Aus dem Institut für Neuro- und Bioinformatik
der Universität zu Lübeck
Direktor: Prof. Dr. rer. nat. Thomas Martinetz

Tracking Gaze and Human Activity

Inauguraldissertation
zur Erlangung der Doktorwürde der Universität zu Lübeck
aus der Technisch-Naturwissenschaftlichen Fakultät

vorgelegt von
Martin Böhme aus Darmstadt

Lübeck 2009

Erster Berichterstatter:	Prof. Dr.-Ing. Erhardt Barth
Zweiter Berichterstatter:	Prof. Dr.-Ing. Alfred Mertins
Dritter Berichterstatter:	Prof. Dr.-Ing. Andreas Kolb
Tag der mündlichen Prüfung:	21. April 2010

Zum Druck genehmigt
Lübeck, den 21. April 2010

gez. Prof. Dr. rer. nat. Jürgen Prestin
Dekan der Technisch-Naturwissenschaftlichen Fakultät

Jetzt bist du da, ein Stück deiner Zukunft dabei,
Es ist schon lange klar, du fühlst dich frei,
Wenn die Zukunft zur Gegenwart wird, hast du's getan.
Das Warten – war es wirklich dein Plan?

Die Fantastischen Vier

Acknowledgements

I have many people to thank for supporting me while I was working on this thesis.

First of all, I thank Erhardt Barth, my supervisor, from whom I learned a lot and who provided both technical guidance and moral support when it was needed. He likes to quip that sometimes, instead of doing pattern recognition, it is more fruitful to create recognizable patterns. I believe this advice holds true not just for pattern recognition but on a more general level: When you are toiling away at a problem, take a step back from time to time and ask yourself if you are actually working on the right problem. I thank Erhardt for pointing me towards the right problems.

I thank Thomas Martinetz for providing a great research environment – and, not least importantly, countless litres of coffee!

I thank Christopher Krause, Michael Dorr and Martin Haker, with whom I worked together on the research projects that resulted in the work described here. In a PhD thesis, one must by necessity describe one's own contribution – but that should not give the wrong impression that this was a solitary effort. Even my own contributions would not have turned out the same without the intensive and fruitful discussions with Christopher, Michael and Martin.

I thank everyone at the Institute for Neuro- and Bioinformatics for the friendly atmosphere that pervades the institute and for the lively discussions in the kitchen – some of the best of which did not revolve around either neuro- or bioinformatics.

It is said that the person who learns the most in the classroom is the teacher – and if that it so, I certainly have a lot of students to thank who endured my teaching over the years. I would add the corollary that the person who learns the most from a thesis is the supervisor – and so I particularly thank André Meyer, Mathis Graw, Martin Lechner and Kolja Riemer, whose theses I supervised.

During my time as a PhD student, I was funded by the German Ministry of Education and Research (BMBF) in the ModKog project, the German Research Council (DFG) in the Locomotor project, and the European Commission in the projects ARTTS and COGAIN. Despite all of the paperwork involved in a research project, one should not forget the valuable role that these institutions play in supporting our research. For all my gratitude, though, they will not let me get away without a disclaimer – so bear with me as I point out that this publication reflects the views only of the author, and that none of the funding institutions can be held responsible for any use which may be made of the information contained herein.

My deepest thanks go to the people who supported me on a personal level during my thesis work – but allow me to keep my thanks personal as well by extending them privately.

Contents

Introduction

Traditionally, people have interacted with machines by manipulating input devices using their hands, feet, and other parts of the body. We control computers with keyboards, mice, and joysticks; we fly aircraft with sticks, rudder pedals, and various levers and handles; we choose songs on our music players using buttons, scrollwheels, and touch screens.

For many tasks, this type of interaction via a physical intermediary is entirely appropriate. A physical interaction device provides *affordances* [99], i.e. indications of how we should interact with the device; it can provide physical feedback: a steering wheel turns more easily on an icy road; and indeed, using a physical object to interact with the environment is human nature: we are a tool-wielding species.

In contrast, much of our interaction with other people does not involve physical contact and relies instead on the senses of seeing and hearing. We communicate verbally, by speaking and listening, and non-verbally, through body language, gestures, facial expressions, and gaze. It seems plausible, then, that sight and sound should also be useful for interacting with machines.

This basic idea has been explored in many different ways. Computers can recognize speech with reasonable accuracy as well as synthesize speech, and these capabilities have been used for telephone dialogue systems, home automation, text input, and the like. Computer vision can be used to detect the presence or absence of humans, to measure body posture, to recognize gestures and facial expressions, or to track the direction of gaze, and these methods can in turn be used for sign language recognition, computer games, or alternative communications systems for people with disabilities.

This thesis will deal with the use of computer vision for human-machine interaction. Specifically, I will deal with eye tracking – measuring the direction of gaze – as well as applications of the time-of-flight camera, a combined range and image sensor, to human activity tracking.

Eye tracking has an obvious use in human-machine interaction as a pointing device: Because we can consciously control where we look and, moreover, because what we look at is usually what we are interested in, eye tracking has been used successfully to replace traditional pointing devices such as the mouse. This is of particular value to those who suffer from severe motor impairments, for whom the eyes are often among the few remaining parts of the body they can control. For these users, eye tracking can be a valuable communication tool. However, eye tracking has appeal as an interaction device for able-bodied users, too. For example, controlling games through gaze is not only fun but can actually be more effective than using traditional interaction devices [36].

On a more subtle level, eye tracking can be used not only for conscious interaction but to provide the machine with clues about what the user is interested in and where their attention lies. This information can then be used to modify the display depending on where the user is looking – for example, an interactive three-dimensional display can render the visual periphery in less detail and use the computing power that is saved to provide more detail in the region that the user is looking at directly. Moving one step further, we can modify the display to influence and thus optimize the user's gaze pattern – that is, to provide gaze guidance [8].

All of these applications require eye trackers that are accurate, easy to use, and, preferably, cheap. Only in the last few years have eye trackers become available that simultaneously satisfy at least the first two of these requirements. For a long time, eye trackers were only used in research because they were difficult to set up and burdensome to use: These eye trackers either required the user's head to be fixated in the device, or they were heavy devices worn on the user's head. Recent years have seen the development of remote eye trackers, which do not require physical contact with the user and allow the head to be moved within a certain range.

In this thesis, I will describe the development of such a remote eye tracker. On the hardware side, it consists of two infrared light sources, which provide controlled illumination, and a high-resolution camera, which records an image of the user's face. Algorithmically, there are two main problems to be solved: First, we must detect the user's eyes and measure the position of suitable image features on the eyes (the *image analysis* step); second, we must use these measured positions to infer the user's direction of gaze (the *gaze estimation* step).

Developing and testing algorithms for eye tracking is complicated by the fact that the ground truths for many intermediate quantities, such as the position of the eye in space or the various dimensions of the eye, are hard to determine accurately. For this reason, I will also describe a software framework that simulates the measurements made by the camera or cameras in a video-oculographic eye tracker. I will use this framework to validate the gaze estimation algorithms for the remote eye tracker.

Because some properties of the eye, such as the offset between the fovea and the visual axis, are hard or impossible to observe from the outside, existing video-oculographic eye trackers require *user-specific calibration*. During calibration, the user is asked to fixate one or several calibration targets; the measurements made during this calibration phase allow the system to adapt itself to the user's eyes. Because the calibration phase is burdensome for the user, it is desirable to eliminate it. I will describe an alternative approach that performs unobtrusive calibration in settings where the user is using a mouse for interaction. This approach makes use of the fact that when users click the mouse, there is a certain probability that they will be looking at the mouse cursor. This information can be used to obtain a calibration that is almost as good as a manual calibration, but without an explicit calibration phase.

The second part of this thesis deals with applications of the time-of-flight (TOF) camera to tracking humans and their activities. The time-of-flight camera is a novel type of sensor that delivers a range map and a corresponding intensity image with high temporal resolution (typically 30 frames per second or more). The camera performs a range measurement at each pixel by emitting modulated infrared light and measuring the time taken by the light to travel to the object and back to the camera. Given the speed at which light travels, the achievable accuracies (around 5 mm in optimal conditions) are impressive, but the range map that is obtained is still relatively noisy. I will describe a technique for improving the accuracy of the range map based on the observation that the range map and intensity image are not independent but are linked by the *shading constraint*: If the reflectance properties of the surface are known, a certain range map implies a corresponding intensity image. By enforcing this shading constraint, the intensity image can be used to improve the accuracy of the range map.

The range map delivered by the TOF camera can be used to compute features that describe geometric properties of the imaged object. I will show how a certain type of geometric features, the *generalized eccentricities*, can be used to distinguish different surface types. Furthermore, these features can be made scale-invariant by computing them on the irregularly sampled surface of the object instead of on the regularly sampled pixel grid. I will show how the geometric features, evaluated both on the range map and the intensity image, can be combined with a very simple classifier to yield a robust facial feature detector and tracker.

Finally, I will show how the popular face detection algorithm of Viola and Jones can be extended to use both intensity and range features. The resulting detector has a higher detection rate as well as requiring less execution time than a conventional detector trained only on the intensity images.

In academia, the actual software that implements an idea is often treated as an unloved child, relegated to a side note in the results section. Nevertheless, this thesis only exists as the result of a lot of code being written, and I am not about to disavow my offspring. Both the eye tracker and the interactive TOF applications are based on a high-performance image and video processing framework whose development I initiated and led. It consists of around 70,000 lines of C++ and assembly and is being used by three teams at the institute. I hope this child continues to mature and thrive.

Many of the results described in this thesis were obtained as part of a group effort. In these cases, I will, at the beginning of the corresponding chapter, identify which contributions are my own. For the sake of consistency, I will use the personal pronoun "we" throughout, even when describing results that are solely my own. I will deviate from this convention in only a few cases when expressing my own personal opinions.

Part I

Eye Tracking

1 Eye Tracking – An Overview

1.1 Introduction

Eye tracking is the process of measuring the orientation or motion of the eyeball. The original motivation was basic research: Besides being a tool for investigating the neurophysiological mechanisms that trigger and control eye movements, eye tracking is widely used to study attention and cognition, because gaze position is strongly correlated with visual attention and thus provides a good objective measure for it.

As eye trackers became easier to use, they started to be used for other purposes. Eye tracking can be used to diagnose diseases of the visual system [81]; in commercial applications, eye trackers are used to study the usability of web pages or the effectiveness of advertising media; eye trackers are starting to be fitted to cars and trucks to monitor driver alertness and attention [82]; and besides these passive monitoring applications, eye trackers are also used as input devices for human-computer interation (HCI), both for the general population [72] and, particularly, for disabled users [68] as a means for augmentative and alternative communication (AAC), environmental control, and wheelchair control (see [28, 29] for an extensive survey). Finally, eye trackers may be used to implement gaze-contingent displays, which change some aspect of the image as a function of where the user is looking (see e.g. [88, 48]), and gaze guidance, where the display is changed suitably to guide the user's eye movements and thus optimize the flow of information [8].

