edge detection and contour fitting techniques.

Using the locations found in the previous step, the fixation point is calculated by estimating the gaze direction, the eyes' locations in the scene and the projection of the fixation point in the screen. This steps are explained in Chapter 4, Location of the Fixation Point.

Chapter 5 contains the conclusions of the project.

Two additional processes needed for the location of the fixation point are the camera calibration and the user calibration, which are in Appendices A and B, respectively.

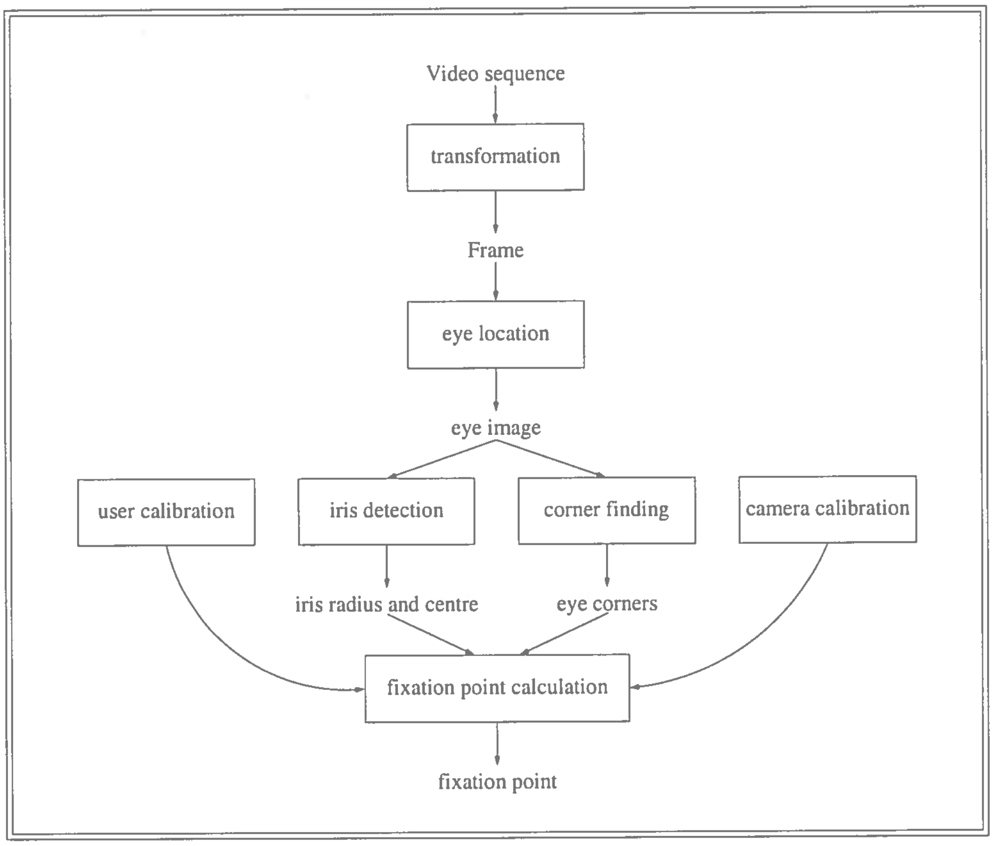


Figure 1.1: *Sequence for Finding the Fixation Point*

Chapter 2

Literature Review

2.1 Introduction

In general terms, the great diversity of research work related to human face image processing can be classified into the following categories:

Face detection The objective is to determine the presence or absence of a human face in the analysed image [Rowley *et al.* 96]. Therefore, this task can be considered as a prerequisite for the remaining categories [Yang *et al.* 98, Yow & Cipolla 97a].

Face recognition Consists of the identification of the face image under analysis, *i.e.* decide if the face coincides with one of the faces within a collection or database.

Tracking The purpose of this category is to follow the position of the face and/or facial features, through a sequence of images in which the face is moving. The research approaches can be divided into: *(a)* face tracking and *(b)* eye tracking.

Gaze direction The aim is to identify the fixation point to which the attention of the subject in the image is focused on. This task can be accomplished in two ways [Gee & Cipolla 95]: *(a)* by face direction and *(b)* by eye direction.

A summary of their relationships can be seen in Figure 2.1.

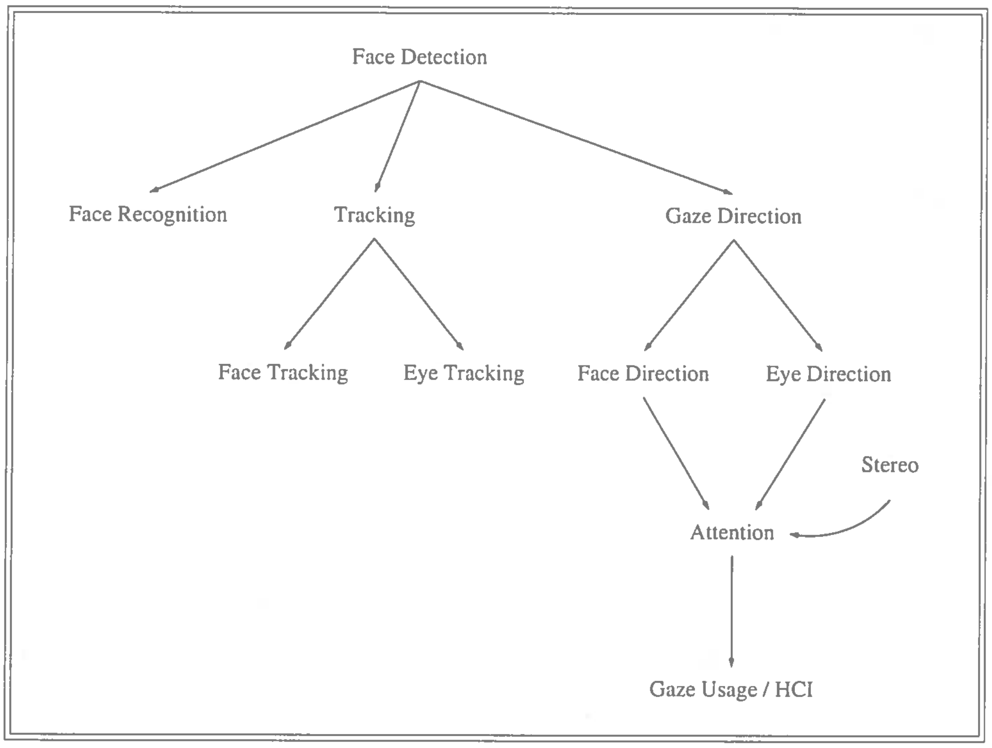


Figure 2.1: *Problem Decomposition*

In the following sections, the research work related to each category will be briefly described, along with the techniques employed to achieve their purposes.

Two additional sections are considered. **Gaze usage** comprises the applications done using gaze direction and fixation point. **Stereo matching** is described in Chapter 4. For the purposes of the present work, it is considered as part of the tasks needed to determine the fixation point, since the information provided by both eyes will be used.

2.2 Face Detection

The approaches taken for this category can be classified as follows:

- Whole face
    - * Skin colour
    - * Template matching
- Features
- Neural networks

2.2.1 Skin Colour

This technique consists of two stages [Yow & Cipolla 97a]: *(a)* each pixel is labelled, according to its similarity to skin colour and *(b)* each subregion is labelled as a face, if it contains a large blob of pixels labelled as skin colour.

The purpose of [Yang *et al.* 98] is to present the statistical study of a skin colour model. The technique used is: *(a)* skin colour distribution is analysed in the RGB space, *(b)* goodness-to-fit tests are performed, and *(c)* a maximum likelihood method is used to predict or approximate new parameters. This paper shows the feasibility of this method for face detection.

Some applications that use this approach are:

[Jebara & Pentland 97], which obtained its own skin colour model. Training samples of skin colour were obtained, with different tones and illumination. The method used is the one described above by [Yow & Cipolla 97a].

[Stiefelhagen *et al.* 97] used a statistical colour model like the one described in [Yang *et al.* 98], and the method used is the one described above by [Yow & Cipolla 97a].

[Bala *et al.* 97] and [Liu 98] used a statistical colour model as well, but selected skin colour by adaptive thresholding and a look-up table. A separation of fore- and background by segmentation was also performed.

Further descriptions can be found as follows: [Jebara & Pentland 97] in Sections 2.2.4 and 2.4, [Stiefelhagen *et al.* 97] in Sections 2.2.3 and 2.4, [Bala *et al.* 97] in Sections 2.2.3 and 2.5, and [Liu 98] in Sections 2.2.3 and 2.7.

2.2.2 Template Matching

[Scassellati 98] describes an active vision system that saccaded to a person facing it. Two cameras were used: a peripheral camera and a foveal camera. The technique used can be described as: *(a)* the face was detected by applying a ratio template comparison, *(b)* the system saccaded to the highest score face location, using sensori-motor mapping, and *(c)* obtained a high resolution image of the eye, using the location provided by the ratio template and a peripheral-to-foveal mapping. This application could detect faces in frontal views, under varying lighting conditions and in a cluttered environment.

2.2.3 Features

In this technique [Yow & Cipolla 97a], the basic steps are: *(a)* search the image for a set of facial features and *(b)* group those features into face candidates, based on geometrical relationships.

Some of the applications under this category are:

[Burl *et al.* 95] describes an application that located faces in cluttered environments and with occlusion of features. The technique used was: *(a)* features were found by multi-orientation, multi-scale Gaussian derivative filters, and *(b)* features were grouped in "constellations", which were ranked according to their face-like shape. The system achieved the objective planned, but only in quasi-frontal view faces.

[Yow & Cipolla 97b] and [Yow & Cipolla 97a] follow similar approaches. Their aim was to find face features, with unknown and varying orientation and viewpoint. [Yow & Cipolla 97b] used a second derivative Gaussian filter and [Yow & Cipolla 97a] used two filters: a Gaussian derivative filter and its Hilbert transform. Feature grouping was performed in partial groups, using geometrical knowledge of facial features. False positives were rejected by a belief network. Both applications worked well, but [Yow & Cipolla 97b] failed in extreme illumination conditions, occlusions and different facial expressions. [Yow & Cipolla 97a] failed to detect images where the face was too small.

[Stiefelhagen *et al.* 97] assumes an initial frontal view. Face detection is described in Section 2.2.1. The features were located following the next steps: *(a)* pupils were located by iterative thresholding, using geometric constraints, *(b)* lip corners were located by horizontal integral projection and horizontal edge detection, based on the position of the eyes and the face model, and *(c)* the nostrils were located by iterative thresholding and using the position of the eyes, lips and the face model.

[Bala *et al.* 97] and [Liu 98] detect the eyes by blinking, comparing luminance differences of successive images. The pupil was searched by looking for a circle-like dark region and its centre was calculated as the mid-point of a rectangular block.

Further descriptions are: [Stiefelhagen *et al.* 97] in Section 2.4, [Bala *et al.* 97] in Section 2.5, and [Liu 98] in Section 2.7.

2.2.4 Neural Networks

The stages followed in this technique are [Yow & Cipolla 97a]: *(a)* the regions of the image are subsampled to a standard size sub-image and *(b)* the sub-images are passed through a neural network filter.

[Rowley *et al.* 96] detected frontal views of faces in gray scale images. The system performed a preprocessing step by lighting correction and histogram equalisation. Two neural networks with retinal connections in their input layer were used, arbitrating their outputs by different heuristics (*ANDing*, *ORing* and counting the number of detections within a neighbourhood). The system detected all the faces present in the images, only if they were frontal or slightly rotated. It also used a bootstrap algorithm to add non-face examples during the training phase. The face images used were collected in a database at CMU and Harvard (*http://www.ius.cs.cmu.edu/IUS/har1/har/usr0/har/faces/test/*). The non-face images were created using random pixel intensities.

[Maurer & von derMalsburg 96], represented face landmarks as graphs, composed of jets (filter response of Gabor wavelets). It used a gallery of frontal faces as a training and test set. [Jebara & Pentland 97] used dark symmetry transforms, 3D warping and eigenfaces. Some examples can be found in *http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/computerVision_withFrameInit.html.*

[Krüger *et al.* 97] describes a system that automatically determines position, size and pose of the head. It used a bunch of graphs, that represented landmarks of the face. Elastic Graph Matching was used to estimate pose, size and position. Statistical methods were used to improve speed and performance, based on principles that stated how to select landmarks. It used the FERET database, provided by the DARPA/ARL FERET program (for information of this database contact Jonathan Phillips, at *jphillip@nvl.army.mil*).

[Talmi & Liu 98] used a combination of techniques: *(a)* eyes were detected by PCA (eigeneyes), *(b)* the 3D position of the eyes was calculated by stereo-matching, *(c)* the pupils were detected by using a LED light source, which produced a reflection in the eye, *(d)* a Sobel operator and a Circle Hough Transform were applied to calculate the centre and diameter of the pupil.

Further descriptions can be found in: [Maurer & von derMalsburg 96] and [Jebara & Pentland 97] in Section 2.4, and [Talmi & Liu 98] in Section 2.7.

2.3 Face Recognition

There is a lot of work done in this category. The early proposed classifications can be found in [Samal & Iyengar 92], [Valentin *et al.* 94] and [Chellappa *et al.* 95], where nonconnectionist and connectionist approaches are described. In [Fromherz *et al.* 97] the works are organised into frontal, profile and view-tolerant recognition.

Frontal recognition includes approaches where a preprocessing step finds and extracts facial features in 2D face images, which are matched against the corresponding features of a face database.

Profile recognition is relative easier to analyse than frontal recognition, allowing fast algorithms. But not much confidence has been put in this approach and pure profile algorithms are rare.

View-tolerant recognition employs various techniques to correct for perspective or pose-based effects, due to illumination and the 3D nature of the head.

2.4 Face Tracking

The techniques used for this category can be divided into feature-based and neural network approaches.

2.4.1 Feature-Based

[Gee & Cipolla 94b] describes a non-intrusive gaze tracking system by head orientation. Feature localisation was done by hand in the first frame (eyes and mouth corners). The tracking phase consisted of: *(a)* prediction of features by linear extrapolation over the previous two frames, *(b)* the pupil was detected by finding the darkest pixel in the window around the expected position, the mouth line was detected by searching for a line of darkest pixels and the corner was detected by pixel contrast, and *(c)* the tracking was smoothed by a first order low pass filter. The tracking was improved by stating multiple hypothesis, using facial geometry constraints.

[Gee & Cipolla 95] did face tracking using a minimal subset of data to estimate pose. Feature detection is the same as in [Gee & Cipolla 94b]. Pose estimation was done by a RANSAC algorithm, implemented in three forms: *(a)* a consensus tracker, where candidate poses were selected based on the size of the consensus set, *(b)* a temporal continuity tracker, where the selection was done based on the smoothness of the implied motion, and *(c)* a temporal consensus tracker, which combined both trackers by a Bayesian inference framework. The consensus tracker had problems with occlusions, the temporal continuity tracker did not cope with fast head movements, but the temporal consensus tracker worked well, selecting the best subset.

[Heinzmann & Zelinsky 97] tracked faces in real-time, implementing a gesture recognition application. Feature detection was performed by template matching, applied by a MEP Fujitsu tracking vision system. Individual search windows helped each other to track their features, using geometric relationships. Kalman filters were used to merge uncertain tracking data with the model. The tracking was reliable under stable illumination and head rotation not exceeding 60^o. It automatically initialised tracking without restrictions on head position and coped with occlusions. Gesture recognition was done in parallel.

[Stiefelhagen *et al.* 97] describes a non-intrusive, automatic face tracker. After the face and features were detected (as described in Sections 2.2.1 and 2.2.3), features were searched on windows around the last feature position. Predictions were made by linear extrapolation over the two last previous positions. Outliers were detected by using the algorithms proposed in [Gee & Cipolla 95]. Tracking recovery was done by checking if the features were within the face region. A comparison was made between the current features and the model, initialising the search windows and geometrical restrictions to the previous found pose.

2.4.2 Neural Networks

[Maurer & von derMalsburg 96] presents a face tracking system on image sequences which were continuous in time, coping with face rotation. Feature detection is described in Section 2.2.4. Tracking was made by computation of the displacement of single nodes, in two consecutive frames by disparity estimation in stereo images. The learning process was made by the following steps: *(a)* one direction was assumed, *(b)* the average movement direction of all nodes, deviation and cost values of nodes were computed, *(d)* the lowest cost nodes were considered as part of the optimal graph. The system assumed that the first frame had a frontal view of the face.

[McKenna *et al.* 97] used Gabor wavelets and a Point Distribution Model (PDM) to detect features. Tracking was initialised manually, by positioning feature points on the first frame. Estimations of new position were made by Gabor wavelet displacement estimation, which were aligned with the PDM shape model and projected onto eigenvectors. The shape was reconstructed and realigned to the image, to give the new position. The system was generic, so it can be applied to other objects. PDM had problems to cope with large deformations and rotations in depth.

2.4.3 Hybrid

[Jebara & Pentland 97] describes an automatic face tracking system, in real-time. Face and feature detection can be found in Sections 2.2.1 and 2.2.4. Tracking was made by defining windows for features and applying template matching by SSD correlation. Tracking recovery used structure from motion, Kalman filters and eigenheads. The resulting system was robust and fast, with automatic initialisation and re-initialisation upon failure.

2.5 Eye Tracking

[Young *et al.* 95] performs eye tracking by iris localisation. The user must wear a head-mounted camera. Iris detection was performed using an eye model, applying a Canny edge detector algorithm and a Hough transform algorithm, finding iris radius and centre. Tracking was implemented in two ways: *(a)* by locating the iris in each frame and *(b)* by applying an active contour method. The application was accurate, except for situations where strong highlights were present.

[Bala *et al.* 97] presents an automatic face and eye tracking system. Face and feature detection are presented in Sections 2.2.1 and 2.2.3. The tracking stage used luminance-adapted block matching. In case of tracking failure, the system returned to the initialisation stage.

2.6 Face Direction

[Gee & Cipolla 94a] describes a non-intrusive system for gaze direction determination, based on head orientation. The model was based on measurements taken from the eyes, nose and mouth. The estimation of gaze was based on facial normal, eye line, and gaze direction.

[Horprasert *et al.* 96] estimated the 3D head orientation in a single monocular image. Feature detection is provided by the application described in [Black & Yacoob 95]. The system computed roll, yaw and pitch based on eye corners and nose tip, projective invariants of face symmetry and statistical analysis of face structure. The application obtained the three rotation angles, but only if the face is frontal (*i.e.* with eyes and nose visible).

2.7 Eye Direction

[Baluja & Pomerleau 94] presents a gaze tracking system, which is non-intrusive and the user is free to move. The eye was located by searching for a specular reflection of a stationary light, in the face of the subject. A window surrounding the reflection was extracted and used as an input to a three layer, feed-forward network. Various architectures were tested. The system was used in an eye-mouse application, but needed a large amount of data to be collected. Each image was considered in isolation to make the predictions.

[Copeland & Trivedi 97] describes an experimental framework for evaluating target acquisition. The subject had to use a fixed helmet and a chin rest, and a small glossy black paper circle was placed below the eye. The pupil and the reference mark were found by thresholding, connectivity analysis and roundness measurement by ellipse fitting. The centroids of both features were stored, to compensate small head movements. The fixation point was determined using data from tracking, the images loaded on the screen when the test was conducted and other measurements, which were fitted in a geometrical model. The results were used to compute statistics for target detection tasks.

[Heinzmann & Zelinsky 98] tracked a face, estimating 3D head pose and gaze fixation point in real-time. Feature detection and tracking is the same as in [Heinzmann & Zelinsky 97]. Pose estimation used affine transformation, based on a three point model fitting. The gaze vector direction was determined by the iris and eye corners location. Both locations were transformed to an eye orientation vector, considering head pose. The results of both eyes were merged, according to their confidence.

[Talmi & Liu 98] used eye tracking for a stereoscopic display. Feature detection is detailed in Section 2.2.4. The gaze vector is calculated by the positions of the pupil and the reflex of a LED light source, as well as an eye model. Head movement compensation was also added as well. The system was used to adjust the depth of focus of the stereoscopic display.

[Liu 98] determined the point of fixation, allowing head movement. Face and feature detection can be found in Sections 2.2.1 and 2.2.3. The 3D position of the eyes was calculated using the 2D eye coordinates and optical and geometrical parameters of the camera. The fixation point was estimated by calculating the gaze vector, using an eye model, compensating head movement and intersecting the gaze vector with the display coordinates, by a geometrical model.

2.8 Gaze Usage

[Velichkovsky & Hansen 96] presents an overview of the progress in human-computer interaction. Gaze can be applied in: *(a)* user interfaces, like eye-mouse or non-command multimedia applications, and *(b)* communication tools, to transmit user's intention in face-to-face interaction, to help with interactive documentation and cooperative support, or to facilitate visualisation of non-verbal perception.

[Stiefelhagen *et al.* 98] describes a system that tracked the focus of attention of several participants in a meeting. The head pose determination was implemented by two neural networks, one for pan and one for tilt. The networks were multi-layer perceptrons, with one hidden layer and used back-propagation with momentum. The focus of attention was modeled by a Hidden Markov Model (HMM), where gaze estimates were considered as probabilistic functions of the different states.

[Pappu & Beardsley 98] classified the focus of attention of a car driver. The steps followed are: *(a)* manual initialisation, by placing an ellipsoid, coincident to the driver's head, in fronto-parallel position, *(b)* synthetic views (*i.e.* rotated, shifted, etc.) were generated off-line, *(c)* new images were matched against the synthetic views, and *(d)* a pose space histogram was generated, showing the head poses that were most frequently adopted.

[Salvucci 99] performed fixation tracing of eye movements, using a HMM. The inputs to the system were eye movements data, target areas on the screen, and a process model grammar. The fixation centroids were found by converting raw eye movements to a sequence of fixation centroids, using a saccade-fixation model. The tracing was made by mapping the fixation centroids onto predictions of the process model, using a centroid sub-model for each target area. The sub-models were used to construct the traces model, incorporating the model grammar.

2.9 Conclusions

As stated in section 2.1, gaze direction can be estimated either by face direction or eye direction. The research works that utilise face direction are more focused on tracking and they trade accuracy for speed, as stated in [Gee & Cipolla 94a]. Therefore, the deployed applications under this category do not rely on an accurate determination of the fixation point, like the application described in [Stiefelhagen *et al.* 97].

On the other hand, the estimation of the gaze direction based on the eye direction provides the means to determine the fixation point in a more accurate way. Generally speaking, the steps followed by the latest research works are:

- Pupil / iris centre localisation
- Gaze vector estimation
- Gaze vector location in the scene, incorporating 3D eye position information
- Fixation point determination, intersecting the gaze vector with the screen

The techniques used for each step differ between works. [Copeland & Trivedi 97] and [Talmi & Liu 98] considered the information provided by one eye to determine the fixation point on the screen. [Heinzmann & Zelinsky 98] used the information of both eyes, but the fixation point was not calculated. The last step was the estimation of the gaze vector, placing its origin between the eyes. Nevertheless, it was stated that the fixation point can be calculated by intersecting the gaze vector with a world model.

The best approaches done so far are the ones described in [Talmi & Liu 98] and [Heinzmann & Zelinsky 98], since the applications were non-intrusive and the user could move freely. The disadvantage of [Talmi & Liu 98] is that a LED light source was required and in [Heinzmann & Zelinsky 98] a special equipment to perform the tracking was used.

Chapter 3

Iris Location and Position Estimation

As stated in Chapter 1, the process assumes that the eye is already located in the image by a higher-order process. Therefore, the first step is the iris location in the eye image. This chapter describes the techniques used for iris detection, namely edge detection and contour fitting, and then the techniques for estimating its position within the eye.

3.1 Iris Edge Detection

In image analysis, one of the most common tasks is edge detection [Jain *et al.* 95, Parker 97]. There is a great number of algorithms devoted to edge enhancing and detection due their importance.

Edge detection is the first step used in the iris location process, because an edge defines the outline of an object. Edges separate the object from the background and other objects, and are usually locations where the image has large intensity changes. Once the edges are found, the objects can be located and their properties can be measured. This means that, if the edges belonging to the iris are found, a model (*e.g.* a circle) can be fitted to those edges by contour fitting and, consequently, the iris position can be estimated.

In general, edge detection operators can be classified as:

Derivative operators These identify places where intensity changes take place.

Template matching The edge is modelled as an small image which contains the abstracted properties of a perfect edge.

Parametric edge models Mathematical models of the edge are used. The best operators incorporate a model of the noise.

Based on the review done in Chapter 2 of the research that perform eye features finding using edge detection, the operators tested were Sobel and Canny, since these were the ones used on those projects [Young *et al.* 95, Talmi & Liu 98]. By suggestion of Dr. Craig Robertson, Research Fellow in the Machine Vision Unit and second supervisor of this project, the Shen-Castan operator was also tested.

3.1.1 Sobel Operator

This is a derivative operator, where a 3x3 pixels neighbourhood is used to compute the gradient. The implementation of the partial derivatives can be made as a convolution mask. More importance is given to the pixels closer to the centre of the mask.

3.1.2 Canny Operator

The first derivative of a Gaussian is used, as well as an approximation of an operator that optimises the product of the signal-to-noise ratio and localisation. The steps involved are:

1. Image smoothing by a Gaussian filter.

2. Gradient magnitude and orientation calculation by finite-difference approximations for the partial derivatives.

3. Non-maxima suppression to the gradient magnitude.

4. Double thresholding to detect and link edges.

3.1.3 Shen-Castan Operator

This operator agrees with the Canny operator in the general form of the detector, but it uses a different optimal filter function called the **Infinite Symmetric Exponential Filter** (ISEF).

A detailed description of the operators used for edge detection can be found in [Jain *et al.* 95, Trucco & Verri 98]. A comparison between the Canny and Shen-Castan operators is made in [Parker 97].

3.1.4 Experiments

Two implementations of the operators were used: the programs provided by the HIPS image processing system and the programs written by Dr. Robertson.

The edge operators were applied using the default values, unless otherwise stated. The eye images were manually cropped from the original images using XV.

The images used in Figures 3.1, 3.2, and 3.3 were provided by Arturo Espinosa-Romero, PhD student. The examples were taken from the Face Images Database he collected as part of his research project about representation, identification and detection of human faces. A detailed description of his project can be found at *http://www.dai.ed.ac.uk/daidb/people/students/arturoe.html*

The rest of the images were taken with the camera used for the project. The characteristics of the camera and the parameters used when the images were taken can be found in Chapter A. All these images have a colour bitmap format, with a resolution of 640 x 480 pixels.

HIPS Algorithms

The first algorithms tested were the ones provided by the HIPS image processing system. A detailed description of the HIPS system can be found at *http://www.cns.nyu.edu/home/msl/hipsdescr.cgi*

Since the HIPS algorithms require images in byte format, the images were transformed from their original formats (colour and gray-scale tiff and colour bitmap) to the HIPS format using XV.

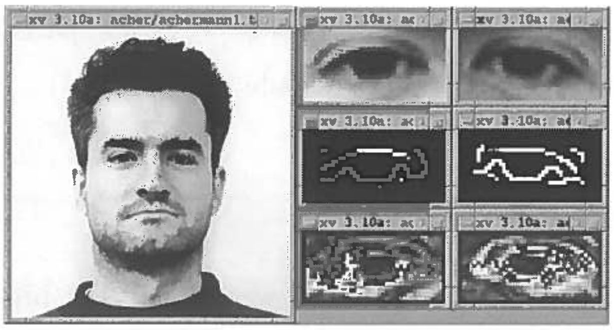
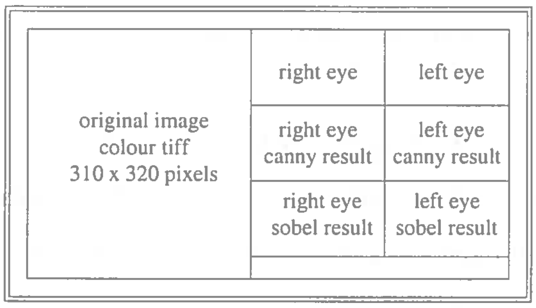


Figure 3.1: *HIPS Edge Detection Operators Comparison (1)*



The resolution of this image is too low *(310 x 320)*, compared to the one provided by the camera used for the project *(640 x 480)*, so the edge operators were tested on images with higher resolution *(570 x 739)*.