There is a host of different technologies and devices for eye tracking, which we will survey briefly in Section 1.3. The different technologies differ not only in terms of cost, accuracy, and the burden they place on the user (equipment mounted on the eye or head, restrictions on movement) but also in the type of data they deliver: Eye trackers may be binocular, meaning that they measure both eyes, or monocular, measuring only one eye. Also, different eye trackers measure different quantities. Some eye trackers measure rotation angles of the eye relative to the head; others measure the point of regard (POR) on a screen or some other planar display; still others deliver a line of sight (LOS) vector in three-dimensional world coordinates. Some authors refer to the measurement of eye rotation angles as *eye tracking* and the measurement of point of regard or line of sight as *gaze tracking*, but this convention is not universal. For clarity, we will therefore speak of *eye rotation angles*, *point of regard*, or *line of sight* if a particular type of measurement is meant; we will use *gaze direction* as a general term for all three

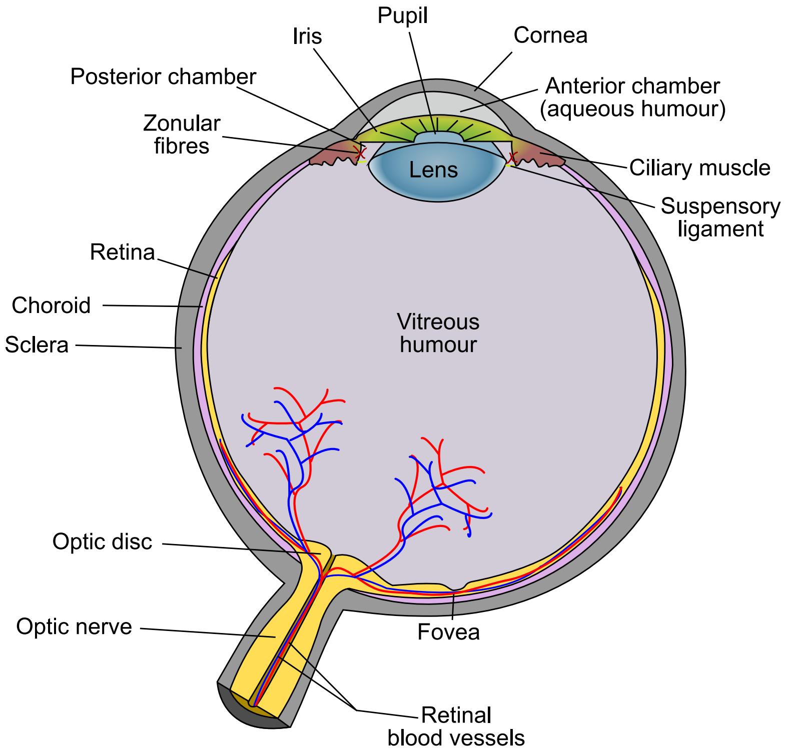


Figure 1.1: Anatomical structure of the human eye; see Section 1.2.1 for a detailed description. (Modified from a public domain image on Wikipedia.)

types of measurement; and we will use *eye tracking* to mean any process of measuring gaze direction.

A general property of many eye trackers is that they require user-specific calibration. This is because every person's eyes are slightly different, and thus the same measured quantity may correspond to different gaze directions in different people. Calibration is usually done by presenting one or several fixation targets at known positions and asking the user to fixate them; the measurements made for these known gaze directions are then used to calibrate the eye tracker.

1.2 The Human Eye

1.2.1 Anatomy

Figure 1.1 shows the anatomical structure of the human eye. (For a detailed treatment, see e.g. [4] and [123].) Most of the eye is covered by the *sclera*, a white fibrous tissue that serves mainly to protect the eye. At the front of the eye, the sclera joins to the *cornea*, a transparent tissue that protects the eye while allowing light to enter. As a curved, approximately spherical surface, the cornea plays an important role in the optics of the eye, providing the majority of the refractive power (around 42 dioptres). The second refracting component in the eye is the *lens*, a mass of transparent tissue surrounded by an elastic capsule. The lens is attached by the *zonular fibres* to the *ciliary muscle*; when this muscle expands and contracts, it changes the shape of the lens and thereby its refractive power. This process is known as *accommodation* and allows the eye to focus on objects at different distances. The refractive power of the lens can vary between about 19 dioptres for an object at infinity and 30 dioptres for an object 10 cm from the eye.

The *iris* is a ring of pigmented tissue located in front of the lens. It contains two antagonistic muscles that allow it to expand and contract, changing the size of its aperture, the *pupil*, to control the amount of light entering the eye.

The space between the iris and the cornea is called the *anterior chamber*. It is filled with the *aqueous humour*, a colourless liquid that maintains the shape of the cornea and provides nutrients to the cornea and lens. The *posterior chamber* – the space between the iris and the ciliary body – is also filled with aqueous humour. The *vitreous chamber* is the space behind the lens. It is filled with the *vitreous humour*, a gelatinous mass that stabilizes the shape of the eye.

The inner wall of the vitreous chamber is lined with the *retina*, a light-sensitive tissue, and the *choroid*, which lies between the retina and the sclera and contains vessels supplying blood to the retina.

The retina is composed of several layers of cells. The photoreceptor cells are contained in one of the outer layers (i.e. close to the choroid), and on top of this photoreceptor layer lie several layers of nerve cells that relay and modify the signals from the photoreceptors. An inner layer of nerve fibres converges towards the back of the eye to form the *optic nerve*, which transmits the sensory signals to the brain.

The photoreceptor cells in the retina can be divided into two types: *rods* and *cones*. Cones, in turn, exist in three subtypes, which respond to different wavelengths of light and thus allow different colours to be distinguished. Rods, on the other hand, cannot distinguish between wavelengths but are more sensitive in low-light conditions.

The density of rods and cones is not constant but varies across the retina; correspondingly, visual acuity is not constant either but is highest in the centre of the visual field and falls off towards the periphery. The centre of the visual field corresponds to the *fovea*, a circular region on the retina subtending about five degrees of visual angle, where cones predominate. The highest-resolution part of the fovea, known as the *foveola*, contains virtually no rods and subtends about one degree of visual angle. Because visual acuity varies across the retina in this way, the eyes are constantly moving to bring objects of interest onto the foveola.

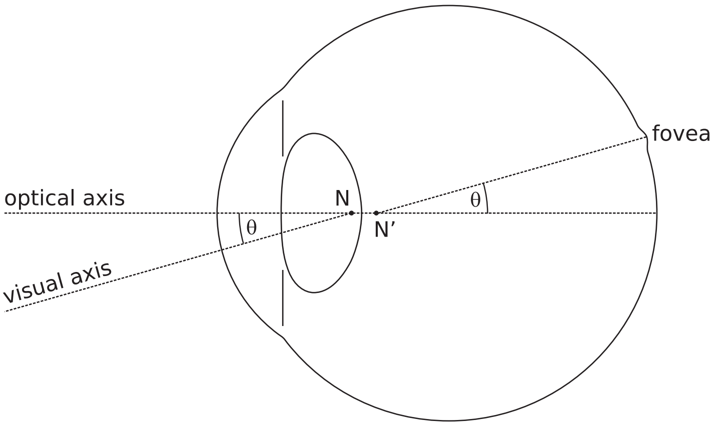


Figure 1.2: Optical characteristics of the human eye that are relevant to eye tracking. N: Front nodal point. N′: Rear nodal point.

The part of the retina where the optic nerve and retinal blood vessels enter the eye is called the *optic disc*. This part of the retina does not contain any photoreceptors and is therefore also known as the blind spot.

Six *extraocular muscles* are arranged around the eye; these muscles form three pairs of antagonistic muscles, and each pair rotates the eye around a different axis, allowing horizontal, vertical, and torsional eye movements.

1.2.2 Optics

Figure 1.2 illustrates the optical aspects of the human eye that are relevant to eye tracking. (For a detailed treatment of the optics of the human eye, see [4].)

The eye contains four refracting surfaces: The anterior and posterior surfaces of both the cornea and the lens. The ciliary muscle can change the shape of the lens to alter its refractive power and thereby the focal length of the entire system, allowing the eye to focus on objects at different distances.

It is useful to think of the refracting surfaces in the eye as being rotationally symmetric and as being aligned along a common axis of symmetry, the *optical axis*. In reality, this is only approximately true, and so to define the optical axis stringently, one needs to define it as the best-fit axis according to some error measure.

The *nodal points* of an optical system (marked as N and N′ in Figure 1.2) have the property that a ray travelling towards N at a certain angle to the optical axis will be refracted so that it appears to emanate from N′ at the same angle.

The nodal points are important for eye tracking because the fovea is displaced from the optical axis. Therefore, to find the point that the eye is fixating, we need to trace a ray from the fovea to the rear nodal point N′, find the angle between this ray and the optical axis, and then trace a second ray outward from the front nodal point N under the same angle. This second ray, which intersects the point that the eye is fixating, is known as the *visual axis*.

1.2.3 Eye Movements

There are two main reasons why the human eye needs to move within its orbit [81]:

The first is that when we walk or run, the head is jerked about. If the eye was rigidly connected to the head, these movements would cause the image on the retina to blur. To compensate for head movements, the human visual system contains two mechanisms: the *vestibulo-ocular reflex* (VOR), which generates compensating eye movements based on head accelerations sensed by the vestibular system in the inner ear; and the *optokinetic reflex*, which is based on the speed of image drift on the retina.

The second reason for eye movements is that visual acuity is highest at the fovea and falls off sharply towards the periphery. Because of this, the eye must constantly be moving, either to move the fovea onto different objects of interest or to keep the fovea on a moving target. Within this second group of eye movements, we will distinguish the following basic types:

Saccade A saccade is a quick eye movement that repositions the fovea onto a new target of interest. Saccades have peak velocities of several hundred degrees per second, depending on saccade amplitude [81, Chapter 3].

Smooth pursuit Smooth pursuit eye movements maintain the fovea on a moving target, or on a stationary target during self-motion. Smooth pursuit can reach velocities up to around 100 degrees per second [81, Chapter 4].

Fixation A fixation occurs when the eye is held stationary to examine a static target. Even during a fixation, however, the eye is still making small movements: Fixation may drift slightly over time, *microsaccades* (typically less than a third of a degree) may occur, and there is a constant tremor of less than 0.01 of a degree, with a frequency of up to 150 Hz. Since an image that is perfectly stabilized on the retina fades and disappears, it seems plausible that the purpose of tremor could be to counteract this effect, but this explanation has not been proved or disproved conclusively.

Vergence Depending on the distance of the object of interest, the eyes must have a certain orientation relative to each other to ensure that the object appears on both foveas at the same time. Vergence is a movement of both eyes in opposite directions which changes the relative orientation of the eyes, thereby adjusting them to different object distances.

Besides these basic categories of eye movements, there are several more that we will not discuss here; for details, see [81].

1.3 A Brief Survey of Eye Tracking Technology

There is a host of different devices and techniques for eye tracking; some of these are only of historical interest, but a whole range of techniques remains in active use. Which of these techniques is most suitable depends on the requirements of the application.

For an overview of both historical and contemporary techniques, see Wade and Tatler [134], Richardson and Spivey [112], Young and Sheena [142], and Schneider and Eggert [117].

Historically, the first attempts at eye tracking involved the experimenter directly observing the subject's eye [134, Section 1.3]. Even though humans can estimate another person's gaze direction to a certain degree [125], direct observation has obvious disadvantages in terms of accuracy and objectivity.

The first methods for recording eye movements [32, 67] used a plaster-of-Paris eye cup that was placed on the anaesthetized eye; a lever on the eye cup was connected to a pen, which recorded the eye movements on a rotating drum.