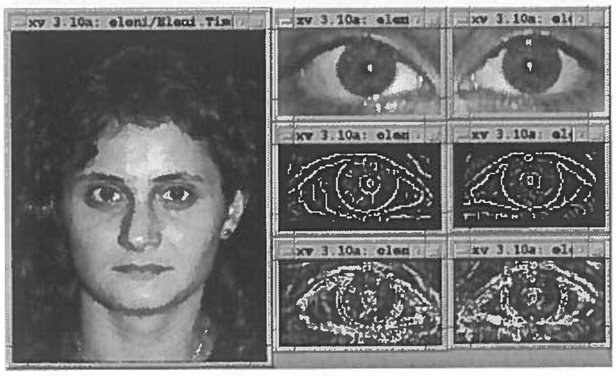


Figure 3.2: *HIPS Edge Detection Operators Comparison (2)*

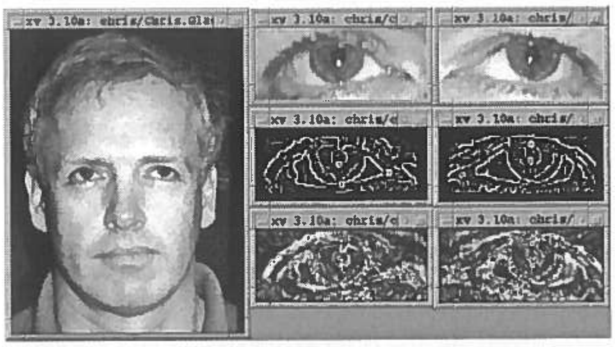


Figure 3.3: *HIPS Edge Detection Operators Comparison (3)*

original image gray scale tiff 570 x 739 pixels	right eye	left eye
	right eye canny result	left eye canny result
	right eye sobel result	left eye sobel result

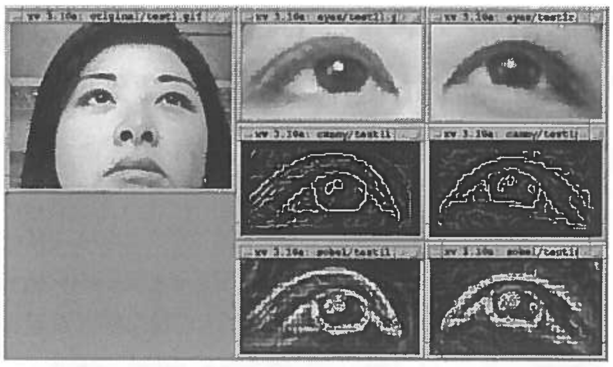
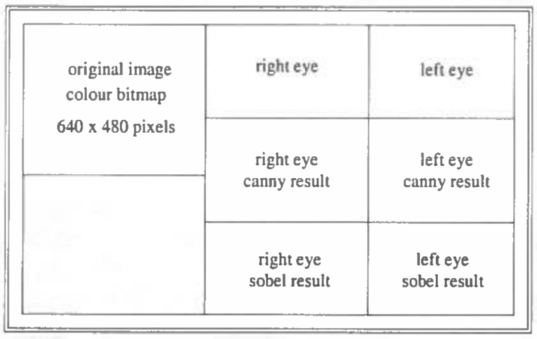


Figure 3.4: *HIPS Edge Detection Operators Comparison (4)*



Dr. Robertson's Implementation

Since the results produced by the operators of the HIPS system were too noisy (*i.e.* too many edges that do not belong to the iris), other implementations of the operators were tested. These programs were provided by Dr. Robertson and are based on the code in [Parker 97]. As stated earlier, the Shen-Castan operator was also tested.

The algorithms require images in PGM format, so the images were transformed from their original format in colour bitmap to gray-scale PGM images using XV.

From the tests carried out, the decision was to use the Canny operator algorithm implemented by Dr. Robertson, since the resulting images contained most of the iris edges and little noise.

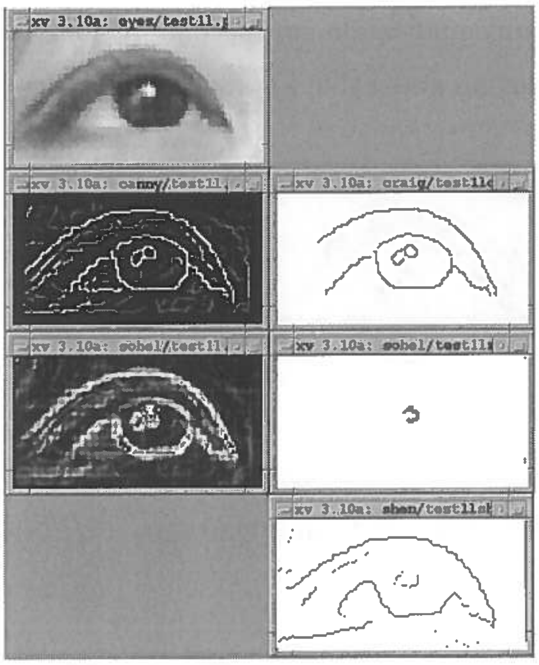


Figure 3.5: *Edge Detection Operators Comparison (1)*

right eye	
HIPS canny result	Dr. Robertson's canny result
HIPS sobel result	Dr. Robertson's sobel result
	Dr. Robertson's shen-castan result

Source Images Comparison

Another comparison was made to determine if there was any difference between using the original image and the image with histogram equalisation. As seen in Figure 3.6 it is better to use the original image, since the image with histogram equalisation produced more noise in the edge detection results and there was no noticeable improvement.

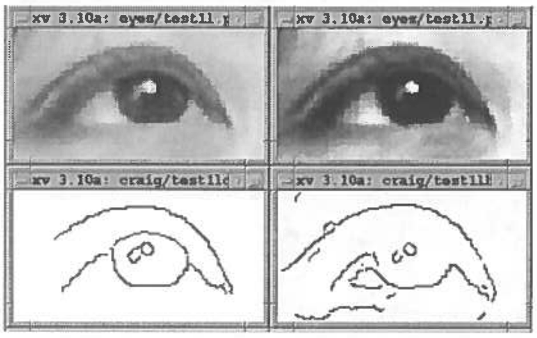


Figure 3.6: *Edge Detection Operators Comparison (2)*

right eye original	right eye with histogram equalisation
Dr. Robertson's canny result	Dr. Robertson's canny result (from histogram equalisated image)

Canny Operator Parameters Variation

The parameters used by the Canny operator were tested using different values. From the tests conducted, the operator that produced the best results (*i.e.* detection of most of the edges of the iris outline and less noise) was the Canny operator provided by Dr. Robertson using the default parameters (*low threshold=0*, *high threshold=1* and *Gaussian width=1*). The Canny operator selected was applied to additional images.

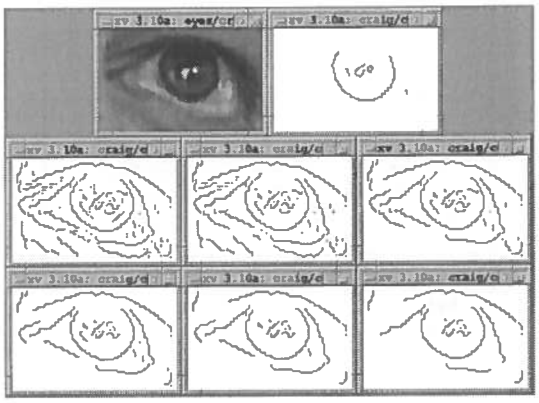
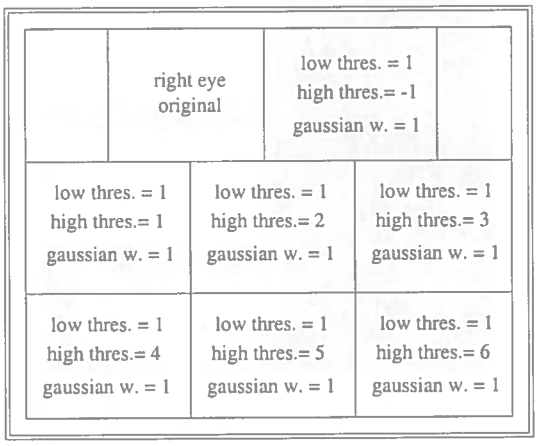


Figure 3.7: *Canny Operator Parameters Variation (1)*



right eye original	low thres. = 1 high thres.= -1 gaussian w. = 1	
low thres. = 1 high thres.= 1 gaussian w. = 1	low thres. = 1 high thres.= 2 gaussian w. = 1	low thres. = 1 high thres.= 3 gaussian w. = 1
low thres. = 1 high thres.= 4 gaussian w. = 1	low thres. = 1 high thres.= 5 gaussian w. = 1	low thres. = 1 high thres.= 6 gaussian w. = 1

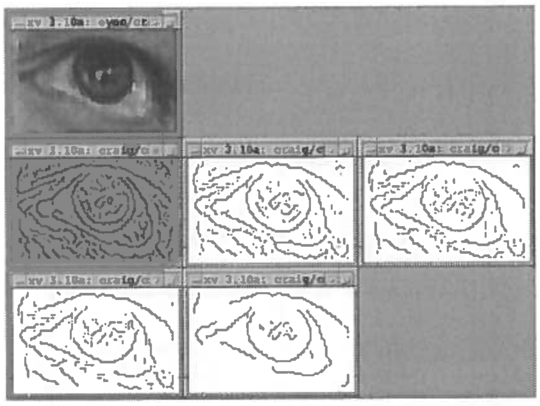


Figure 3.8: *Canny Operator Parameters Variation (2)*

right eye original		
low thres. = 1 high thres.= 5 gaussian w. = 1	low thres. = 1 high thres.= 5 gaussian w. = 2	low thres. = 1 high thres.= 5 gaussian w. = 3
low thres. = 1 high thres.= 5 gaussian w. = 4	low thres. = 1 high thres.= 5 gaussian w. = 5	

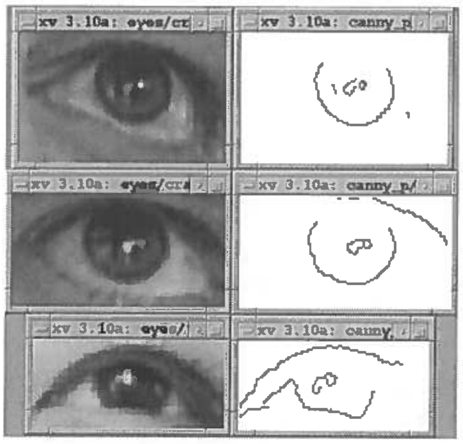


Figure 3.9: *Canny Operator Results*

left eye original	Dr. Robertson's canny result
right eye original	Dr. Robertson's canny result
left eye original	Dr. Robertson's canny result

3.2 Iris Contour Fitting

Once the edges of the iris outline are detected, the next step is to fit a model to those edges by contour fitting. A contour is a representation for a region boundary, where edges are linked [Jain *et al.* 95]. Contours can be modelled as ordered lists of edges or as curves.

If contours are modelled as curves, two techniques can be used: **curve interpolation** means that a curve passes through a list of points; **curve approximation** means that a curve passes close to a list of points. The model considered for the iris is a circle, so the technique used is curve approximation.

Two methods that use curve approximation are the Hough Transform and Random Sample Consensus (RANSAC). Similarly to the edge detection process, the technique considered for contour fitting was the Hough transform, since it was used in some research works reviewed in Chapter 2 like [Young *et al.* 95, Talmi & Liu 98]. The RANSAC algorithm was considered by suggestion of Dr. Robertson.

3.2.1 Hough Transform

This kind of method uses a parameter estimation technique by a voting mechanism. Each point on a curve votes for several combinations of parameters. The parameters that score the majority of the votes are considered as the winners.

If the model is a circle, it has three parameters: two for the centre and one for the radius. If the gradient angle of the edges is available, generally as a by-product of the edge detection algorithm, the value of the radius will be the only unknown parameter, since the gradient determines the direction of the vector from the centre of the circle to each edge.

3.2.2 RANSAC

This regression paradigm finds a significant group of points that are consistent with a model and rejects the remaining points as outliers [Fischler & Bolles 81]. The basic steps are:

1. Choose three points at random and fit the circle model to them.
2. Back-project the fitted model into the image, identifying the points that lie within an error margin. These points will be part of the *consensus* or evidence set for each candidate model.
3. Repeat steps *1* and *2* a determined number of trials and choose the model which is supported by the largest set.

The circle model used to fit the points is as follows:
Implicit equation for the circle, with radius r and centre (x_0, y_0):

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Having three points $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ and $p_3 = (x_3, y_3)$, the origin of the coordinate system is placed at point p_1:

$$\begin{aligned} x' &= x - x_1 \\ y' &= y - y_1 \end{aligned}$$

Therefore, the equation of the circle is transformed to:

$$(x' - x'_0)^2 + (y' - y'_0)^2 = r^2$$

Transforming the original coordinates into the new coordinate system for the three points in the implicit equation of the circle:

$$\begin{aligned}
x_0'^2 + y_0'^2 - r^2 &= 0 && (3.1)\\
x_2'^2 - 2x_2'x_0' + x_0'^2 + y_2'^2 - 2y_2'y_0' + y_0'^2 - r^2 &= 0 && (3.2)\\
x_3'^2 - 2x_3'x_0' + x_0'^2 + y_3'^2 - 2y_3'y_0' + y_0'^2 - r^2 &= 0 && (3.3)
\end{aligned}$$

Subtracting Equation (3.1) from Equations (3.2) and (3.3):

$$\begin{aligned}
2x_2'x_0' + 2y_2'y_0' &= x_2'^2 + y_2'^2 && (3.4)\\
2x_3'x_0' + 2y_3'y_0' &= x_3'^2 + y_3'^2 && (3.5)
\end{aligned}$$

Finding x_0 and y_0 from Equations (3.4) and (3.5):

$$x_0' = \frac{x_2'}{2} + \frac{y_2'^2}{2x_2'} - \frac{y_2'y_0'}{x_2'}$$

$$y_0' = -\frac{x_2'x_3'}{2y_2'} + \frac{x_3'^2}{2y_3'} - \frac{x_2'y_3'^2}{2x_3'y_2'} + \frac{y_3'}{2} + \frac{x_2'^2}{2y_2'} - \frac{x_2'x_3'}{2y_3'} + \frac{y_2'}{2} - \frac{x_3'y_2'^2}{2x_2'y_3'}$$

which are the coordinates of the circle centre in the new coordinate system. To obtain the coordinates in the original coordinate system, (x_1, y_1) should be added to (x_0', y_0'). The radius can be calculated from Equation (3.1).

Success Probability

Since RANSAC chooses points at random, the probability of obtaining a good model from those points must be measured. A **bad triplet** is formed by one or more **bad points**, which are those points that do not correspond to the model (*i.e.* they are not part of the iris circle but are included in the result of the edge detection operator).

The probability of cluttering (*i.e.* select a bad point) is calculated by:

$$P_{clutter} = \frac{numberofbadpoints}{numberofpoints}$$

The probability of selecting a good triplet on a single trial can be obtained by:

$$\begin{aligned} P_{good} &= (1 - P_{clutter})^3 \\ P_{good} &= \left(1 - \frac{numberofbadpoints}{numberofpoints}\right)^3 \end{aligned}$$

The probability of selecting a good triplet can be very low if there is a lot of cluttering in the image. Therefore, the algorithm should have multiple trials.

The probability of failure of the algorithm over N consecutive trials is:

$$\begin{aligned} P_{failure} &= (1 - P_{good})^N \\ P_{failure} &= \left(1 - \left(1 - \frac{numberofbadpoints}{numberofpoints}\right)^3\right)^N \end{aligned}$$

The probability of success of the algorithm can be estimated by:

$$\begin{aligned} P_{success} &= 1 - P_{failure} \\ P_{success} &= 1 - \left(1 - \left(1 - \frac{numberofbadpoints}{numberofpoints}\right)^3\right)^N \end{aligned} \tag{3.6}$$

To determine the number of trials that the algorithm must be run to guarantee a probability of success $P_{success}$, from Equation (3.6):

$$trials = \frac{\ln(1 - P_{success})}{\ln\left(1 - \left(1 - \frac{numberofbadpoints}{numberofpoints}\right)^3\right)}$$

3.2.3 Experiments

Although the Hough transform was considered for the contour fitting process in the first instance, the RANSAC algorithm was a better option. The Hough transform technique is more complex and its results are more difficult to analyse.

The number of parameters used cannot be too many, because the number of accumulators increases exponentially with the dimensions of the parameter space. This makes the Hough transform technique computationally inefficient if the model to fit is complex.

Once the votes for the parameters are obtained, convolution masks must be applied in order to smooth the peaks produced by outliers and determine which peaks are significant, but this is not a trivial task. Therefore, the algorithm tested was RANSAC.

Success Probability

From the explanation in section 3.2.2, to calculate the probability of success of the algorithm, the first step is to count the edge points of the image that do not correspond to the iris circle. This was done manually on the result images of the edge detection operator that appear in Figures 3.6 (the original image with no histogram equalisation) and 3.9.

The image in Figure 3.6 has 250 bad points from 366 points; the image on the top row of Figure 3.9 has 41 bad points of 137 points; the image in the middle row has 101 of 200, and the image in the bottom row has 144 of 256. The percentages of cluttering are 68.30%, 29.92%, 50.5%, and 56.25%, respectively.

Considering the worst case (68.30%) and a probability of success of 99.9%, the number of trials can be estimated by:

$$trials = \frac{\ln(1 - P_{success})}{\ln(1 - (1 - P_{clutter})^3)}$$

$$
\begin{aligned}
trials &= \frac{\ln(1-0.999)}{\ln(1-(1-0.683)^3)} \\
trials &= 213
\end{aligned}
$$

First Tests

The algorithm was tested using the results from the edge detection process. The first results obtained were not good because the values for some of the constraints used in the algorithm were wrong.

The constraints used to obtain better results from the algorithm were the location of the centre and the radius of the circle. For the circle centre the only values accepted were those that lay within the image, *i.e.* the coordinates of the centre were not negative (above or to the left of the image) or greater to the number of rows and columns of the image (below or to the right of the image. The value of the radius was limited between a lower bound and an upper bound. The lower bound was finally set to 25% of the image height to avoid circles that were too small, and the upper bound to 50%, to avoid circles that were bigger than the eye image.

The first results were not good, since the limits for the radius were too small. These values were set to 10% and 25% of the image height for the lower and upper limits, respectively. The problem was corrected by modifying the values to 25% and 50% of the image height. The results are shown in Figure 3.10.

Additional Tests

Once the constraints of the algorithm were corrected, the algorithm was applied to more images. The face was rotated only in the x direction to the left and right (considering a coordinate system with the x axis in the horizontal direction, the y axis in the vertical direction, and the z axis pointing to the screen).

From that face positions, the eyes were looking at different locations on the screen, as displayed in Figure 3.11.

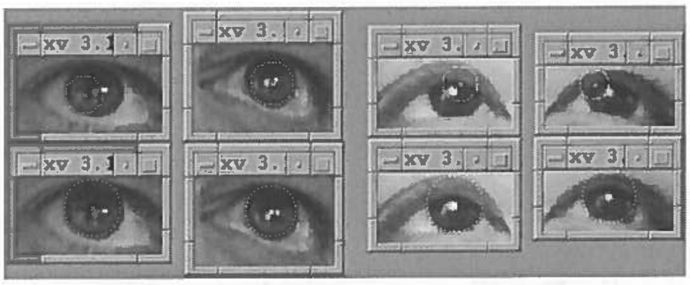
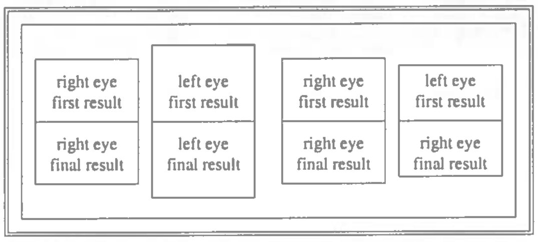


Figure 3.10: *RANSAC Results (1)*



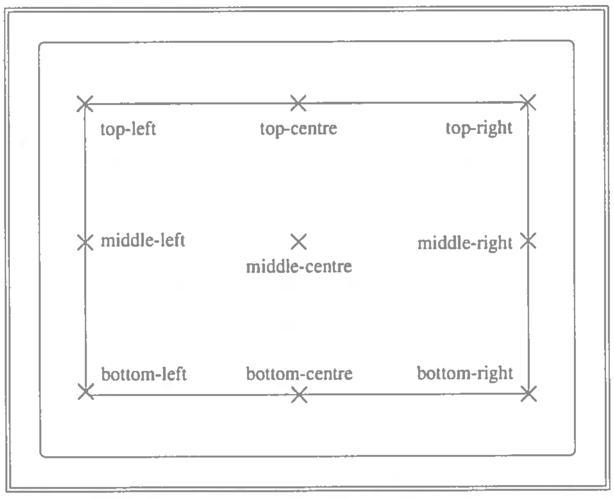


Figure 3.11: *Screen Locations*

Not all the results were good, since the circle was not placed correctly. From the 62 images, 7 of them were completely misplaced and 5 were slightly shifted. The wrong results can be seen in Figure 3.12.

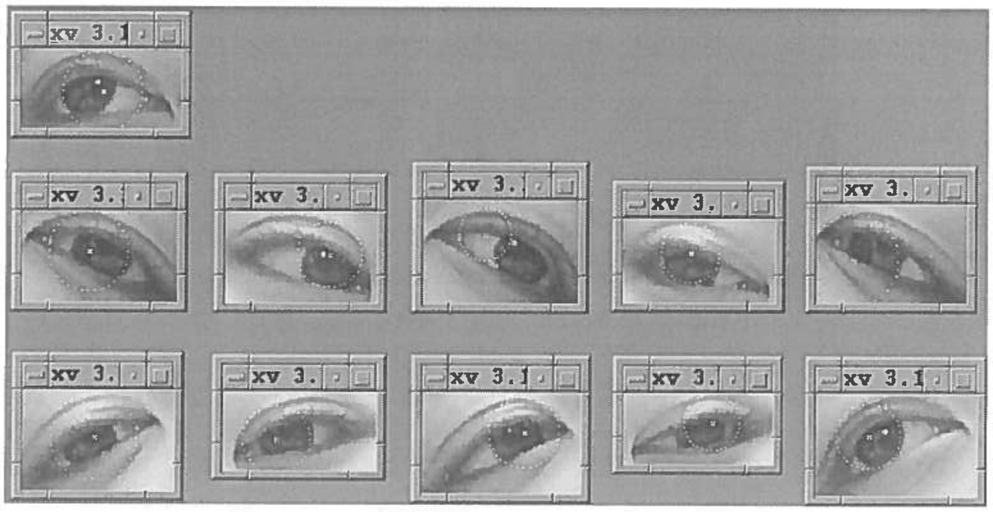


Figure 3.12: *RANSAC Results (2)*

Eye: right Rotation: none Point: middle-right				
Eye: left Rotation: left Point: bottom-centre	Eye: right Rotation: left Point: bottom-left	Eye: left Rotation: left Point: bottom-left	Eye: right Rotation: left Point: bottom-right	Eye: left Rotation: left Point: bottom-right
Eye: right Rotation: right Point: bottom-centre	Eye: left Rotation: right Point: bottom-centre	Eye: right Rotation: right Point: middle-left	Eye: left Rotation: right Point: bottom-left	Eye: right Rotation: right Point: middle-right

To see if better results could be achieved for these wrong results, the parameters of the Canny operator were modified so more edges could be detected. The parameters were set to *low threshold=1*, *high threshold=5* and

Gaussian width=1. Nevertheless, the results achieved were not very good, because some wrong cases had better circle fitting but the correct cases had the circle misplaced.

From the same 62 images, 53 had worst results, 4 were slightly better and 5 obtained the same result. Some examples are shown in Figure 3.13.

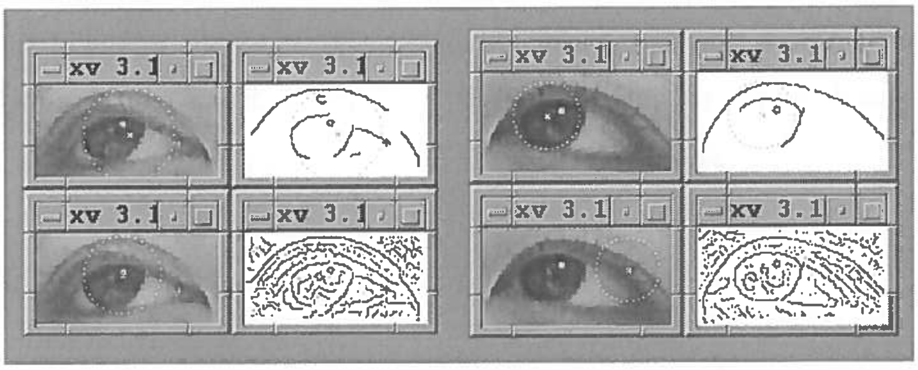


Figure 3.13: *RANSAC Results (3)*

wrong case first result	wrong case edges detected first result	correct case first result	correct case edges detected first result
wrong case second result	wrong case edges detected second result	correct case second result	correct case edges detected second result

3.3 Iris Position Estimation

Once the iris centre is located, other features of the eye should be found that allow the estimation of the position of the iris within the eye. In the first instance, the candidates considered were the eye corners. Some techniques were considered but none of them proved to be reliable.

3.3.1 Edge Detection and Contour Fitting

Following the same principles as for the iris finding, the first idea was to apply edge detection and contour fitting. In this case, the model of the eye was considered like the intersection of two circles, as displayed in Figure 3.14.

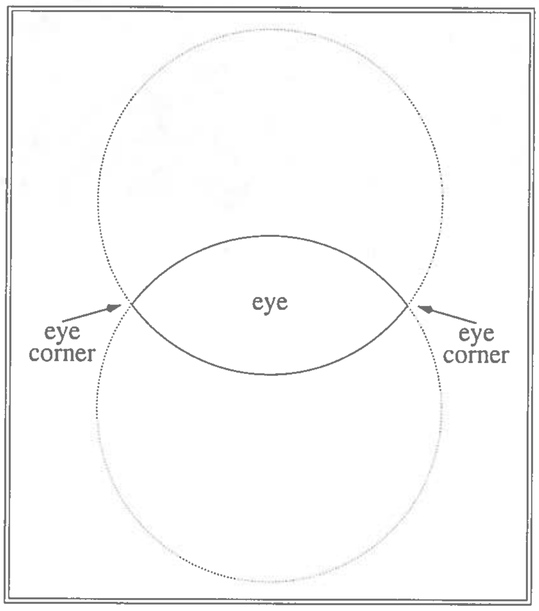


Figure 3.14: *Eye Model*

This method would discard the edge points belonging to the iris and try to fit circles for the upper and lower arcs. Then, the eye corners would be calculated as the intersections of both circles. However, when the actual images of the eyes were analysed this was not the case, specially with the lower arc, as seen in some examples in Figure 3.15.

Even if this idea was followed, the selection of the model for the eye arcs is not a trivial task, because the curve shape varies due to many factors like eye shape, eye location with respect to the camera, head rotation, eyelids position, etc. Therefore, this approach was discarded.

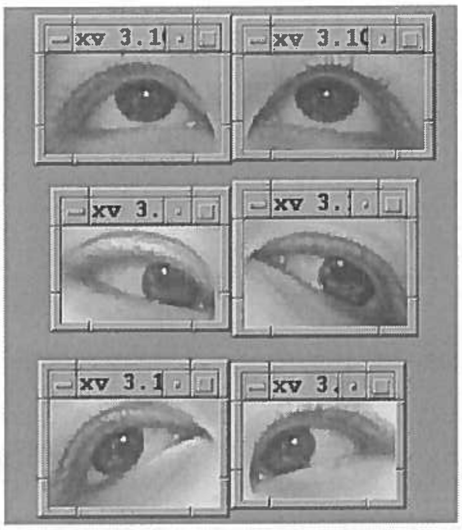


Figure 3.15: *Eye Images*

3.3.2 Region Segmentation

The second approach was to perform region segmentation. In this case, the objective was to find the sclera (the white of the eye) based on the difference between the gray level values, since the iris is darker than the sclera.

The problem in this technique was to find reasonable gray level values to perform the segmentation by thresholding, because some parts of the skin surrounding the eye have similar gray levels to the sclera.

Using Visilog, some thresholding tests were done to get an idea of the parameters that could determine if a pixel belongs to the region or not. On the one hand, if the lower limit of the threshold was stopped before the pixels belonging to the skin merged with the pixels of the sclera, the region obtained covered a very small part of the sclera. On the other hand, if the lower threshold was reduced until most part of the sclera was detected, a great part of the pixels belonging to the skin were included in the region seeked. Some of the results are shown in Figure 3.16.