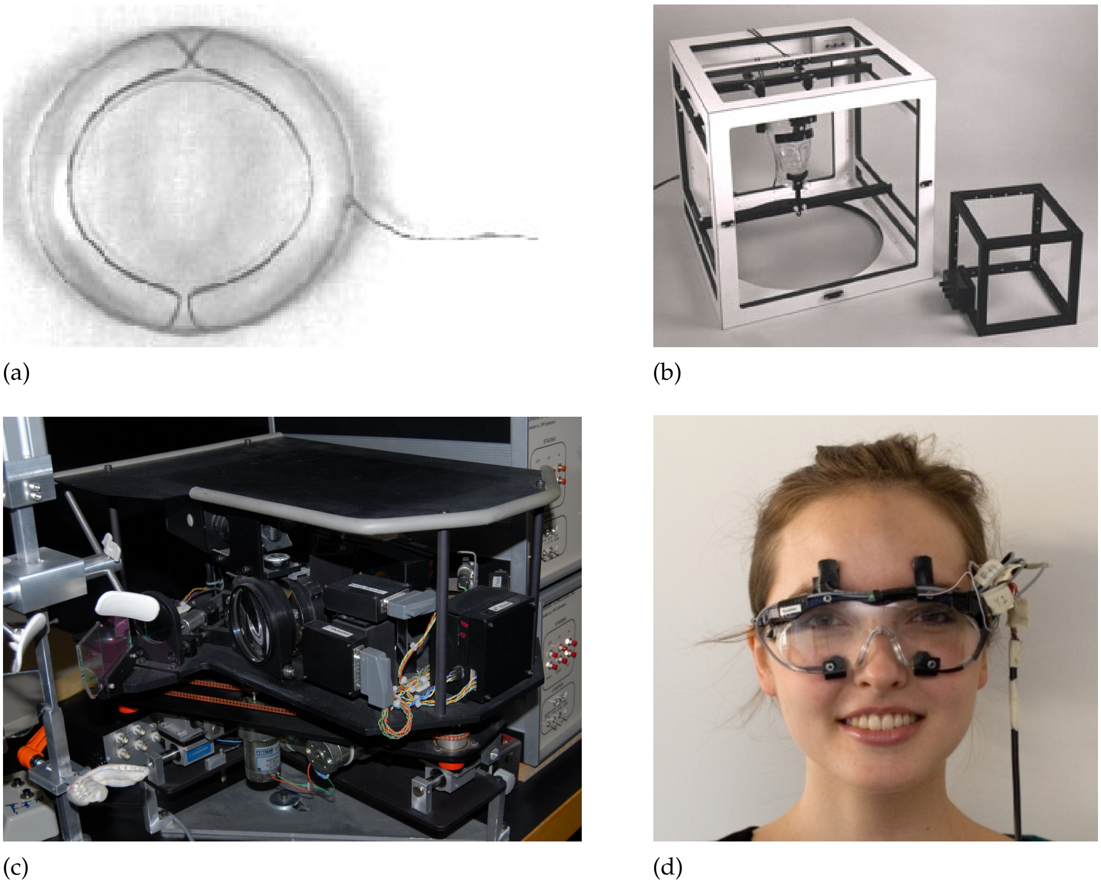
Dodge and Cline [35] developed a method that avoided direct contact with the eye; they reflected a vertical line of light on the cornea, passed it through a horizontal slit to generate a point of light, and recorded the movement of this point of light on a moving photographic plate.

Photographic techniques were widely used up to the 1960s, for example in Yarbus's seminal work on scanpaths [139]. However, they are quite laborious to use – for each recording, a photographic plate or film has to be developed – and if one wishes to record both horizontal and vertical eye movements, the temporal dynamics of the eye movements are lost and one obtains only the scanpath.

These disadvantages prompted the development of other eye tracking techniques, many of which are still in use today.

The *scleral search coil* technique [114] uses a contact lens introduced into the eye that contains one or two wire coils (see Figure 1.3a). A time-varying magnetic field is generated around the subject's head (see Figure 1.3b), and this induces a voltage in the search coils whose amplitude depends on the orientation of each coil to the magnetic field. Wires connect the search coils to a device that uses these signals to deduce the orientation of the eye.

Because the search coil technique offers excellent accuracy and temporal resolution, it is still considered the "gold standard" by which other techniques are judged. However, the contact lens may be uncomfortable for the subject to wear, and because of this, other techniques are preferred if the accuracy of the search coil is not required.



(a) (b) (c) (d)

Figure 1.3: Examples of different eye tracking techniques. (For examples of video-oculgraphic eye trackers, see Figure 1.4.) (a) Scleral search coil. (Image courtesy of Skalar Analytical, Breda, the Netherlands, reproduced with permission.) (b) Field frames for use with a scleral search coil. Coils in the field frames generate a time-varying magnetic field, which induces a current in the search coil. (Image courtesy of Skalar Analytical, Breda, the Netherlands, reproduced with permission.) (c) Dual Purkinje image (DPI) eye tracker by Fourward Technologies, Inc. (Image courtesy of Jan Drewes, reproduced with permission.) (d) Goggles for electro-oculography (EOG), from [23]. (Image courtesy of Bulling et al., reproduced with permission.)

A *dual Purkinje image* (DPI) eye tracker [30] (see Figure 1.3c) makes use of the fact that a light source produces a series of reflections, called the Purkinje images, at the front and rear surface of the cornea and the front and rear surface of the lens. The DPI tracker uses the first and fourth Purkinje image, which are imaged onto a photosensor. Servo motors drive mirrors to keep the two Purkinje images superimposed on the photosensor as the eye rotates; the position of these servo motors can then be used to deduce eye orientation. The DPI tracker provides good accuracy and temporal resolution but requires the head to be stabilized using a bite bar.

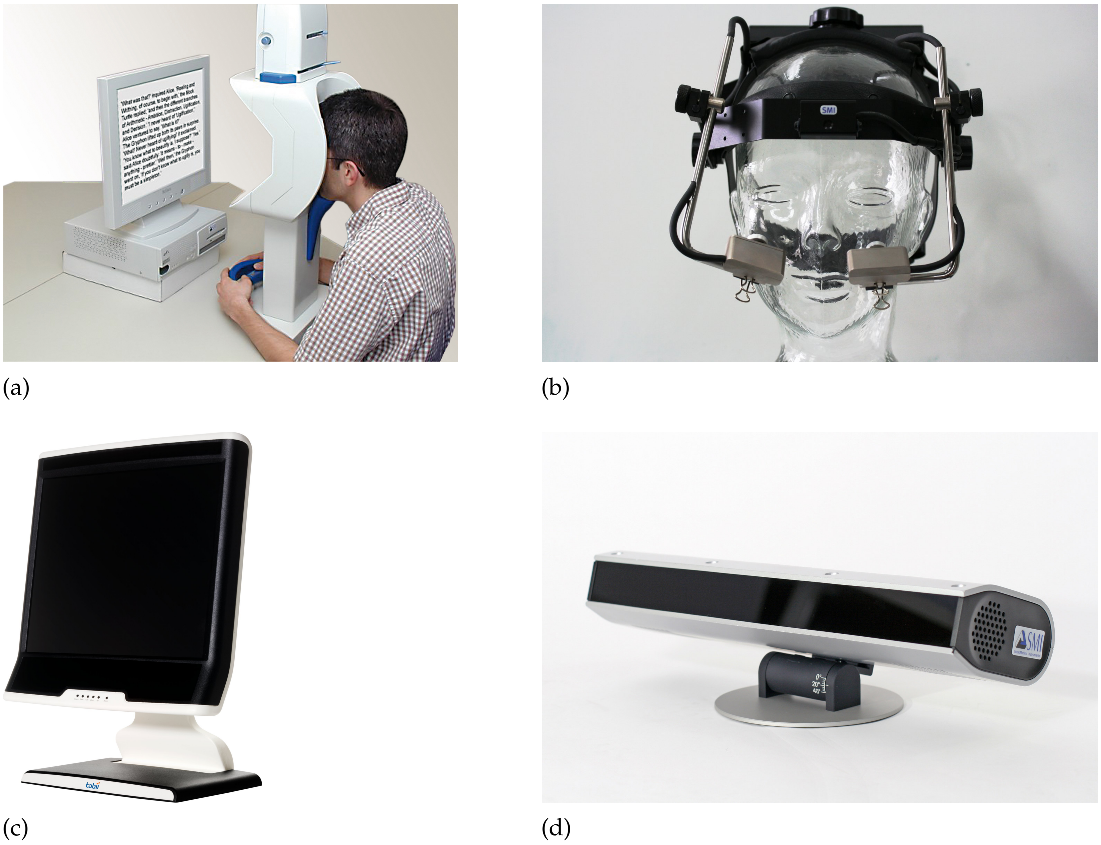
Electro-oculography (EOG) [93, 118] exploits the fact that the retina is an electrical dipole. Because of this, eye movements cause changes in the electrical field surrounding the eye, which can be picked up using electrodes placed around the eye (see Figure 1.3d). EOG is a relatively simple and inexpensive technique, and it is one of the few techniques that allow eye movements to be measured while the eyes are closed, but it is an invasive technique in that it requires the electrodes to be placed on the subject, and it is not as accurate as the scleral search coil or DPI trackers.

Infrared reflection devices (IRD) [141] use an infrared light source along with several photosensors located near the eye. Because the sclera reflects more light than the iris or pupil, the amount of light picked up by each photosensor changes as the eye rotates. This type of eye tracker has good temporal resolution and is inexpensive to build, but vertical eye movements can be hard to measure accurately because they do not cause as much change in the reflectance properties of the eye as horizontal eye movements.

Video-oculography (VOG) is a common term for a variety of techniques that use one or several video cameras to image the eye and measure its orientation, and this is the class of eye tracker that we will focus on in this work. Merchant et al. [89] describe an early VOG tracker; reviews are given for example by Duchowski [37] and Witzner and Ji [57]. An attractive feature of VOG eye trackers is that they can be quite non-intrusive: the camera can be mounted some distance away from the user, and the user may, within certain limits, move naturally, as long as the eye remains within the camera's field of view and provided the software can compensate for these movements. A disadvantage of VOG eye trackers is that their accuracy and temporal resolution is often substantially lower than that of other techniques. The next section will deal with the different types of VOG eye trackers and their characteristics in more detail.

1.4 Video-Oculographic Eye Trackers

A video-oculographic eye tracker uses one or several cameras to image the eye and computes the gaze direction from the information in the image. There is a broad range of devices that fit this general definition, and the category of video-oculographic eye trackers is probably more diverse than any other category of eye trackers. The choices


(a)

(b)

(c)

(d)

Figure 1.4: Examples of video-oculographic eye trackers. (a) SMI iView X Hi-Speed, a fixed-head eye tracker. (Image courtesy of the copyright holder, SensoMotoric Instruments GmbH, Teltow, Germany, reproduced with permission.) (b) SR Research EyeLink, a head-mounted eye tracker. (Image courtesy of Ingrid Brænne, reproduced with permission.) (c) Tobii T120, a remote eye tracker integrated into an LCD monitor. (Image courtesy of Tobii Technology, Danderyd, Sweden, reproduced with permission.) (d) SMI iView X RED, a remote eye tracker. (Image courtesy of the copyright holder, SensoMotoric Instruments GmbH, Teltow, Germany, reproduced with permission.) A version that is integrated into an LCD monitor is also available.

one is faced with when designing a video-oculographic eye tracker include the following:

Head movement tolerance and invasiveness Early video-oculographic eye trackers usually required the user's head to be fixated, often using a chin rest or bite bar. We will refer to these systems as *fixed-head* systems – see Figure 1.4a for an example. If the head does not move, the camera can be set up so that the eye fills most of the camera frame, which allows the position of relevant points in the image to be measured with good precision. Also, gaze direction is easier to compute if the head is fixed. However, keeping the head still is obviously uncomfortable and unnatural for the user. *Head-mounted* systems (Figure 1.4b) are worn on the user's head. This allows the user to move their head, but the weight of the system and the straps used to hold the system in place may be uncomfortable, particularly for prolonged use. *Remote* eye trackers (Figures 1.4c and 1.4d) are the least invasive variant: They do not require the user to wear any equipment and allow the user to move their head freely within certain limits. However, to keep the eye in the camera frame when the head moves, it is necessary either to increase the field of view of the camera (reducing precision) or to fit the camera with a pan-tilt mechanism (introducing additional mechanical complexity).

Number and type of cameras Fixed-head and head-mounted systems usually use a single camera if one eye is tracked (monocular tracking) or two cameras if both eyes are tracked (binocular tracking). The camera is either mounted close to the eye or is fitted with a tele lens so that the eye fills most of the camera frame.