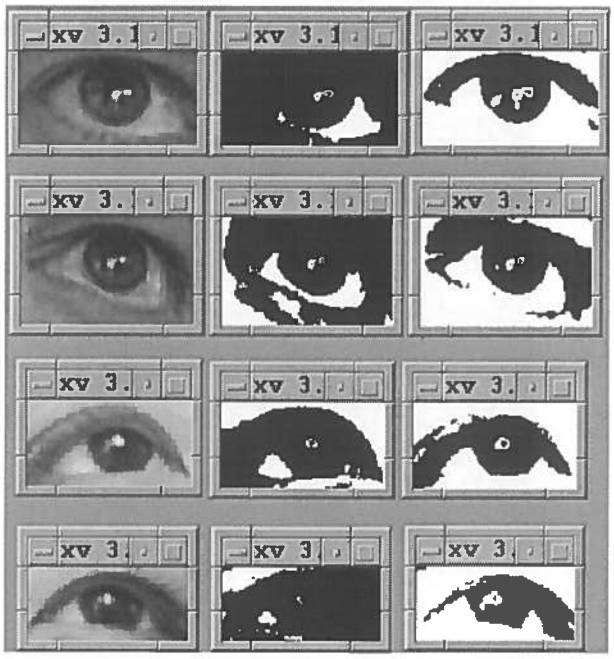
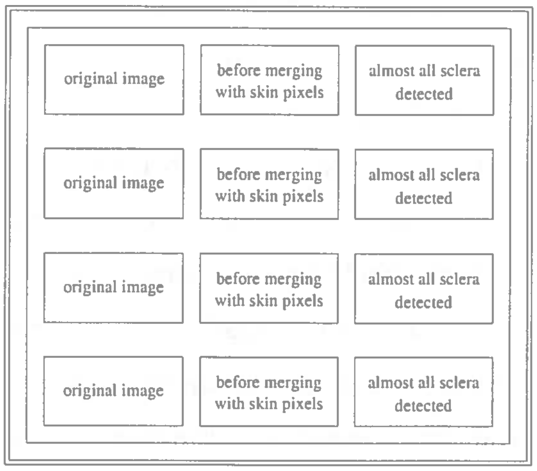


Figure 3.16: *Thresholding*


original image
before merging with skin pixels
almost all sclera detected
original image
before merging with skin pixels
almost all sclera detected
original image
before merging with skin pixels
almost all sclera detected
original image
before merging with skin pixels
almost all sclera detected

3.3.3 Bounding Box Finding

The process to find the eye corners proved to be too complicated if simple image processing techniques were used, so a different method was tried. In this case, the objective was to find a bounding box around the eye that could remain in the same position, independently of the eye characteristics, like size, rotation, iris location, etc.

The steps followed in this approach were:

- Knowing the iris centre and radius from the previous process, the pixel with the maximum gray level was selected, from those at a specified distance in pixels from the iris (10 pixels), as displayed in Figure 3.17.
- Starting from the iris border, rays were projected from the centre to the border of the image, as seen in Figure 3.18, under the constraints:
    * The ray end must be within the eye image.
    * The projection of the ray will stop if the current pixel value is lower than an established threshold below the maximum gray level pixel.
    * The projection of the ray will stop if the current pixel is labelled as an edge pixel by the iris edge detection process.
- A bounding box was then fitted around the termination points of the rays.

The problems encountered in this method were that the resulting bounding box moves with the iris position and it produces reasonable results only with unrotated eyes. Some examples are shown in Figure 3.19.

Since there is not a reliable technique to detect additional eye features, the process will have to assume that other process will provide the location of the eye corners. For the purposes of the reminder of the project the location will be found manually.

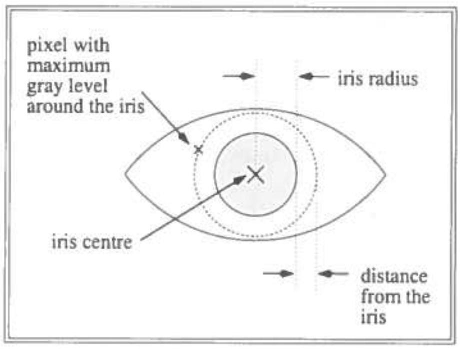


Figure 3.17: *Maximum Gray Level Pixel Selection*

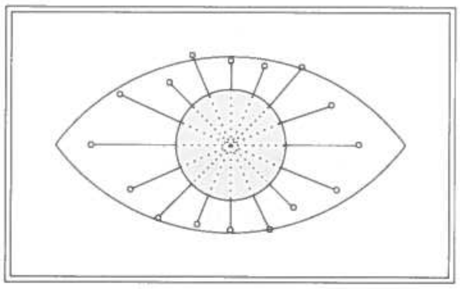


Figure 3.18: *Ray Projections*

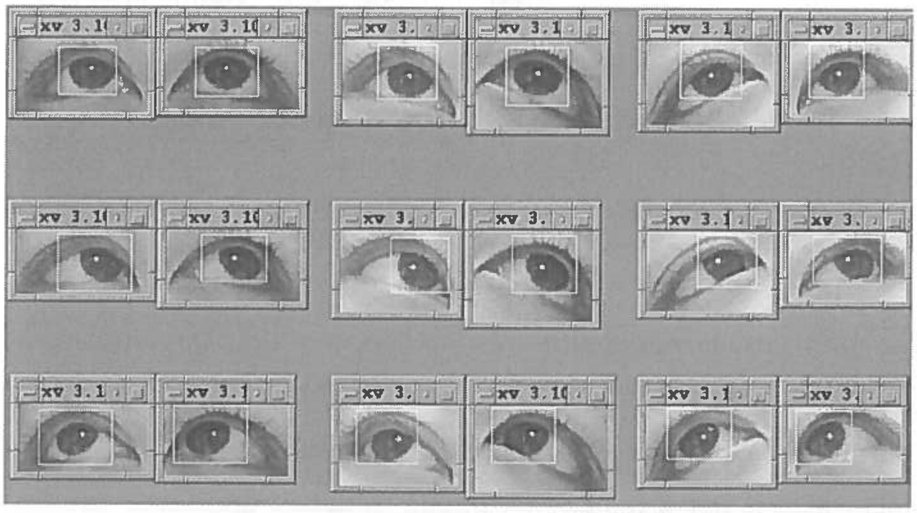
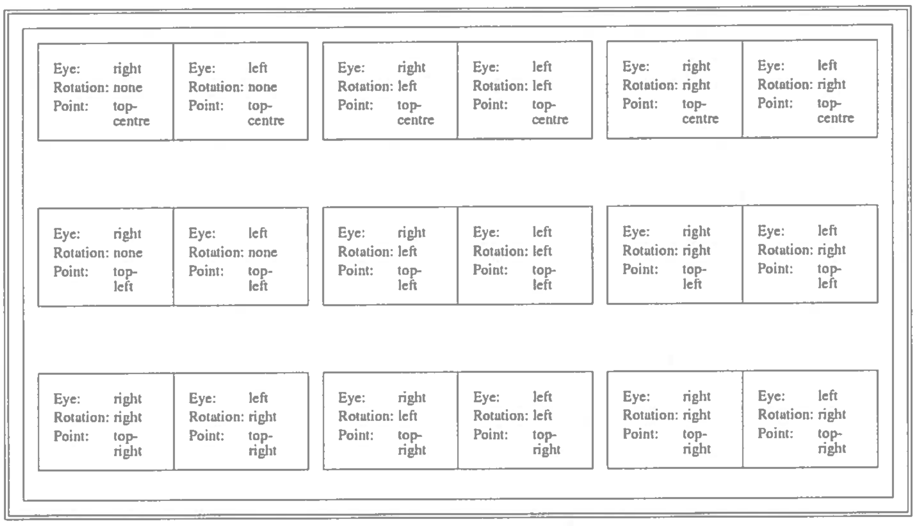


Figure 3.19: *Bounding Box Results*


Eye: right Rotation: none Point: top-centre
Eye: left Rotation: none Point: top-centre
Eye: right Rotation: left Point: top-centre
Eye: left Rotation: left Point: top-centre
Eye: right Rotation: right Point: top-centre
Eye: left Rotation: right Point: top-centre
Eye: right Rotation: none Point: top-left
Eye: left Rotation: none Point: top-left
Eye: right Rotation: left Point: top-left
Eye: left Rotation: left Point: top-left
Eye: right Rotation: right Point: top-left
Eye: left Rotation: right Point: top-left
Eye: right Rotation: right Point: top-right
Eye: left Rotation: right Point: top-right
Eye: right Rotation: left Point: top-right
Eye: left Rotation: left Point: top-right
Eye: right Rotation: right Point: top-right
Eye: left Rotation: right Point: top-right

3.4 Discussion

The process for iris finding does not guarantee perfect results, as shown in the tests performed, where not all the images got a correct circle placement. The success of this process depends on the efficiency of the edge detection and the contour fitting phases.

In edge detection, the Canny algorithm worked well, but its performance relies on the quality of the source image. The usual care recommended for any edge detection process should be taken, like illumination, image contrast, noise presence, image resolution, etc.

The phase of contour fitting involves randomness, so the possibility of getting a bad result cannot be discarded or underestimated. The edge detection images used as a source should contain the least possible amount of noise. The percentage of correct cases was 80.65%, of slightly shifted cases was 8.06% and of completely misplaced cases was 11.29%.

With regard to the search for additional eye features to estimate the iris position within the eye, after trying all the explained techniques, none of them worked adequately.

Some approaches that might deal with this problem would require the use of more complex techniques. One of these approaches is the analysis of colour images, as suggested by Dr. Robertson. Since the sclera and the skin have different colours, the problems encountered in region segmentation due the similarity of gray levels could be overcome.

Other technique used in some of the works reviewed in Chapter 2 was the detection of eye features as part of a face tracking system based on template matching techniques. The disadvantage of this method is that it required special equipment, like the project in [Heinzmann & Zelinsky 98]. But if a global face model was used, the eye position would be known and the iris position could be then reported, relative to the face model.

3.5 Conclusions

Like any image processing task, the success of the overall process depends on the acquisition of the image. In the case of this project, if the image does not have good illumination, is not properly focused or has a low resolution, it will be very difficult for the remaining processes to do their work correctly, since the edge detection relies on the image acquisition and the contour fitting relies on the edge detection.

In relation to feature finding, if the feature considered can be modelled accurately, like the iris as a circle, the process to find it is fairly straightforward and the results are good enough. But if the feature cannot be modelled, because it changes due many factors, like the eye arcs with the view point, orientation, eye shape, etc., the process to find it will require the use of complex techniques and will be a very time-consuming task.

It is not surprising now that, in the review done in Chapter 2, there were few projects performing feature finding with the accuracy required for the task under consideration, except for features like the iris or the pupil.

Chapter 4

Location of the Fixation Point

This chapter explains the process followed to calculate the 3D fixation point. The steps used are: *(a)* find the gaze vectors, *(b)* find the eyes in the scene, *(c)* find the view rays, and *(d)* project the fixation point onto the screen. A brief description of stereo matching is also included, since the determination of the fixation point will use the information provided by both eyes.

4.1 Stereo Matching

One important task in a computer vision system is the calculation of the distance from various points in a scene to the camera position ([Jain *et al.* 95]). A common method to estimate depth is binocular stereo. which uses a pair of images, obtained from two cameras, which are separated by a known distance.

The simplest model, as seen in Figure 4.1, consists of two cameras separated by a baseline distance b in the x direction. The image planes of the camera are considered coplanar.

Placed at different positions in the image plane, the cameras view the same feature in the scene. **Disparity** is the displacement between the locations of the two features in the image plane. The **epipolar plane** is the plane that passes through the centres of the cameras and the feature point. The **epipolar line** is defined by the intersection between the epipolar plane and the image plane.

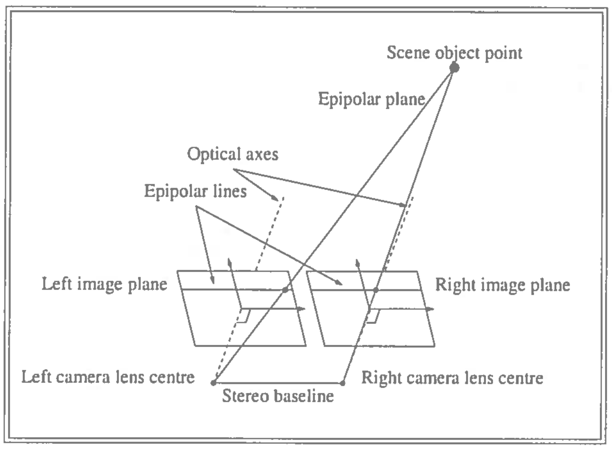


Figure 4.1: *Stereo Matching Model (adapted from [Jain et al. 95])*

In the project, the two cameras are actually the two eyes. Because the eyes can rotate, the principle of parallel optical axes does not apply. The objective is to find the point that is closer to both view rays.

4.2 Finding the Gaze Vectors

4.2.1 Eye Axes

The process begins with the analysis of the image of the face taken by the camera. Figure 4.2 shows a schematic view of one eye. The image coordinate system has its origin at the left-top corner of the image, with axes $\vec{i}$ and $\vec{j}$.

The coordinate system of the eye has its origin at (i_0, j_0), with axes $\vec{u}$ and $\vec{v}$. (i_0, j_0) is also the coordinate of one eye corner and (i_e, j_e) is the coordinate of the other corner.

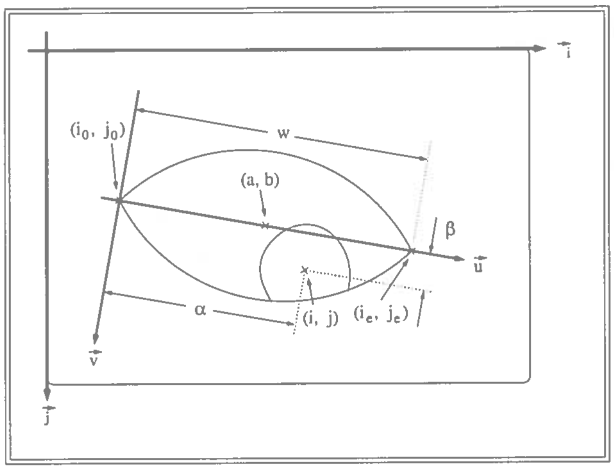


Figure 4.2: *Front View of the Eye*

If (i_0, j_0) and (i_e, j_e) are assumed as known, the rotation and translation of the eye with respect to the image can be estimated by the calculation of the eye axes, $\vec{u}$ and $\vec{v}$, as follows:

$$\vec{u} = (u_x, u_y) = \frac{(i_e - i_0, j_e - j_0)}{\|(i_e - i_0, j_e - j_0)\|}$$

$$\vec{v} = (v_x, v_y) = (-u_y, u_x)$$

4.2.2 Eye Centre

(a, b) are the coordinates of the eye centre, which can be obtained by:

$$\begin{aligned} a &= \frac{\mid i_e + i_0 \mid}{2} \\ b &= \frac{\mid j_e + j_0 \mid}{2} \end{aligned}$$

4.2.3 Eye Width

The eye width w can be estimated as follows:

$$w = \|(i_e - i_0, j_e - j_0)\|$$

4.2.4 Iris Centre

(α, β) is the scale invariant position of the iris centre with respect to the eye origin, (i_0, j_0), as seen in Figure 4.3. These values can be stated as:

$$\alpha = (\alpha_x, \alpha_y) = \frac{(i - i_0, j - j_0) \cdot \vec{u}}{w}$$

$$\beta = (\beta_x, \beta_y) = \frac{(i - i_0, j - j_0) \cdot \vec{v}}{w}$$

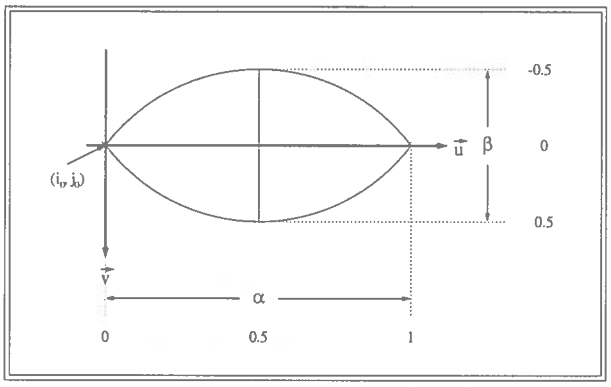


Figure 4.3: *Location of the Iris Centre on the Image*

4.3 Finding the Eyes in the Scene

Figure 4.4 shows a schematic diagram of the scene, in the (x, z) plane. $\vec{c}$ is the camera position, λ is the distance between the eye and the camera, (a, b) are the coordinates of the eye centre obtained in section 4.2.2, $\vec{V}$ is the camera ray passing through the eye centre (which depends on a and b), and Δ is the distance between the centres of the eyes.