For remote systems, the possibilities are more varied. Cameras may have a narrow or wide field of view (FOV) and may be steerable to track the user's face or eyes (using a pan-tilt unit or steerable mirrors). The simplest configuration uses a single fixed camera with a wide FOV [74, 140]. A single narrow-FOV camera may be used with a pan-tilt unit, but this approach appears to be unpopular because of the difficulty of reacquiring the eye if it leaves the camera's FOV. Multi-camera systems come in many different configurations. One option that multiple cameras open up is to use stereo vision [120], which greatly simplifies the problem of determining the depth coordinate of the user's head and eyes. Another popular option is to combine a fixed wide-FOV camera with a steerable narrow-FOV camera; in this setting, the wide-FOV camera is usually used to locate the user's eye, and the narrow-FOV camera is then steered to this position and used for the actual eye tracking [107]. Combinations of both of these ideas are of course also possible; one may for example combine a wide-FOV stereo system with a single steerable narrow-FOV camera [21, 101], or use two stereo units, one fixed and with a wide FOV, the other steerable and with a narrow FOV [11].

Illumination Many video-oculographic eye trackers use active illumination with infrared light. Not only does this make the lighting more predictable, but many systems also exploit the fact that the lights generate reflections on the corneal surface (so-called *corneal reflexes* or CRs) that can be used as convenient reference points. Some systems also exploit the so-called bright-pupil effect: Lights that are located close to the camera axis will cause the pupil to appear as a bright disc in the camera image, while off-axis lights yield a dark pupil. This effect permits easy segmentation of the pupil.

Eye tracking under ambient lighting has the advantage that it does not require any additional light sources, lowering the cost and complexity of the system. Also, this type of system works even in bright sunlight, which tends to drown out active infrared illumination. Conversely, systems that rely on ambient light obviously do not work in low-light conditions, and they must deal with the general problem of variable illumination. Finally, finding suitable reference points on the face is more difficult under ambient lighting because no well-defined corneal reflexes are generated.

Besides these aspects concerning the hardware side of the system, there is also a wide variety of approaches on the algorithmic side. Most eye tracking algorithms are made up of two major components: image analysis, which measures certain quantities in the image (often, the position of relevant features), and gaze estimation, which computes gaze direction from these measurements. These general tasks can be approached in different ways:

Image analysis The first task in image analysis is eye detection, i.e. finding the eye or eyes in the image. Fixed-head and head-mounted systems usually do not need to perform this step explicitly because the eye fills most of the image. Remote eye trackers with active infrared illumination will often search for the typical pattern produced by the corneal reflexes; these can be extracted quite well using a band-pass filter, and heuristics can be applied to eliminate false positives. The bright-pupil effect is another popular way of finding the eye. Eye detection is most difficult in remote eye trackers using ambient illumination; these need to employ more sophisticated techniques such as eigeneyes [106] or cascades of boosted classifiers [42, 132, 135].

Once the eye has been found, we need to measure certain quantities that can be used to determine gaze direction. Most eye trackers measure the position of key features in the image, such as the corneal reflexes, the eye corners, the pupil border, or the limbus. However, more holistic approaches also exist. An active appearance model (AAM) can be fit to the image of the eye; the parameters of the model are then used to compute gaze direction [56]. Another possibility is to train a neural network that maps images of the eye directly to gaze direction [5, 137].

Gaze estimation Gaze estimation algorithms may be divided into two main categories: Those that interpolate between data points obtained during calibration and those that model the actual physical characteristics of the eye. A simple example for the first category is the popular pupil-CR technique (e.g. [95, Section 4]), which maps the difference vector between the pupil center and a corneal reflex to the point of regard using an interpolation function (a biquadratic function, for example). The coefficients of the interpolation function are determined using least-squares estimation on the calibration points.

Model-based methods (e.g. [50]) typically trace light rays from the camera back to the eye and use prior knowledge about the eye and the eye tracker setup to determine the position and orientation of the eye. By incorporating more prior knowledge, model-based methods typically require less user calibration data than interpolation-based methods, particularly for remote eye tracking. The downside is that model-based methods usually take more time to implement and require a calibrated camera setup as well as knowledge about the positions of any active light sources.

2 A Simulation Framework for Eye Trackers

To develop eye tracking algorithms and assess their accuracy, it is necessary to have ground-truth data against which the results of the algorithms can be compared. To obtain these data, subjects can be asked to fixate certain known positions, and the gaze position computed by the eye tracker can be compared against the fixated position.

While this approach is the most realistic way of measuring the overall accuracy of an eye tracker under real-world conditions, it does have a number of shortcomings: (i) The position on the fovea that is brought onto the fixation target may change from fixation to fixation; moreover, the eye is not completely static during a fixation but may drift or perform microsaccades. This means that the true gaze direction is not known precisely. (ii) To find the source of gaze estimation errors, it is very useful to have ground truth values not only for the fixated location but also for intermediate quantities that may be computed by the algorithm, such as the position and orientation of the eyeball in space. It is very hard to measure these values accurately on a human subject. (iii) To examine the effect of hardware changes in the system, the measurements have to be redone. This is relatively time-consuming, which limits the amount of experimentation that can be carried out.

An alternative is to use a simulation of the system; this has the advantage that all ground truth values are known with high accuracy, and that the simulated hardware can be changed easily. The main disadvantage of simulation is that it must always make simplifications, and these can cause the algorithms to behave differently than they would in the real world. Nevertheless, simulation is a valuable tool for developing and testing eye tracking algorithms.

In this chapter, we describe a simulation framework for video-oculographic eye trackers [15] that uses an optical model to determine the coordinates of relevant features (pupil contour and corneal reflexes) in the camera image. These features can then be used by a gaze estimation algorithm to compute point of regard or line of sight. In Chapter 3, the framework will be used to test the gaze estimation algorithms described there.

The framework is written in MATLAB, an interpreted language for numerical computing that facilitates experimentation and allows gaze estimation algorithms to be expressed concisely. The source code for the framework is freely available on the web at `www.uni-luebeck.de/tools-demos/et_simul.zip`.

Some of the work described here has previously been published in [15].

2.1 Simulated Entities

Most video-oculographic eye trackers use the following processing steps: image acquisition, using one or several cameras; image analysis, to determine the position of relevant features in the image; gaze estimation, where the point of regard or line of sight is computed from the observed feature positions; and, optionally, a tracking component that tracks the change in position of the image features from frame to frame.

In the simulation framework, we have chosen not to simulate the image analysis step explicitly. That is, instead of computing the image seen by each camera (using 3D rendering algorithms) and then extracting the position of relevant features from the image, the framework directly computes the positions where these features will lie in the camera image given the spatial positions of the eye, camera, and lights. The effect of finite camera resolution and of inaccuracies in the image analysis algorithms are simulated by perturbing each image feature by a random offset.

Our main reason for leaving out the image analysis step is to increase the speed of the simulation: Rendering and analyzing a full camera image for every test condition would certainly not be practicable in a pure MATLAB implementation. Also, we believe that gaze estimation is where most of the problems lie that are particular to eye tracking; in contrast, the image analysis step usually uses proven existing techniques.

We will now describe in detail how the individual components of the system are modelled. We will also describe the simplifications we have made, i.e. which aspects of the system are not modelled.

2.1.1 Eye

The eye (see Figure 2.1) is the most important and most complex part of the model. It consists of the following components:

Cornea

This is modelled as a spherical cap with a radius of r_{cornea}, a centre of curvature $\boldsymbol{c}_{\text{cornea}}$ lying on the optical axis of the eye, and a cap height of h_{cornea}. (The numerical values for these parameters and others that follow are taken from the standard eye in Boff and Lincoln [13, Section 1.210].) For brevity, we will refer to the centre of corneal curvature simply as the *cornea centre*.

The corneal surface plays a role in two effects that are relevant for eye tracking:

Reflection The cornea acts as a spherical mirror in which reflections of the light sources – the CRS – are observed. Reflection at the surface of the cornea follows the law

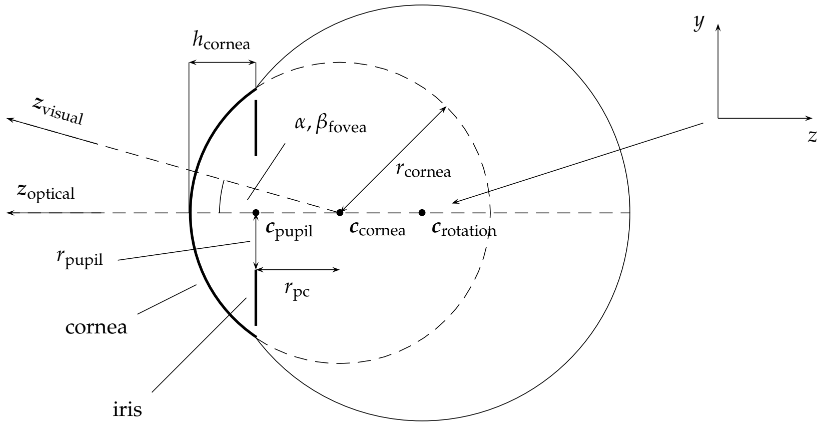


Figure 2.1: Eye model used in the simulation framework. The origin of the eye coordinate system is located at the eye's centre of rotation $\boldsymbol{c}_{\text{rotation}}$ but is shown outside the eye for clarity. The x-axis points into the page. The meaning of the various parameters is explained in Section 2.1.1.

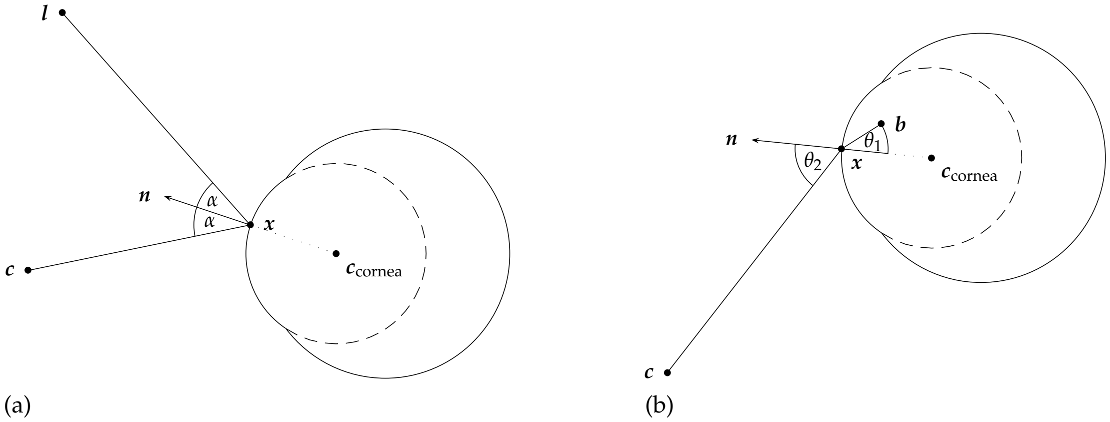


Figure 2.2: (a) Reflection of a ray from the light source $\boldsymbol{l}$ at a point $\boldsymbol{x}$ on the surface of the cornea towards the camera $\boldsymbol{c}$. $\boldsymbol{n}$: surface normal; $\boldsymbol{c}_{\text{cornea}}$: centre of corneal curvature; α: angle of incident and reflected ray with the surface normal. (b) Refraction of a ray from the pupil boundary point $\boldsymbol{b}$ at a point $\boldsymbol{x}$ on the surface of the cornea towards the camera $\boldsymbol{c}$. $\boldsymbol{n}$: surface normal; $\boldsymbol{c}_{\text{cornea}}$: centre of corneal curvature; θ_1, θ_2: angles of incident and refracted ray with the surface normal.

of reflection (angle of incidence equals angle of reflection [44], see Figure 2.2a):

$$\frac{\boldsymbol{c}-\boldsymbol{x}}{\|\boldsymbol{c}-\boldsymbol{x}\|_2}\cdot\boldsymbol{n}=\frac{\boldsymbol{l}-\boldsymbol{x}}{\|\boldsymbol{l}-\boldsymbol{x}\|_2}\cdot\boldsymbol{n}, \tag{2.1}$$

where $\boldsymbol{l}$ is the position of the light source, $\boldsymbol{c}$ is the position of the camera (from where the reflection is observed), $\boldsymbol{x}$ is the position on the corneal surface where the ray is reflected, and $\boldsymbol{n}=\frac{\boldsymbol{x}-\boldsymbol{c}_{\text{cornea}}}{\|\boldsymbol{x}-\boldsymbol{c}_{\text{cornea}}\|_2}$ is the surface normal at $\boldsymbol{x}$. In addition, $\boldsymbol{c}$, $\boldsymbol{l}$, $\boldsymbol{x}$, and $\boldsymbol{c}_{\text{cornea}}$ must be coplanar. Together with the constraint that $\boldsymbol{x}$ should lie on the surface of the cornea, i.e. $\|\boldsymbol{x}-\boldsymbol{c}_{\text{cornea}}\|_2 = r_{\text{cornea}}$, on the half-sphere facing $\boldsymbol{c}$, $\boldsymbol{x}$ is uniquely determined.