The values of $\vec{V}$ are calculated by the camera calibration process, relating the points in the 3D scene with the pixels in the image.

The coordinates of the centres of the eyes, $\vec{e_l}$ and $\vec{e_r}$, can be calculated by:

$$\vec{e_l} = (e_{lx}, e_{ly}, e_{lz}) = \vec{c} + \lambda_l \vec{V_l}(a_l, b_l) \tag{4.1}$$

$$\overrightarrow{e_r} = (e_{rx}, e_{ry}, e_{rz}) = \overrightarrow{c} + \lambda_r \overrightarrow{V_r}(a_r, b_r) \tag{4.2}$$

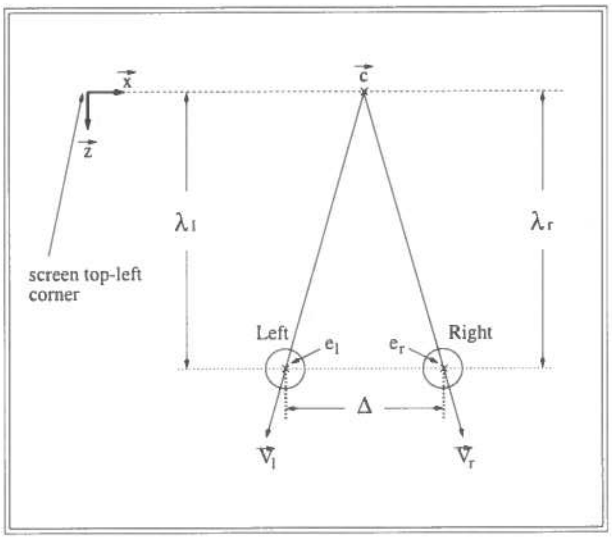


Figure 4.4: *Location of the Eye on the Scene*

Assuming that the face is in a fronto-parallel position (*i.e.* the plane in the z coordinate through the centres of the eyes is parallel to the plane through the camera), and that $\overrightarrow{c}$ and $\overrightarrow{V}$ are known, equations (4.1) and (4.2) can be restated as:

$$\lambda_l V_{lz} = \lambda_r V_{rz} \tag{4.3}$$

$$(\lambda_l V_{lx} - \lambda_r V_{rx})^2 + (\lambda_l V_{ly} - \lambda_r V_{ry})^2 = \Delta^2 \tag{4.4}$$

From equation (4.3), λ_l can be obtained by:

$$\lambda_l = \frac{\lambda_r V_{rz}}{V_{lz}} \tag{4.5}$$

Substituting equation (4.5) in equation (4.4):

$$\left(\frac{\lambda_r V_{rz}}{V_{lz}} V_{lx} - \lambda_r V_{rx}\right)^2 + \left(\frac{\lambda_r V_{rz}}{V_{lz}} V_{ly} - \lambda_r V_{ry}\right)^2 = \Delta^2$$

λ_r is estimated by:

$$\lambda_r = \sqrt{\frac{\Delta^2}{\left(\frac{V_{rz}V_{lx}}{V_{lz}} - V_{rx}\right)^2 + \left(\frac{V_{rz}V_{ly}}{V_{lz}} - V_{ry}\right)^2}}$$

To simplify the notation:

$$D = \left(\frac{V_{rz}V_{lx}}{V_{lz}} - V_{rx}\right)^2 + \left(\frac{V_{rz}V_{ly}}{V_{lz}} - V_{ry}\right)^2$$

therefore:

$$\lambda_r = \sqrt{\frac{\Delta^2}{D}} \tag{4.6}$$

Substituting equation (4.6) in equation (4.5):

$$\lambda_l = \frac{\sqrt{\frac{\Delta^2}{D}}V_{rz}}{V_{lz}} \tag{4.7}$$

To find the centres of the eyes, equations (4.6) and (4.7) are substituted in equations (4.1) and (4.2):

$$\begin{aligned}\overrightarrow{e_l} &= \overrightarrow{c} + \frac{\sqrt{\frac{\Delta^2}{D}}V_{rz}}{V_{lz}}\overrightarrow{V_l}(a_l, b_l) \\ \overrightarrow{e_r} &= \overrightarrow{c} + \sqrt{\frac{\Delta^2}{D}}\overrightarrow{V_r}(a_r, b_r)\end{aligned}$$

4.4 Finding the View Rays

As seen in Figure 4.5, the fixation point can be stated as:

$$\overrightarrow{f_l} = \overrightarrow{e_l} + \mu_l \overrightarrow{d_l}(\alpha_l, \beta_l) \tag{4.8}$$

$$\overrightarrow{f_r} = \overrightarrow{e_r} + \mu_r \overrightarrow{d_r}(\alpha_r, \beta_r) \tag{4.9}$$

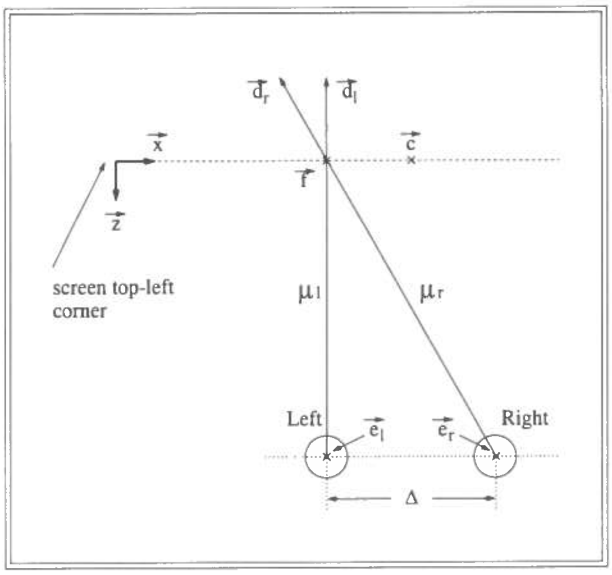


Figure 4.5: *Fixation Point*

The values of $\overrightarrow{d}$ depend on the values of the gaze vector, $(\alpha,\ \beta)$, calculated in section 4.2.4. These values must be customised for each user by a user calibration process, relating each position of the iris centre with the corresponding gaze vector.

To calculate the fixation point, the distances μ_l and μr must be found, such that they minimise:

$$F = \|\overrightarrow{f_l} - \overrightarrow{f_r}\|^2 = (\overrightarrow{f_l} - \overrightarrow{f_r})^T(\overrightarrow{f_l} - \overrightarrow{f_r}) \tag{4.10}$$

which is the point where the two view rays are closest. Theoretically speaking, these rays should intersect, but due to noise they will often just pass closely. To minimise F:

$$\begin{aligned} \frac{\partial F}{\partial \mu_l} &= 0 \\ \frac{\partial F}{\partial \mu_r} &= 0 \end{aligned}$$

$$\frac{\partial F}{\partial \mu_l} = 2\overrightarrow{f_l}^T \frac{\partial \overrightarrow{f_l}}{\partial \mu_l} - 2\overrightarrow{f_r}^T \frac{\partial \overrightarrow{f_l}}{\partial \mu_l} \tag{4.11}$$

$$\frac{\partial F}{\partial \mu_r} = 2\overrightarrow{f_r}^T \frac{\partial \overrightarrow{f_l}}{\partial \mu_r} - 2\overrightarrow{f_l}^T \frac{\partial \overrightarrow{f_r}}{\partial \mu_l} \tag{4.12}$$

Substituting equations (4.8) and (4.9) in equations (4.11) and (4.12):

$$\overrightarrow{e_l}^T\overrightarrow{d_l} + \mu_l\overrightarrow{d_l}^T\overrightarrow{d_l} - \overrightarrow{e_r}^T\overrightarrow{d_l} - \mu_r\overrightarrow{d_r}^T\overrightarrow{d_l} = 0 \tag{4.13}$$

$$\overrightarrow{e_r}^T\overrightarrow{d_r} + \mu_r\overrightarrow{d_r}^T\overrightarrow{d_r} - \overrightarrow{e_l}^T\overrightarrow{d_r} - \mu_l\overrightarrow{d_l}^T\overrightarrow{d_r} = 0 \tag{4.14}$$

As

$$\begin{aligned}\overrightarrow{d_l}^T\overrightarrow{d_l} &= \overrightarrow{d_r}^T\overrightarrow{d_r} = 0\\ \overrightarrow{d_l}^T\overrightarrow{d_r} &= \overrightarrow{d_r}^T\overrightarrow{d_l} = 1\end{aligned}$$

μ_r and μ_l can be obtained from Equations (4.13) and (4.14) by:

$$\mu_r = \frac{(\overrightarrow{e_l}^T\overrightarrow{d_l} + \overrightarrow{e_r}^T\overrightarrow{d_l})(\overrightarrow{d_l}^T\overrightarrow{d_r}) + (\overrightarrow{e_r}^T\overrightarrow{d_r} - \overrightarrow{e_l}^T\overrightarrow{d_r})}{(\overrightarrow{d_l}^T\overrightarrow{d_r})^2 - 1} \tag{4.15}$$

$$\mu_l = \frac{(\overrightarrow{e_r}^T\overrightarrow{d_r} - \overrightarrow{e_l}^T\overrightarrow{d_r})(\overrightarrow{d_l}^T\overrightarrow{d_r}) + (\overrightarrow{e_l}^T\overrightarrow{d_l} + \overrightarrow{e_r}^T\overrightarrow{d_l})}{(\overrightarrow{d_l}^T\overrightarrow{d_r})^2 - 1} \tag{4.16}$$

To obtain the fixation points Equations (4.15) and (4.16) are substituted in Equations (4.8) and (4.9).

The fixation point is obtained by:

$$\overrightarrow{f} = \frac{\overrightarrow{f_l} + \overrightarrow{f_r}}{2}$$

4.5 Projection of the Fixation Point onto the Screen

4.5.1 Fixation Point Coordinates Transformation

Once the fixation point $\overrightarrow{f}$ is found, it should be projected onto the screen (*i.e.* transformed into screen coordinates), as seen in Figure 4.6. Assuming that the camera vector, $\overrightarrow{c}$, is known, an homogeneous matrix can be applied to $\overrightarrow{f}$ [Fisher 98].

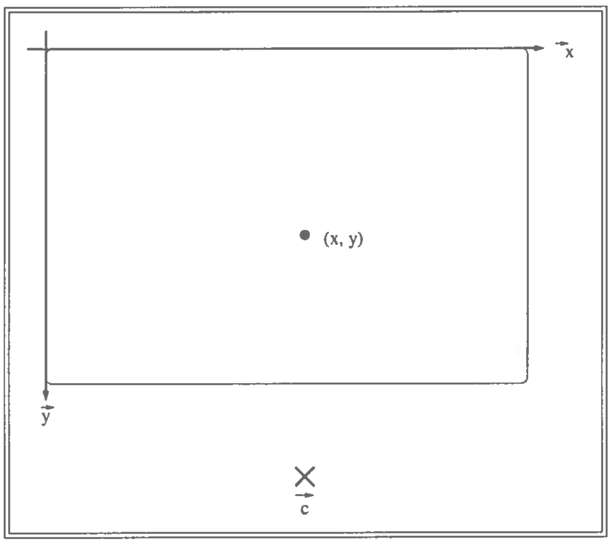


Figure 4.6: *Projection of the Fixation Point onto the Screen*

The steps involved in this process are: first, the point $\overrightarrow{f} = (f_x, f_y, f_z)^T$ is augmented to $\overrightarrow{f} = (f_x, f_y, f_z, 1)^T$. Then, the homogeneous matrix Π is formed by the rotation matrix, R, and the translation vector T:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

$$T = (T_x, T_y, T_z)^T$$

$$\Pi = \begin{pmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finally, the new point is estimated by:

$$\begin{aligned} \overrightarrow{G} &= \Pi \overrightarrow{f} \\ \overrightarrow{G} &= (M_x, M_y, M_z)^T \end{aligned}$$

where:

$$\begin{aligned} M_x &= r_{11} f_x + r_{12} f_y + r_{13} f_z + T_x \\ M_y &= r_{21} f_x + r_{22} f_y + r_{23} f_z + T_y \\ M_z &= r_{31} f_x + r_{32} f_y + r_{33} f_z + T_z \end{aligned}$$

4.5.2 Error Estimation

The true coordinates of the fixation point in the screen are $(\rho_x \cdot r, -\rho_y \cdot s, 0)$, where ρ is the scaling factor of the screen, (*i.e* the equivalence of the pixels in the screen in metric units). Therefore, two error measurements can be stated:

$$\begin{aligned} depth\_error_1 &= M_z \\ pixel\_error_2 &= \left\| \left(\frac{M_x}{\rho_x} - r, \frac{M_y}{\rho_y} - s \right) \right\| \end{aligned}$$

4.6 Experiments

Once the user calibration was made, as explained in Appendix B, nine different images were taken to test the fixation point calculation process. The user was in a nearly fronto-parallel position, but with no restrictions in the movements of the head, as in the case with the user calibration.