We find $\boldsymbol{x}$ by noting that it is constrained to the half-circle formed by intersecting the corneal half-sphere facing $\boldsymbol{c}$ with the plane given by $\boldsymbol{c}$, $\boldsymbol{l}$, and $\boldsymbol{c}_{\text{cornea}}$. We use a one-dimensional root-finder to find the solution for $\boldsymbol{x}$ that satisfies the reflection equation under these constraints.

After the point of reflection $\boldsymbol{x}$ has been found, we check to see if it actually lies within the boundaries of the cornea (i.e. within the spherical cap). If not, no CR is generated.

Refraction The observed image of the pupil is distorted by refraction at the corneal surface (see Figure 2.2b). This is governed by Snell's law [44]:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2, \tag{2.2}$$

where θ_1 is the angle between the incident ray and the surface normal, θ_2 is the angle between the refracted ray and the surface normal, and n_1 and n_2 are the indices of refraction of the two materials.

Given a point $\boldsymbol{b}$ on the pupil border, we wish to find the location $\boldsymbol{x}$ on the corneal surface where an incident ray from $\boldsymbol{b}$ is refracted in such a way that it passes into the camera at $\boldsymbol{c}$. Similar to the case of reflection, we note that $\boldsymbol{c}$, $\boldsymbol{b}$, $\boldsymbol{c}_{\text{cornea}}$ and $\boldsymbol{x}$ are coplanar and that $\boldsymbol{x}$ must lie on the half-sphere facing $\boldsymbol{c}$, giving a unique solution for $\boldsymbol{x}$. Again, we use a one-dimensional root finder to find $\boldsymbol{x}$; if the point of refraction that is found does not lie within the boundaries of the cornea, no image is generated for the pupil border point.

Pupil

The pupil boundary is modelled as a circle of radius r_{pupil} lying in a plane perpendicular to the optical axis with its centre at $\boldsymbol{c}_{\text{pupil}}$ on the optical axis. We will refer to the distance between $\boldsymbol{c}_{\text{pupil}}$ and $\boldsymbol{c}_{\text{cornea}}$ as r_{pc}.

Visual axis

Because the fovea is displaced temporally and slightly upwards from the optical axis, the visual axis, i.e. the line connecting the fixated point and the fovea via the nodal points, is displaced relative to the optical axis. We denote the horizontal and vertical angle of this displacement by α_{fovea} and β_{fovea}. In our eye model, we assume that there is only one nodal point and that it is coincident with $\boldsymbol{c}_{\text{cornea}}$; this is sufficiently accurate for our purposes.

Let z_{optical} and z_{visual} be the direction of the optical and visual axes. We can then define a matrix $\boldsymbol{F}$ that transforms the optical to the visual axis as follows:

$$z_{\text{visual}} = \boldsymbol{F}\, z_{\text{optical}}, \quad \boldsymbol{F} := \boldsymbol{Q}_{\boldsymbol{v}_\alpha}(\alpha_{\text{fovea}}) \cdot \boldsymbol{Q}_{\boldsymbol{v}_\beta}(\beta_{\text{fovea}}), \tag{2.3}$$

where $\boldsymbol{v}_\alpha$ and $\boldsymbol{v}_\beta$ are the axes around which the horizontal and vertical displacement angles are measured, and $\boldsymbol{Q}_{\boldsymbol{v}}(\theta)$ is a rotation matrix that rotates around the vector $\boldsymbol{v}$ by an angle of θ.

Rotation of the eyeball (Listing's law)

When the eye is rotated out of the *primary position* (which coincides approximately with the position where the eye is looking straight ahead), it undergoes a certain amount of torsion. The exact amount of torsion is governed by Listing's law, which states that the torsion of the eyeball will be that which is obtained by rotating around an axis that is perpendicular to both the visual axis in the primary position and to the visual axis in the new position [62, 133].

This is important because the eye is not rotationally symmetric around the optical axis; as stated above, the fovea is offset from the optical axis, and hence torsion will affect the relative orientation of the optical and visual axes.

Mathematically, Listing's law may be stated as follows:

$$z_{\text{visual},0} \cdot \boldsymbol{w} = z_{\text{visual},1} \cdot \boldsymbol{w} = 0, \tag{2.4}$$

where $z_{\text{visual},0}$ is the visual axis in the primary position, $z_{\text{visual},1}$ is the visual axis in the rotated position, and $\boldsymbol{w}$ is the direction of the axis around which the rotation takes place. (In the following, let $z_{\text{visual},0}$, $z_{\text{visual},1}$, and $\boldsymbol{w}$ be unit vectors.)

To find the rotation matrix $\boldsymbol{R}$ that transforms the eye from the primary position to the rotated position, we note that $\boldsymbol{R}$ transforms $z_{\text{visual},0}$ into $z_{\text{visual},1}$ and $\boldsymbol{w}$ into itself. Furthermore, $(\boldsymbol{w}, z_{\text{visual},0}, \boldsymbol{w} \times z_{\text{visual},0})$ form an orthonormal frame, and this frame is transformed by $\boldsymbol{R}$ into $(\boldsymbol{w}, z_{\text{visual},1}, \boldsymbol{w} \times z_{\text{visual},1})$. Thus, we can apply the transformation $\boldsymbol{R}$ to a point by finding the coordinates of this point in the frame $(\boldsymbol{w}, z_{\text{visual},0}, \boldsymbol{w} \times z_{\text{visual},0})$ and multiplying the basis vectors of the frame $(\boldsymbol{w}, z_{\text{visual},1}, \boldsymbol{w} \times z_{\text{visual},1})$ by these coordinates:

$$\boldsymbol{R} = \left(\boldsymbol{w},\ \boldsymbol{z}_{\text{visual},1},\ \boldsymbol{w} \times \boldsymbol{z}_{\text{visual},1}\right) \begin{pmatrix} \boldsymbol{w}^{\mathrm{T}} \\ \boldsymbol{z}_{\text{visual},0}^{\mathrm{T}} \\ (\boldsymbol{w} \times \boldsymbol{z}_{\text{visual},0})^{\mathrm{T}} \end{pmatrix}. \tag{2.5}$$

Limitations of the model

Because of the complex form and function of the eye, an eye model must almost necessarily make certain simplifications. The most important properties of the eye that are relevant for eye tracking but are not captured in the model are the following:

Limbus The limbus, i.e. the boundary between the cornea and the sclera, is currently not modelled as an image feature.

Cornea shape The true shape of the cornea is closer to an ellipsoid than to a sphere [4]; in spite of this, the current implementation uses a spherical model because it simplifies the optical calculations. However, one should keep in mind that this simplification may favour gaze estimation algorithms that likewise model the cornea as a spherical surface, thus underestimating their gaze error; conversely, the framework may actually overestimate the gaze error for algorithms that use a more sophisticated ellipsoidal cornea model (e.g. [11]).

Refraction at the posterior cornea surface The cornea and the aqueous humour have slightly different refractive indices (Boff and Lincoln [13] give values of 1.376 and 1.336, respectively). Because of this, refraction occurs at the boundary between these two media, additionally distorting the image of the pupil as viewed from the outside. The framework does not model this effect because it is comparatively small compared to the refraction at the air-cornea interface.

Occlusion by eyelids In real eye trackers, the eyelids may, in certain situations, hide parts of the pupil boundary or of the limbus, or they may obscure the CRs. This is an important effect, but it is hard to model realistically because there are a number of factors affecting eyelid position: the normal eyelid opening angle may vary between individuals and between races; bright lighting may lead to squinting; and vertical eye movements are accompanied by movements of the eyelids, particularly the upper lid (a phenomenon known as *lid-eye coordination* [124]). For this reason, the framework does not currently model eyelids.

Pupil In the real eye, the pupil centre will not, in general, lie exactly on the optical axis. Indeed, the pupil is not perfectly circular, so it is hard to define exactly what its centre is in the first place. Additionally, when the pupil contracts or expands, its shape may change, and its centre may shift [26].

2.1.2 Cameras

The framework models cameras using the pinhole camera model [44]. A camera has two parameters: The (horizontal and vertical) resolution in pixels, and the focal length, i.e. the distance between the pinhole and the image plane. By a convention that is common in computer vision (see e.g. [129]), focal length is measured in pixels.

Pan-and-tilt cameras are simulated; however, camera movement happens instantaneously, i.e. the framework does not simulate latency, inertia, maximum pan and tilt speeds, and so on.

Image acquisition and image analysis are simulated by projecting relevant feature points (points on the pupil contour and CRs) onto the image plane. If a feature point falls outside the boundaries of the simulated image sensor, it is marked as invalid. To simulate the combined effects of finite image resolution, finite signal-to-noise ratio, residual errors after camera calibration, and inaccuracies in the image analysis step, each point can be perturbed by a random vector. We call this random perturbation the *feature position error*.

Since many gaze estimation algorithms use the pupil centre as an image feature, this point is also determined by fitting an ellipse to the (perturbed) pupil boundary points in the image using the algorithm of Halíř and Flusser [55] and taking the centre of the ellipse as the pupil centre. (Note that because of perspective foreshortening, this point is not identical to the projection of the true pupil centre – which cannot, of course, be observed directly – onto the image plane.)

Limitations of the model

The simulated camera does not exhibit any lens distortion or other imperfections; an implementation of the simulated algorithms on real cameras will often require the internal parameters of the camera to be calibrated (see e.g. [44]). Furthermore, the simulated camera has infinite depth of field; in reality, depth of field is one of the limiting factors for head movement tolerance in remote eye trackers.

2.1.3 Lights

All lights are simulated as point light sources that radiate in all directions.

Limitations of the model

The simulation does not account for the fact that real light sources have spatial extent and that the apparent shape of the light source can change depending on the direction from which it is viewed. In real systems, this effect means that, when the light source is viewed from different directions, the centroid of the light source, as determined by the image analysis algorithms, can shift relative to the idealized point position of the light source.

2.2 A Simple Example

To demonstrate the use of the framework, we will show how to implement a simple eye tracker with one camera and one light source that uses the pupil-CR technique (see e.g. [95, Section 4]). This technique maps the difference vector between the pupil centre and a corneal reflex to the point of regard using an interpolation function. In this example, we will use a bilinear interpolation function

$$\begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} 1 \\ d_x \\ d_y \\ d_x\, d_y \end{pmatrix}, \tag{2.6}$$

where $\boldsymbol{d} = (d_x, d_y)$ is the difference vector between the position of the pupil centre and the corneal reflex in the camera image, and $\boldsymbol{g} = (g_x, g_y)$ is the gaze position on screen. The coefficients $a_{11}, \ldots, a_{24}$ are determined using least squares estimation on the calibration data.

In the simulation framework, an eye tracker is represented by a MATLAB structure that contains the camera(s), the light source(s), and the positions of the calibration points. The eye tracking algorithms are implemented as a pair of functions: (i) the *calibration function*, which is supplied with the positions of the image features observed for each calibration point and uses these to calibrate the eye tracker; and (ii) the *gaze estimation function*, which is supplied with observed image feature positions and uses these to compute the gaze position.

Figure 2.3 shows the calibration and evaluation function for the pupil-CR eye tracker. The calibration function `interpolate_calib` takes two arguments: `et` is the eye tracker structure, and `calib_data` is a structure containing calibration data, i.e. the positions of the image features observed for each calibration point. The calibration function uses this information to compute the matrix of coefficients for the interpolation function and stores these in the `state` field of the eye tracker object.

The gaze estimation function `interpolate_eval` also takes two arguments, the eye tracker structure `et` and a camera image structure `camimg`, which contains the positions

```
function et = interpolate_calib(et, calib_data)
    % Calculate pupil-CR vector for each calibration point
    for i=1:size(et.calib_points, 2)
        pcr = calib_data{i}.camimg{1}.pc- ...
              calib_data{i}.camimg{1}.cr{1};
        X(:,i) = [1 pcr(1) pcr(2) pcr(1)*pcr(2)]';
    end

    % Find least-squares solution for coefficients of
    % interpolation function
    et.state.A=et.calib_points/X;

function gaze = interpolate_eval(et, camimg)
    % Calculate pupil-CR vector
    pcr = camimg{1}.pc-camimg{1}.cr{1};

    % Evaluate interpolation function
    gaze=et.state.A*[1 pcr(1) pcr(2) pcr(1)*pcr(2)]';
```

Figure 2.3: Implementation of a simple eye tracker with bilinear interpolation.

of the observed image features. Using the pupil-CR vector and the matrix of coefficients stored in `et.state`, the function computes the gaze position and returns it in `gaze`. Note how the built-in matrix and vector features of MATLAB allow the algorithms to be expressed concisely.

3 Implementation of a Single-Camera Remote Eye Tracker

When implementing a remote eye tracker, a fundamental design choice is whether to use a single camera or multiple cameras. A multi-camera system can triangulate points in the scene to determine their position in space. Triangulation is not possible with a single camera, which makes gaze estimation more difficult, but single-camera systems have the advantage of being smaller and typically less expensive to manufacture.

In this chapter, we will describe a high-accuracy single-camera remote eye tracker that we have developed [14, 20, 92, 131]. At the time that we started this work, the most accurate remote eye tracking systems described in the literature used multiple cameras and achieved an accuracy of 0.5 to 1.0 degrees [11, 21, 101, 120, 140]. A commercial system [127] achieved similar accuracy using a single camera, but no implementation details had been published, and we were not aware of any comparable system in the literature. Since then, other academic groups have described similar single-camera approaches (e.g. [50, 65]), and several companies have introduced new single-camera eye trackers (e.g. [2, 122]).

Our eye tracker consists of a single camera mounted below a computer screen, with two infrared light sources to either side of the camera (see Section 3.1). The image analysis component is based on the Starburst algorithm [83], which was reimplemented and modified to fit the needs of the remote eye tracking setting (see Section 3.2). The gaze estimation algorithm uses a physical eye model to determine the position and orientation of the user's eyes (see Section 3.3). The complete system achieves an accuracy of around one degree and allows head movements of 20 cm between the extremes of the working range on all three spatial axes (see Section 3.6).

Parts of this chapter are joint work with others. André Meyer developed the image analysis algorithms (see Section 3.2) and implemented the C++ eye tracking software as part of his diploma thesis [91], which he conducted under my supervision. Some of the work described in this chapter has previously been published in [14, 20, 92, 131].

3.1 Hardware Setup

Figure 3.1 shows the hardware setup of the eye tracker; it consists of the following components:

Camera The camera (Lumenera Lu175 [86]) is placed centrally under the display. It has a monochrome ⅔-inch CMOS sensor with a resolution of 1280 × 1024 pixels and runs at a frame rate of 15 frames per second for full frames; the frame rate increases if only part of the pixel array is read out, but this feature was not used. The camera is connected to the computer by a USB 2.0 interface.

The lens (Pentax C1614-M [105]) has a fixed focal length of 16 mm, providing a field of view of around 60 degrees horizontally and 50 degrees vertically; at a distance of 50 cm from the camera, this corresponds to around 60 cm horizontally and 50 cm vertically. The eye tracker is designed to be used with the head at a distance of 60 cm from the screen, and the camera's focus is set to this distance; the field of view and depth of focus of the camera allow head movements in a volume of about 20 × 20 × 20 cm around this point.

To minimize the effect of ambient light, the lens was fitted with an infrared filter (Heliopan RG830 [64]), which blocks wavelengths below around 770 nm, reaching 50% transmission at around 830 nm and 90% transmission at around 870 nm. This filter response matches well against the infrared illuminators, which have a peak wavelength of 870 nm (see below).

The intrinsic parameters of the camera (focal lengths, principal point, and nonlinear lens distortion coefficients) were calibrated using the calibration routines from the OpenCV library [103], which are based on the camera calibration technique of Zhang [144].

Illumination Two infrared illuminators (epitex L870-66-60-550 [40]) are mounted below the display, one on either side of the camera. Each illuminator consists of 60 light-emitting diode (LED) chips and is fitted with a glass lens and a heat sink. The illuminator emits light with a peak wavelength of 870 nm and a half width of 40 nm. The maximum radiated power is 950 mW at a current of 800 mA and a voltage of 7.5 V. The illumination is pulsed; a pulse is triggered by the camera shutter and lasts for 7 ms.

Computer A PC with a 3 GHz Intel Pentium 4 CPU and 1 GB of RAM performs the image analysis and gaze estimation.

Display An 18-inch colour LCD display (ViewSonic VG800) with a display area of 36 × 28 cm is used to display calibration targets and test stimuli.

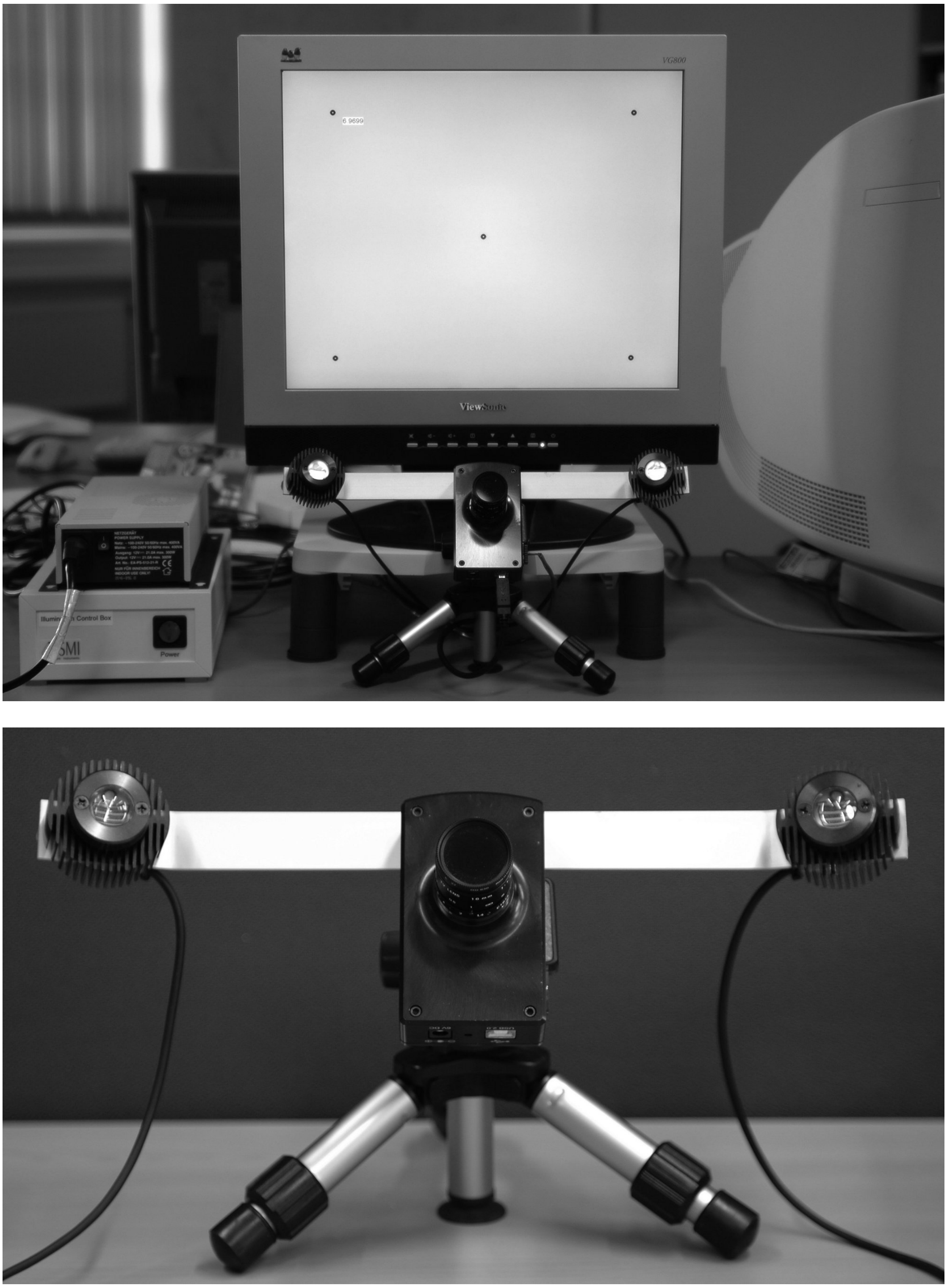


Figure 3.1: Hardware setup of the remote eye tracker. A camera and two infrared light sources are mounted below the display. The illumination control electronics are housed in the box to the left of the display. (Images courtesy of André Meyer.)

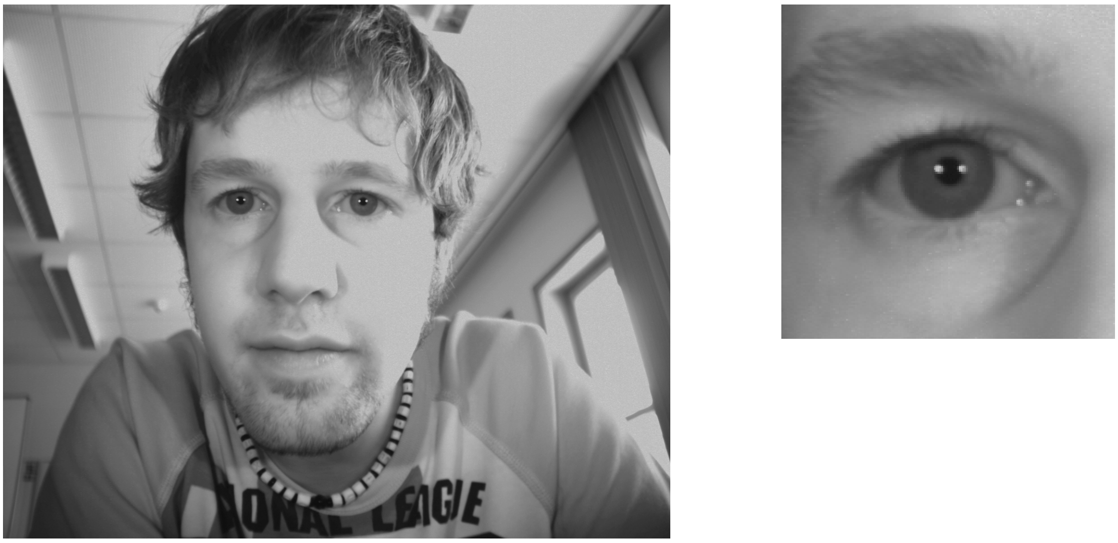


Figure 3.2: Left: Sample camera image from the remote eye tracker. Right: Close-up of the eye region. (Image courtesy of André Meyer.)