The steps followed in the process are:

- Image transformation from colour bitmap to gray-scale PGM using XV.
- Eyes location by hand and cropping by program.
- Iris centre calculation following the process explained in Chapter 3.
- Gaze vectors estimation as stated in Section 4.2.
- Eye centres finding as exposed in Section 4.3.
- View rays calculation as explained in Section 4.4.
- Fixation point projection onto the screen as stated in Section 4.5.

To evaluate the performance of the fixation point calculation, the true coordinates of the fixation point on the screen were measured and compared with the results obtained.

The performance was not quite good. There is not a systematic or constant error that could be identified, *e.g.* a larger error in points located on the sides than in the points on the centre of the screen, or a constant error due to noise.

The reason of these errors can be attributed to the user calibration results. The points used for the calibration were not enough and no interpolation was performed to obtain the missing values.

Nevertheless, some results obtained were close to their true location. The best result had a difference of *34* millimetres in the x axis, *5* millimetres in the y axis, and *66* millimetres in the z axis. The worst result had a difference of *153* millimetres in the x axis, *192* millimetres in the y axis, and *176* millimetres in the z axis. The complete list of results is shown in Tables 4.1 and 4.2, and a schematic figure of the points used can be seen in Figure 4.7. The coordinates units and the depth error are in millimetres. The pixel error is in pixels.

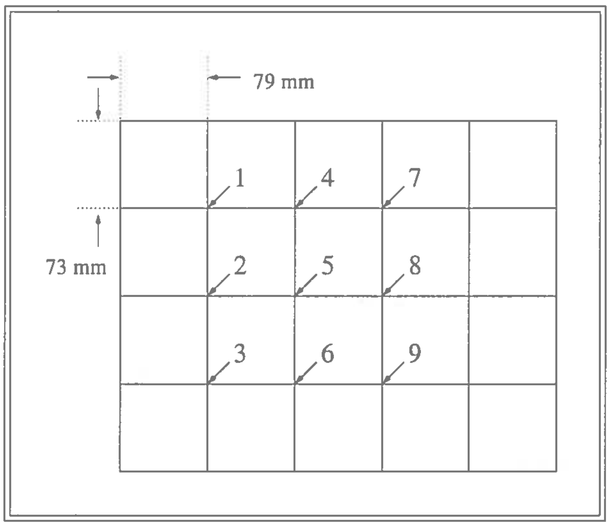


Figure 4.7: *Pattern Points Used on Experiments*

With these large variations over the results is not easy to evaluate the overall performance of the system, but as stated before, the source that contributed more to the errors is the user calibration process.

4.7 Discussion

The calculation of the fixation point is a straightforward process and it should produce good results if the processes that provide the source information are accurate. Nevertheless, as seen in the experiments, if any process fails to produce its results with the accuracy required, the overall process feels the effects of that failure.

In this case, the process that affected the calculation was the user calibration process. As stated before, there was not enough information available, since not enough points were used to calculate the corresponding gaze vectors and no interpolation was made to estimate the missing values.

Two alternatives can correct this error. The first one is to repeat the user calibration process. The problem with this choice is that it is a tiring process for the user. If any application wants to use the output of this process, a good choice suggested by Dr. Robertson is to consider more points on the centre of the screen and less points on the sides of the screen since the users generally focus more their attention on the centre than on the sides of the screen.

The second alternative is to perform interpolation to estimate the missing values. The drawback of this method is that the accuracy will not be as good as using real data.

Point	x true	x calculated	y true	y calculated	z true	z calculated
1	79	232	73	265	0	-163
2	79	183	146	254	0	-176
3	79	191	219	219	0	-81
4	158	205	73	208	0	-21
5	158	195	146	298	0	-28
6	158	192	219	214	0	-66
7	237	201	73	194	0	-20
8	237	188	146	192	0	-17
9	237	183	219	218	0	-95

Table 4.1: *Results of the Fixation Point Calculation*

Point	x error	y error	z error/depth error	pixel error
1	153	192	-163	835
2	104	108	-176	505
3	112	0	- 81	359
4	47	135	- 21	498
5	37	62	- 28	249
6	34	- 5	- 66	109
7	-36	121	- 20	441
8	-49	46	- 17	228
9	-54	- 1	- 95	177

Table 4.2: *Errors of the Fixation Point Calculation*

4.8 Conclusions

Similarly to the conclusions stated in Chapter 3, the success of the overall process depends on the success of its components, since the accuracy of each estimation depends on the accuracy of the preceding estimation.

The process is still very time consuming, because some processes should be done by hand (image transformation, eye finding, corner finding, partial results checking of the camera calibration and user calibration processes and scene measurements).

The results obtained are not good, but if additional work is done on the components, great improvements can be made.

Chapter 5

Conclusions

The estimation of the gaze direction using the eyes direction produces more accurate measurements than using the face direction, as seen in the research already done in face image processing reviewed in Chapter 2.

The projects developed based on the face direction give an estimate of the gaze movement, but they cannot give a precise location of the fixation point. Nevertheless, these efforts produced good results for the applications they were intended for.

When more accurate estimations of the fixation point were needed, the projects used the eyes direction as the indicator of the gaze direction. The results obtained were good, but the major drawback of many of the projects is that they required special equipment or were user-intrusive.

This project investigated the estimation of the fixation point, based on the information provided by both eyes, without using special equipment or being user-intrusive.

As seen in Chapter 3, the success on the iris location, like any image processing task, relies on the image acquisition. If the image quality is not good, no approach will work, no matter which technique is used.

The finding of face features is not a trivial task. In the case of the iris, the process was fairly straightforward and the results were good enough because the model used was accurate enough. But regarding the eye corners, the process did not work even though different techniques were tested.

The fixation point calculation was not as accurate as expected, as shown in Chapter 4. The success of the overall process depends on the success of its components since each stage depends on the results obtained by its predecessor. Consequently, any failure in any component will affect the results obtained. But it also means that as more work is done on each component to produce better results, the accuracy of the process will be better.

Some parts of the process are still done by hand and there is not an automated flow between the components. Therefore, there are many aspects of this project that can be enhanced if the aim is to use the result in any application like the ones mentioned in Chapter 2.

Bibliography

[Ashbrook 99] A. P. Ashbrook. Shadow striper tutorial. Machine Vision Unit. Artificial intelligence. University of Edinburgh, March 1999.

[Bala *et al.* 97] L. P. Bala, K. Talmi, and J. Liu. Automatic detection and tracking of faces and facial features in video sequences. *NTG-Fachberichte*, (143):251–256, 1997.

[Baluja & Pomerleau 94] S. Baluja and D. Pomerleau. Non-intrusive gaze tracking using artificial neural networks. Technical Report CMU-CS-94-102, Carnegie Mellon University, School of Computer Science, January 1994.

[Black & Yacoob 95] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *5th International conference on Computer vision*, pages 374–381. IEEE Computer Society, IEEE Computer Society Press, 1995.

[Burl *et al.* 95]	M. C. Burl, T. K. Leung, and P. Perona. Face localization via shape statistics. In M. Bichsel, editor, *International Workshop on Automatic Face- and Gesture-Recognition*, pages 154–159. IEEE Computing Society. Swiss Informaticians Society, IEEE Computer Society Press, 1995.
[Chellappa *et al.* 95]	R. Chellappa, C. L. Wilson, and S. Sirohey. Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 83(5):705–740, 1995.
[Copeland & Trivedi 97]	A. C. Copeland and M. M. Trivedi. Integrated framework for developing search and discrimination metrics. In *Proceedings of SPIE - the International Society for Optical Engineering*, volume 3062, pages 53–58, 1997.
[Fischler & Bolles 81]	M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
[Fisher 98]	R. B. Fisher. Machine vision lecture notes. MSc in Artificial Intelligence Course, 1998.
[Fisher *et al.* 96]	R. B. Fisher, A. Fitzgibbon, A. Gionis, M. Wright, and D. Eggert. A hand-held optical surface scanner for environmental

modeling and virtual reality. In *Virtual Reality World*, 1996.

[Fromherz *et al.* 97] T. Fromherz, Strucki, and M. Bichsel. A survey of face recognition. Technical Report MML TR 97.01, University of Zurich, Department of Computer Science, MultiMedia Laboratory, 1997.

[Gee & Cipolla 94a] A. Gee and R. Cipolla. Determining the gaze of faces in images. Technical Report CUED/F-INFENG/TR174, University of Cambridge. Department of Engineering, March 1994.

[Gee & Cipolla 94b] A. Gee and R. Cipolla. Non-intrusive gaze tracking for human-computer interaction. In *International Conference on Mechatronics and machine vision in practice*, pages 112–117. IEEE, University of Southern Queensland, 1994.

[Gee & Cipolla 95] A. Gee and R. Cipolla. Fast visual tracking by temporal consensus. Technical Report CUED/F-INFENG/TR207, University of Cambridge, Department of Engineering, February 1995.

[Heinzmann & Zelinsky 97] J. Heinzmann and A. Zelinsky. Robust real-time face tracking and gesture recognition. In *15th International Joint Conference on Artificial Intelligence*, volume 2, pages

1525–1530. Morgan Kaufmann Publishers, 1997.

[Heinzmann & Zelinsky 98] J. Heinzmann and A. Zelinsky. 3-d facial pose and gaze point estimation using a robust real-time tracking paradigm. In *3rd International Conference on Automatic Face and Gesture Recognition*, pages 142–147. IEEE Computer Society, IEEE Computer Society Press, 1998.

[Horprasert *et al.* 96] T. Horprasert, Y. Yacoob, and L. Davis. Computing 3-d head orientation from monocular image sequence. In *2nd International Conference on Face and Gesture Recognition*, pages 242–247. IEEE, 1996.

[Jain *et al.* 95] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision.* McGraw-Hill, Inc., 1995.

[Jebara & Pentland 97] T. S. Jebara and A. Pentland. Parametrized structure from motion for 3d adaptive feedback tracking of faces. In *Conference on Computer Vision and Pattern Recognition*, pages 144–150. IEEE Computer Society, IEEE Computer Society Press, 1997.

[Krüger *et al.* 97] N. Krüger, M. Pötzsc, and C. von der Malsburg. Determination of race position with a learned representation based on labelled graphs. *Image and Vision Computing*, 15:665–673, 1997.

[Liu 98]	J. Liu. Determination of the point of fixation in a head-fixed coordinate system. In A. K. Jain, S. Venkatesh, and B. C. Lovell, editors, *14th International Conference on Pattern Recognition*, pages 501–504. IEEE Computer Society, IEEE Computer Society Press, 1998.
[Maurer & von derMalsburg 96]	T. Maurer and C. von der Malsburg. Tracking and learning graphs and pose on image sequences of faces. In *2nd Second International Conference on Automatic Face and Gesture Recognition*, pages 176–181. IEEE Computer Society, IEEE Computer Society Press, 1996.
[McKenna *et al.* 97]	S. J. McKenna, S. Gong, R. P. Würtz, J. Tanner, and D. Banin. Tracking facial feature points with gabor wavelets and shape models. In J. Bigün, G. Chollet, and G. Borgefors, editors, *1st International Conference on Audio- and Video-Based Biometric Person Authentication*, volume 1026 of LNCS, pages 35–42. Springer-Verlag, 1997.
[Pappu & Beardsley 98]	R. Pappu and P. Beardsley. A qualitative approach to classifying gaze direction. In *3rd International Conference on Automatic Face and Gesture Recognition*. IEEE, 1998.

[Parker 97]	J. R. Parker. *Algorithms for Image Processing and Computer Vision.* John Wiley and Sons, 1997.
[Rowley *et al.* 96]	H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In D. S. Touretsky, M. C. Mozer, and M. E. Hasselmo, editors, *9th Conference on Advances in Neural Information Processing*, pages 875–881. MIT Press, 1996.
[Salvucci 99]	D. D. Salvucci. Inferring intent in eye-based interfaces: Tracing eye movements with process models. In *Conference on Human Factors in Computing Systems (CHI 99)*, 1999.
[Samal & Iyengar 92]	A. Samal and P. A. Iyengar. Automatic recognition and analysis of human faces and facial expressions: a survey. *Pattern Recognition*, 25(1):65–77, 1992.
[Scassellati 98]	B. Scassellati. Eye finding via face detection for a foveated, active vision system. In *15th National Conference on Artificial Intelligence*, pages 969–976. American Association for Artificial Intelligence, AAAI Press, 1998.
[Stiefelhagen *et al.* 97]	R. Stiefelhagen, J. Yang, and A. Waibel. A model-based gaze tracking system. *International Journal on Artificial Intelli-*

gence Tools (Architectures, Languages, Algorithms), 6(2):193–209, June 1997.

[Stiefelhagen *et al.* 98] R. Stiefelhagen, M. Finke, J. Yang, and A. Waibel. From gaze to focus of attention. In *Workshop on Perceptual User Interfaces (PUI98)*, 1998.

[Talmi & Liu 98] K. Talmi and J. Liu. Eye and gaze tracking for visually controlled interactive stereoscopic displays. *Image Communication*, 1998.

[Trucco & Verri 98] M. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[Trucco *et al.* 94] E. Trucco, R. B. Fisher, and A. M. Fitzgibbon. Direct calibration and data consistency in 3-d laser scanning. In E. Hancock, editor, *British Machine Vision Conference*, pages 489–498. British Machine Vision Association, BMVA Press, 1994.

[Valentin *et al.* 94] D. Valentin, H. Abdi, A. O'Toole, and G. W. Cottrell. Connectionist models of face processing: a survey. *Pattern Recognition*, 27(9):1209–1230, 1994.

[Velichkovsky & Hansen 96] B. M. Velichkovsky and J. P. Hansen. New technological windows into mind: There is more in eyes and brains for human-computer interaction. In *Conference on Human factors in computing systems (CHI-96)*. ACM Press, 1996.

[Yang *et al.* 98] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. In R. Chin and T. C. Pong, editors, *3rd Asian Conference on Computer Vision*, volume II, pages 687–694. Springer-Verlag, 1998.

[Young *et al.* 95] D. Young, H. Tunley, and R. Samuels. Specialised hough transform and active contour methods for real-time eye tracking. Technical Report Cognitive Science Research Paper 386, Sussex University, School of Cognitive and Computer Science, July 1995.

[Yow & Cipolla 97a] K. C. Yow and R. Cipolla. Feature-based human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1094–1099. IEEE Computer Society, IEEE Computer Society Press, 1997.

[Yow & Cipolla 97b] K. C. Yow and R. Cipolla. Finding initial estimates of human face location. In *2nd Asian Conference on Computer Vision*, volume 3, pages 514–518, 1997.