3.2 Image Analysis

The task of the image analysis step is to find the eyes in the image (if they are present) and to measure the position of the pupil centre and the two corneal reflexes in the image. Figure 3.2 shows a typical camera image and a close-up of the eye region.

Eyes are detected by searching for the typical pattern of the corneal reflexes – two small bright spots in close proximity. Once the eyes have been found, we find points on the pupil contour using a variant of the Starburst algorithm [83], which works by shooting rays from an initial pupil centre guess and looking for maxima in the derivative along this ray. Finally, we fit an ellipse to the pupil contour points. The rest of this section will describe each of these steps in more detail.

3.2.1 Eye Detection and Corneal Reflex Measurement

To find the corneal reflexes (CRs) in the image, we first apply a difference of Gaussians (DoG) filter [49, Chapter 4] to the image. This filter has a bandpass characteristic, and if its pass band is suitably matched to the size of the CRs, it leaves these intact while suppressing most of the other image content. We found that a DoG consisting of a 3×3 and a 5×5 binomial filter was a good match for the size of the CRs. Figure 3.4b shows the effect of applying this filter to an eye region.

Next, we threshold the filtered image to find candidate locations for CRs. This stage often produces a number of false detections; we eliminate these by requiring that CRs appear in pairs with a certain maximum distance. This simple rule is very effective; any possibly remaining false detections are identified later on in the pupil segmentation step if no valid pupil can be found near the CRs.

The thresholding of the DOG-filtered image yields an initial segmentation of the CRs. We refine this segmentation using a flood fill algorithm, which starts at the pixel with the highest intensity and uses an adaptive threshold based on this intensity. Finally, we compute the centroid of the segmented pixels to obtain the centre of the CR.

3.2.2 Pupil Segmentation

After the CRs have been found, we search for the darkest pixel in a rectangular region of interest (ROI) around the CRs; we expect this pixel to lie somewhere within the pupil. We create another ROI, centred around this pixel, and apply an adaptive threshold based on the intensity of the pixel to obtain an initial segmentation of the pupil. The segmented region may still contain other dark areas in the eye region, such as eyelashes, but the centroid of the region is already a good initial guess for the pupil centre.

We now determine adaptive thresholds for the intensities of the pupil and iris; these will be used to ensure that contour points that are found do indeed lie on the border between the pupil and iris.

Figure 3.3 shows the gray value histogram of a 50 × 50 pixel region around the preliminary pupil centre. The two large peaks correspond to pupil pixels and iris pixels, and we wish to find intensity ranges $[\text{pupil}_{\text{min}}, \text{pupil}_{\text{max}}]$ and $[\text{iris}_{\text{min}}, \text{iris}_{\text{max}}]$ that contain the pixels in the pupil and iris, respectively. We find these ranges as follows:

1 Search for the maximum histogram entry for intensities greater than 40. This is $\text{iris}_{\text{peak}}$. (The lower threshold of 40 is used because the pupil peak is sometimes higher than the iris peak.)

2 Starting from an intensity of zero, search for the first local maximum; this is $\text{pupil}_{\text{peak}}$. (Histogram entries containing fewer than five pixels are ignored to avoid spurious local maxima.)

3 Find the minimum histogram entry between $\text{pupil}_{\text{peak}}$ and $\text{iris}_{\text{peak}}$. This intensity is $\text{pupil}_{\text{max}}$; set $\text{iris}_{\text{min}} := \text{pupil}_{\text{max}} + 1$.

4 Set $\text{pupil}_{\text{min}} := 0$. Set $\text{iris}_{\text{max}} := \text{iris}_{\text{peak}} + (\text{iris}_{\text{peak}} - \text{iris}_{\text{min}})$.

Now that we have determined the intensity ranges for pupil and iris, the next step is to identify points on the pupil contour. We do this using a variant of the Starburst algorithm [83], which works by shooting rays radially from the initial pupil centre guess and searching for a maximum in the intensity derivative along each ray (see

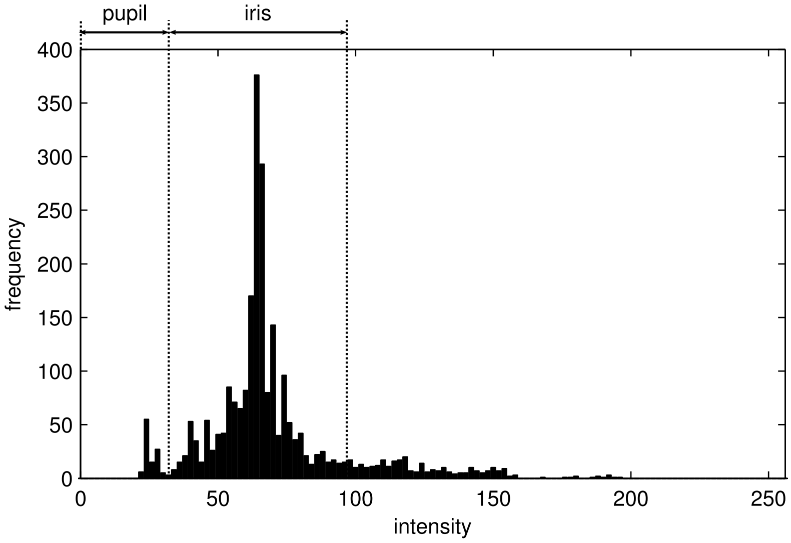


Figure 3.3: Intensity histogram of a 50 × 50 pixel region around the initial pupil centre. The intensity ranges corresponding to the pupil and iris are marked in the histogram.

Figure 3.4c). False positives are discarded by requiring that the contour point must lie between a pupil pixel (with a gray value in $[\text{pupil}_{\text{min}}, \text{pupil}_{\text{max}}]$) and an iris pixel (with an intensity in $[\text{iris}_{\text{min}}, \text{iris}_{\text{max}}]$).

From each contour point that was found, we now shoot a fan of secondary rays back towards the pupil and find contour points on these secondary rays (see Figure 3.4d). The rationale for this is that if the initial pupil centre guess was not good, some of the primary rays will cross the pupil border at an oblique angle, and no contour points may be detected on these rays. The secondary rays ensure that we detect contour points along the whole pupil border.

Finally, we fit an ellipse to the detected pupil contour points, using the algorithm of Halíř and Flusser [55]. Figure 3.4f shows the result of this ellipse fitting.

More details on the image analysis process are given in Chapter 4 of André Meyer's diploma thesis [91].

3.3 Gaze Estimation

The gaze estimation step takes the position of the pupil and the CRs for each eye and uses these to compute a line-of-sight (LOS) ray. The LOS rays for the two eyes are intersected with the screen plane, and we take the midpoint between the two intersection points to obtain the point of regard (POR) on screen.

The gaze estimation technique is based on a physical eye model that is very similar to the one used in the simulation framework (see Section 2.1.1). The only differences

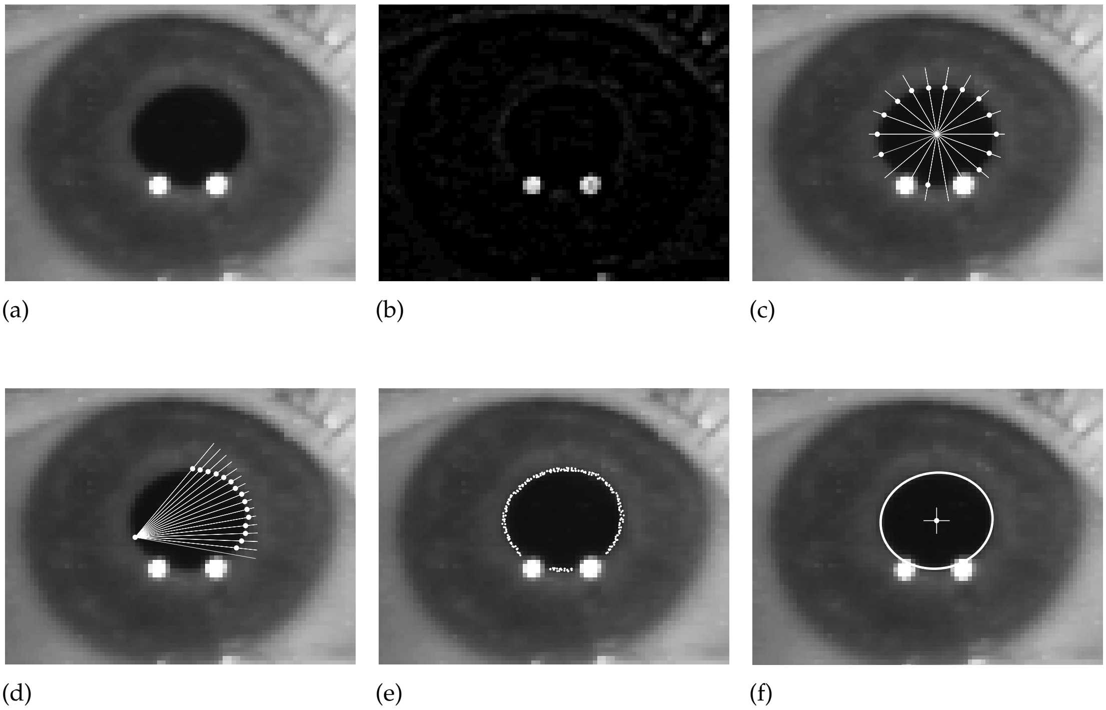


Figure 3.4: Overview of the image analysis process. (a) Eye region from the input image with the corneal reflexes (CRs). (b) Eye region filtered with a difference of Gaussians (DoG). The CRs are preserved while the rest of the image content is suppressed. (c) Primary rays are shot from the initial pupil centre guess, and contour points are detected on these rays. (d) A secondary ray is shot from each contour point, and additional contour points are detected on the secondary rays. (e) Contour points detected in steps c and d. (f) An ellipse is fitted to the contour points. (Images courtesy of André Meyer.)

are that for gaze estimation, the cornea is modelled not as a spherical cap but as a sphere, and that only the pupil center but not the pupil boundary is used.

In the simulation framework, the eye model was used to predict where the pupil and CRs will be observed in the camera image, given the position and orientation of the eye relative to the camera. For gaze estimation, the eye model will be used to solve the inverse problem of determining eye position and orientation from the position of the pupil and the CRs in the camera image.

The eye model contains four user-specific parameters: r_{cornea}, r_{pc}, α_{fovea}, and β_{fovea}. These parameters can either be set to population averages or estimated from data obtained during the calibration phase (see Section 3.4).

This approach of using a physical eye model, as well as the actual model used, is similar to that of several other authors [11, 50, 65, 74, 94, 102].

In particular, our approach – though developed independently – is very similar to that of Guestrin and Eizenman [50], who use the four user-specific parameters listed above as well as an additional fifth parameter, the effective index of refraction of the aqueous humour and cornea combined. As we do, they use a single camera and estimate the eye model parameters from calibration data. The main differences between their approach and ours is that we take Listing's law for the torsional component of eye rotation into account (see Sections 2.1.1 and 3.3.4) and that we use a statistical model to estimate the user-specific parameters (see Section 3.4).

Other existing work includes the following: Beymer and Flickner [11] model the reflective and refractives properties of an ellipsoidal cornea as well as the offset between the optical and visual axes; they successfully estimate the user-specific parameters of this model using a system composed of two wide-field-of-view cameras and two steerable narrow-field-of-view cameras.

Ohno et al. [102] describe a single-camera system that uses a single glint and determines the distance of the eye from the camera using depth-from-focus. They perform gaze estimation using an eye model; however, this model does not account for the offset between the optical and visual axes, and the eye model parameters are set to population averages. Kaminski et al. [74] describe a similar approach but use a face model to estimate the distance of the user from the camera.