Appendix A

Camera Calibration

If a computer vision application needs to determine the position of the objects in 3D space or reconstruct the 3D structure of a scene, it should estimate the relationship between the coordinates of the points in 3D space and the coordinates of the corresponding image points [Trucco & Verri 98].

This relationship is estimated by the camera calibration process, which is one of the inputs for the fixation point estimation. In this chapter the process used for camera calibration is explained, as well as the characteristics of the camera used for the project.

A.1 Camera Calibration Methods

Camera calibration is accomplished by relating the coordinate systems of the points in space and the image points with the camera reference frame, considering two type of parameters:

- Extrinsic parameters. These determine the orientation and location of the camera reference frame in a known world reference frame:
    - * Rotation matrix
    - * Translation vector

- Intrinsic parameters. These associate the coordinates of the camera reference frame with the pixel coordinates of the image:

  * Focal length
  * Location of the image centre in pixel coordinates
  * Effective pixel size in the horizontal and vertical directions
  * Radial distortion coefficient

Two of the methods proposed for camera calibration are: the direct recovering of the parameters, proposed by Tsai, and the estimation of the projection matrix without explicitly solving the parameters, proposed by Faugeras and Toscani. Both methods are explained in detail in [Trucco & Verri 98]. This kind of method requires a precise model of the camera.

One simpler method was used by Dr. Anthony Ashbrook, Research Fellow in the Machine Vision Unit, for a shadow stripper. A detailed explanation of the process, applied to a range finder is in [Trucco *et al.* 94] and applied to a camera used in a range finder in [Fisher *et al.* 96].

The advantage of this method is that it uses an empirical calibration instead of a model. The empirical calibration over-covers any systematic deviations encountered in the normal pinhole camera model.

The ideas used in this method are as follows: a known pattern, like the one seen in Figure A.1, is placed at a known distance z_1 from the camera and a first image is taken. Then, the camera is placed at a known distance z_2 and a second image is taken, as seen in Figure A.2. The pattern has a circle that is lighter than the surrounding circles. The lighter circle in the first image is considered as the origin of the world coordinate system.

The images are thresholded and the centroids of the circles are estimated. With these centroids, the vectors $\overrightarrow{V}$ from the camera to the points on the images are calculated, as seen in Figure A.3. Finally, the camera position $\overrightarrow{c}$

is obtained by a least-squares method, looking for the point that minimises the distances between the vectors $\overrightarrow{V}$.

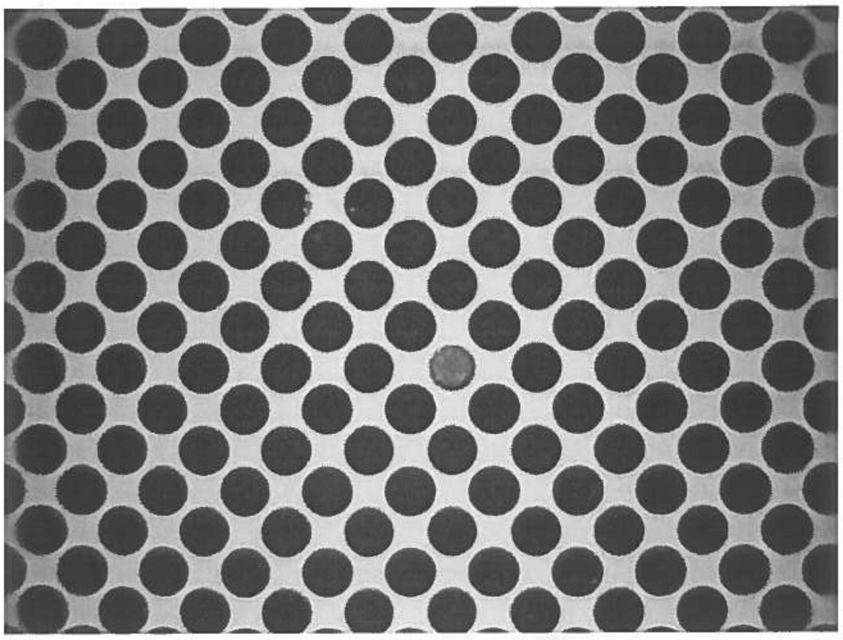


Figure A.1: *Pattern Used for Camera Calibration*

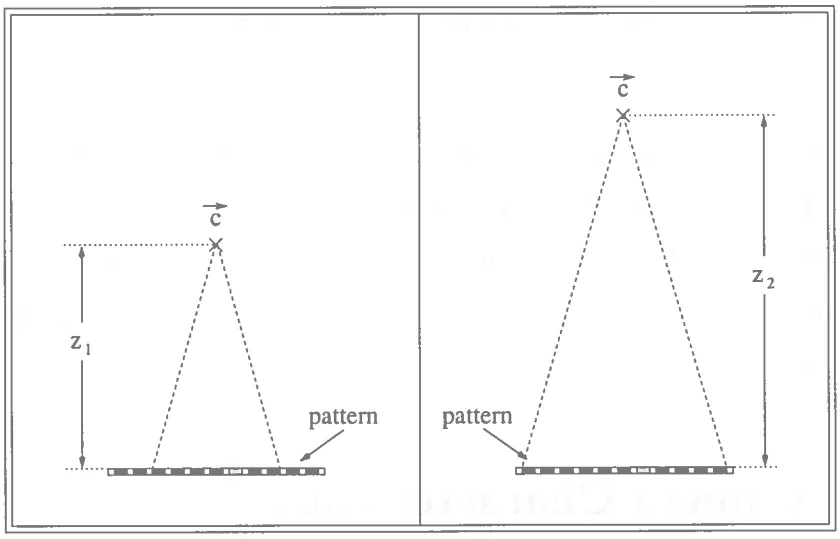


Figure A.2: *Images Acquisition for Camera Calibration*

The 3D position of each centroid of the pattern circles and the camera position $\overrightarrow{c}$ are known, so it is possible to estimate the vector $\overrightarrow{d_i}$ through each centroid of the pattern circles. If linear interpolation is applied to the nearby vectors from the circles, it is possible to calculate the view vector for each pixel in the image.

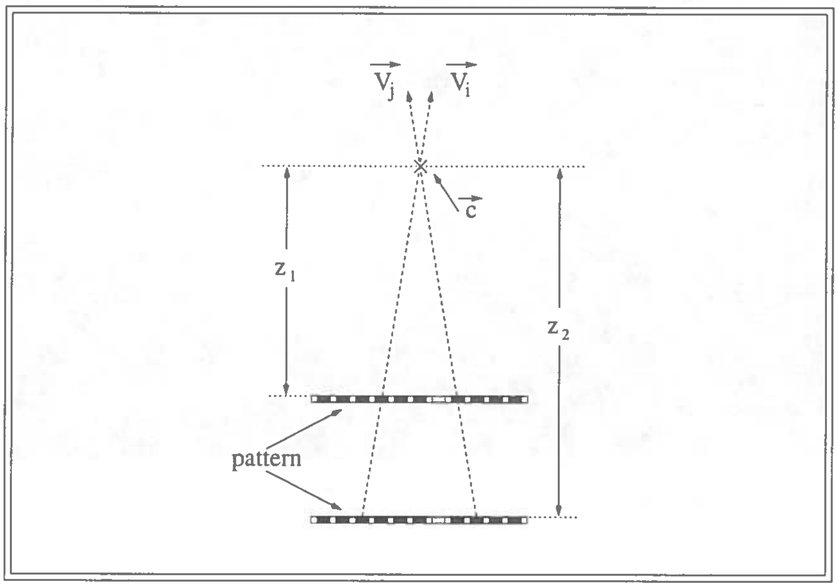


Figure A.3: *Parameters Calculation for Camera Calibration*

The camera calibration file was obtained by a modified version of the software for the Shadow Striper System, used in the Machine Vision Unit, provided by Dr. Ashbrook. The modification was due the difference between the resolution of the cameras. The steps followed in this process are explained in detail in [Ashbrook 99].

A.2 Camera Characteristics

A complete description of the camera characteristics can be found at *http://www.pccam.com/products/kodak/dvc323%20specs.htm*

Kodak DVC323 Video Camera Specifications

Specification	Characteristics
Image Sensor:	Kodak 640(H) x 480(V) pixels, interline transfer, progressive, scan colour CCD with square pixels
Photo Resolution:	640 x 480, 24-bit colour, 16.7 million colours
Colour Video Formats and Resolution:	YUV9: 160 x 120, 320 x 240 YUV12: 160 x 120, 320 x 240 176 x 144 (QCIF), 352 x 288 (CIF)
Lens:	3-element, 6.2 mm focal length lens with f/2.5 aperture
Focus:	Manual adjustment, 5" (12.7 cm) to infinity
Zoom:	Digital, 1X-2X
Field of View:	User selectable: 20, 30, or 42 degrees horizontal

A.3 Camera Parameters

The parameters set on the camera to take the images shown on the experiments are as follows:

- Image size and quality
  - * Zoom: middle point between *telephoto* and *wide*
  - * Video quality
    - ∗ Sharpness: *high*
    - ∗ Frame rate: *low*

  - Finished size: *640 x 480*

- Exposure and colour
  - Exposure
    - Brightness: 3/4 between *dark* and *light*
    - Contrast: middle point between *low* and *high*
  - Shutter speed: *normal*
  - Colour
    - Light source: *auto-balance*
    - Hue: middle point between *green* and *red*
    - Saturation: middle point between *low* and *high*

Appendix B

User Calibration

The process to estimate the gaze vector $\overrightarrow{d}$ based on the scale invariant iris position α and β is done by a direct relationship between those measurements in a user calibration process, which should be customised for each person. It is assumed that the camera is calibrated as stated in Appendix A.

The steps followed in this process are:

- Calibration pattern selection
- Scene set-up
- Images acquisition
- Images processing
- Parameters calculation

B.1 Calibration Pattern Selection

The patterns selected should contain a balanced number of points to where the user has to look at, *i.e.* not too many, so the user does not get tired and not too few, so the accuracy of the calculations is compromised.

For the project, two different patterns were selected and their schematic configuration can be seen in Figure B.1. These patterns gives the possibility of 30 points for the one in the left and 9 points for the one in the right.

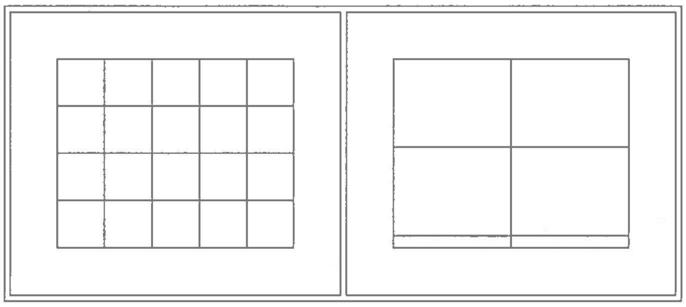


Figure B.1: *Patterns Used for User Calibration*

B.2 Scene Set-up

The next step is to set-up the conditions to take the images as controlled as possible. This means that the positions of the screen and the user must be measured and the user head must stay as still as possible and in a fronto-parallel position. The used configuration is shown in Figures B.2 and B.3.

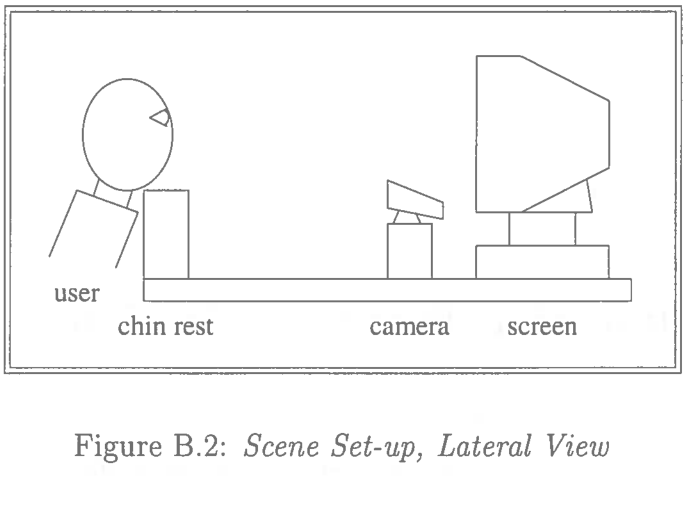


Figure B.2: *Scene Set-up, Lateral View*

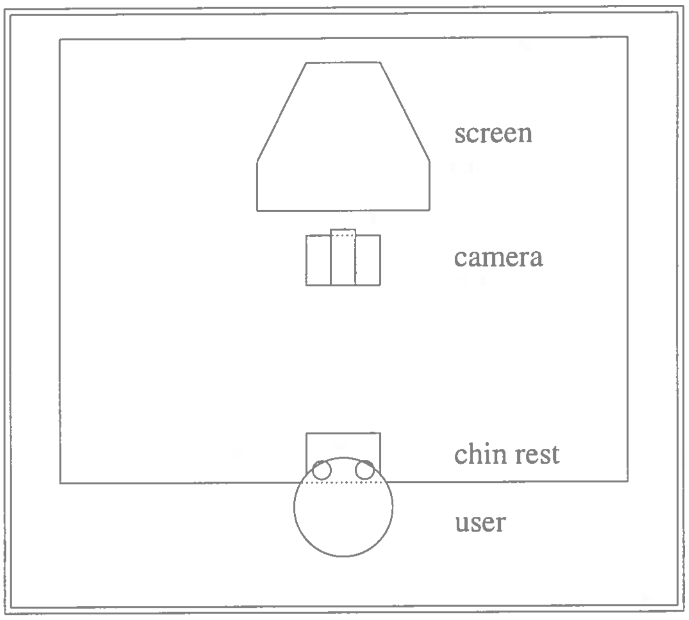


Figure B.3: *Scene Set-up, Top View*

B.3 Images Acquisition

Once the user is comfortably placed, images of he/she are taken, looking at the different points of the pattern. The usual care for image acquisition should be taken, as illumination, camera focus, etc.

B.4 Image Processing

The images acquired will pass through the same steps described in Chapters 3 and 4, namely:

- Eye images cropping
- Edge detection
- Iris location
- Corner detection

- Gaze vectors finding

To assure the accuracy of the process, the iris location process should be verified, to correct those images where the iris circle is misplaced. In the project this was done by hand. The eye images cropping and corner detection was done by hand as well. The gaze vectors finding involves the calculations explained in section 4.2.

B.5 Parameters Calculation

The last step is to relate the gaze vectors obtained in the previous step with the corresponding points in the scene, using the measurements taken in Section B.2, namely the position of the user's eye centres and the locations of the points with respect to the origin of the coordinate system, which is placed at the top-left corner of the screen.