The system of Hennessey et al. [65] consists of a single camera with two illuminators; they use an eye model to determine the position and orientation of the eye in space, but the offset between the optical and visual axis is not modelled, and the eye model parameters are set to population averages.

3.3.1 Algorithmic Prerequisites

In the algorithms that follow, we assume the existence of the following subroutines:

$\boldsymbol{x} = \texttt{unproject}(\boldsymbol{p}, d)$
Takes a point $\boldsymbol{p}$ on the camera's image plane and returns the point $\boldsymbol{x}$ at a distance of d from the camera's centre of projection (measured along the direction of projection) that has the image $\boldsymbol{p}$.

$(\boldsymbol{x}, \boldsymbol{d}) = \texttt{unproject\_ray}(\boldsymbol{p})$
Takes a point $\boldsymbol{p}$ on the camera's image plane and returns the ray (defined by the origin $\boldsymbol{x}$ and the direction $\boldsymbol{d}$) that passes through all points that have the image $\boldsymbol{p}$.

$\boldsymbol{x} = \texttt{intersect\_ray\_sphere}((\boldsymbol{o}, \boldsymbol{d}), \boldsymbol{c}, r)$
Takes a ray (defined by its origin $\boldsymbol{o}$ and direction $\boldsymbol{d}$) and a sphere (defined by its centre $\boldsymbol{c}$ and radius r) and finds the intersection $\boldsymbol{x}$ between the ray and the sphere that is closest to $\boldsymbol{o}$. It is assumed that some kind of error code is returned if the ray does not intersect the sphere.

$(\boldsymbol{u}_0, \boldsymbol{u}_\text{d}) = \texttt{refract\_ray\_sphere}((\boldsymbol{o}, \boldsymbol{d}), \boldsymbol{c}, r, n_\text{outside}, n_\text{sphere})$
Takes a ray (defined by its origin $\boldsymbol{o}$ and direction $\boldsymbol{d}$) and a sphere (defined by its centre $\boldsymbol{c}$ and radius r) and returns the point $\boldsymbol{u}_0$ where the ray strikes the sphere along with the direction $\boldsymbol{u}_\text{d}$ of the refracted ray. The refractive index outside the sphere is n_outside, the refractive index of the sphere is n_inside. It is assumed that some kind of error code is returned if the ray does not intersect the sphere.

$\boldsymbol{x} = \texttt{lines\_closest\_point}((\boldsymbol{u}_0, \boldsymbol{u}_\text{d}), (\boldsymbol{v}_0, \boldsymbol{v}_\text{d}))$
Takes two non-parallel lines (defined by points $\boldsymbol{u}_0$, $\boldsymbol{v}_0$ and directions $\boldsymbol{u}_d$, $\boldsymbol{v}_d$, respectively) and returns the point $\boldsymbol{x}$ that has the smallest sum of squared distances to the two lines.

$\boldsymbol{x} = \texttt{intersect\_ray\_plane}((\boldsymbol{u}_0, \boldsymbol{u}_\text{d}), (\boldsymbol{o}, \boldsymbol{x}_1, \boldsymbol{x}_2))$
Takes a ray and a plane, intersects the ray with the plane, and returns the point of intersection in terms of a two-dimensional coordinate system within the plane. The ray is defined by its origin $\boldsymbol{u}_0$ and direction $\boldsymbol{u}_d$; the coordinate system within the plane (and thereby the plane itself) is defined by its origin $\boldsymbol{o}$ and two perpendicular vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ that define its coordinate axes.

3.3.2 Estimating the Position of the Cornea

The first step in gaze estimation is to compute the position of the cornea centre $\boldsymbol{c}_\text{cornea}$ in space from the position of the two CRs in the image. For a given configuration of lights, any particular set of observed CR positions corresponds to a uniquely deter-

```
$\boldsymbol{c}_{\text{cornea}} = \texttt{estimate\_cc}((\boldsymbol{k}_i))$

Input:       $\boldsymbol{k}_i\ (i = 1, 2)$    Positions of corneal reflexes in image

Constants:   $r_{\text{cornea}}$              True radius of cornea
             $\boldsymbol{c}$                 Position of camera in space
             $\boldsymbol{l}_i\ (i = 1, 2)$    Positions of lights in space

Output:      $\boldsymbol{c}_{\text{cornea}}$ Position of corneal centre in space

Initialize estimated distance: $d_{\text{est}} = 0.5$ (metres)
repeat
    $\hat{\boldsymbol{k}}_i = \texttt{unproject}(\boldsymbol{k}_i, d_{\text{est}})\ \forall i = 1, 2$
    repeat
        $\boldsymbol{b}_i = \frac{\boldsymbol{c} - \hat{\boldsymbol{k}}_i}{\|\boldsymbol{c} - \hat{\boldsymbol{k}}_i\|_2} + \frac{\boldsymbol{l}_i - \hat{\boldsymbol{k}}_i}{\|\boldsymbol{l}_i - \hat{\boldsymbol{k}}_i\|_2}$
        $\boldsymbol{c}_{\text{cornea}} = \texttt{lines\_closest\_point}((\hat{\boldsymbol{k}}_1, \boldsymbol{b}_1), (\hat{\boldsymbol{k}}_2, \boldsymbol{b}_2))$
        $\hat{r}_{\text{cornea}} = 0.5 \left( \|\hat{\boldsymbol{k}}_1 - \boldsymbol{c}_{\text{cornea}}\|_2 + \|\hat{\boldsymbol{k}}_2 - \boldsymbol{c}_{\text{cornea}}\|_2 \right)$
        $\hat{\boldsymbol{k}}_i = \texttt{intersect\_ray\_sphere}(\texttt{unproject\_ray}(\boldsymbol{k}_i), \boldsymbol{c}_{\text{cornea}}, \hat{r}_{\text{cornea}})$
    until no $\hat{\boldsymbol{k}}_i$ moves by more than $\varepsilon_1$
    $d_{\text{est}} = d_{\text{est}} \sqrt{\frac{r_{\text{cornea}}}{\hat{r}_{\text{cornea}}}}$
until $|1 - \frac{r_{\text{cornea}}}{\hat{r}_{\text{cornea}}}| < \varepsilon_2$
```

Algorithm 3.1 (estimate_cc): Estimates the position of the cornea in space.

mined position of the cornea centre. This relationship can be expressed as a system of nonlinear equations, which can then be solved numerically (see e.g. [65]).

We have chosen a slightly different approach (see Algorithm 3.1) using an iterative procedure that adjusts the estimated distance of the eye from the camera (d_{est} in the algorithm) until the corneal radius that an eye at distance d_{est} would need to have to generate the observed CRs is equal to the known actual corneal radius.

The algorithm uses a double iteration: The outer loop converges on an estimate d_{est} for the distance of the eye from the camera and uses this distance in each step to find the position of the CRs in space. This will lead to both CRs ending up at the same distance d_{est} from the camera. In reality, however, the distance of the two CRs is usually slightly different. The inner loop thus refines the position estimates based on the observation that the following two constraints must simultaneously be fulfilled:

- The cornea centre must lie at the intersection of the surface normals at the two CRs. The surface normal at a CR can be obtained by taking the bisector (denoted by $\boldsymbol{b}_i$ in the algorithm) between the ray from the CR to the light and the ray from the CR to the camera.

- The cornea centre must lie at equal distances from the CRs.

The inner loop obtains positions for the cornea centre and the CRs that fulfill these two constraints. It does this by iterating the following procedure: Compute the surface normals at the CRs, intersect these to find the cornea centre, measure the distances from the CRs to the cornea centre and take their mean, then construct a sphere with this mean corneal radius around the cornea centre and intersect with the rays from the camera towards the two CRs to obtain new position estimates for the CRs.

Among other things, the inner loop thus determines the radius $\hat{r}_{\text{cornea}}$ of the corneal surface that would give rise to the observed CRs if placed at a distance of d_{est} from the camera. This distance estimate d_{est} is now adjusted depending on the relationship between $\hat{r}_{\text{cornea}}$ and the true corneal radius r_{cornea}. All other things being equal, it can be shown that corneal radius has to increase with the square of the distance to produce the same CRs (to a first-order approximation), which leads to the update rule used in the algorithm.

3.3.3 Finding the Optical Axis of the Eye

We will now determine the direction of the optical axis by finding the position of the pupil centre in space; the optical axis is then found as the line connecting the cornea centre and the pupil centre.

We find the pupil centre by tracing the light ray from the centre of the pupil in the image back into space (see Algorithm 3.2). We refract the ray at the surface of the cornea and trace it further into the eye. We now have two constraints on the position of the pupil centre: first, it must lie on the ray, and second, its distance to the cornea centre is r_{pc}. This means that we can find the pupil centre by intersecting the ray with a sphere of radius r_{pc} around the cornea centre; the first point of intersection is the pupil centre.

Note that we incur a systematic error in this step. To determine the position of the pupil centre in the image, we fit an ellipse to the pupil boundary, then take the centre of this ellipse as the pupil centre (see also Section 3.2.2). Under an orthographic camera projection, the centre of the ellipse would be identical to the projection of the pupil centre – which cannot, of course, be observed directly – onto the image plane; perspective foreshortening, however, leads to a slight difference between the ellipse centre and the true pupil centre, and this difference causes a systematic gaze estimation error. We will examine the magnitude of this error in Section 3.6.1; as we will see, the error is relatively small compared to other sources of error and can, moreover, be eliminated almost entirely during calibration.

```
$z_{\text{optical}} = \texttt{find\_optical\_axis}(\boldsymbol{c}_{\text{cornea}}, \boldsymbol{p}, r_{\text{cornea}}, r_{\text{pc}})$

Input:      $\boldsymbol{c}_{\text{cornea}}$   Position of cornea centre in space
            $\boldsymbol{p}$                   Position of pupil in image
            $r_{\text{cornea}}$                True radius of cornea
            $r_{\text{pc}}$                    Distance between $\boldsymbol{c}_{\text{cornea}}$ and $\boldsymbol{c}_{\text{pupil}}$
Constants:  $n_{\text{air}}$                   Refractive index of air
            $n_{\text{cornea}}$                Refractive index of cornea
Output:     $z_{\text{optical}}$               Direction of optical axis

$(\boldsymbol{p}_0, \boldsymbol{p}_d) = \texttt{unproject\_ray}(\boldsymbol{p})$
$(\boldsymbol{p}_0, \boldsymbol{p}_d) = \texttt{refract\_ray\_sphere}((\boldsymbol{p}_0, \boldsymbol{p}_d), \boldsymbol{c}_{\text{cornea}}, r_{\text{cornea}}, n_{\text{air}}, n_{\text{cornea}})$
$\boldsymbol{c}_{\text{pupil}} = \texttt{intersect\_ray\_sphere}((\boldsymbol{p}_0, \boldsymbol{p}_d), \boldsymbol{c}_{\text{cornea}}, r_{\text{pc}})$
$z_{\text{optical}} = \boldsymbol{c}_{\text{pupil}} - \boldsymbol{c}_{\text{cornea}}$
```

Algorithm 3.2 (`find_optical_axis`): Computes the direction of the optical axis.

There is an alternative approach to finding the pupil centre in space that avoids the pupil centre approximation error: Instead of tracing a light ray from the pupil centre in the image back into space, we can trace a ray from every pupil boundary point (see e.g. [65]). We then estimate the position and orientation of the pupil disc that best agrees with these rays. This eliminates the pupil centre approximation error but introduces a different problem: The existing approach to estimating the position of the pupil [65] requires the radius of the pupil to be known. This radius can be estimated from the size of the pupil in the camera image, but the fact that the image of the pupil is distorted by refraction at the cornea surface introduces an error into this estimate. In practice, we have found that this alternative technique for estimating the pupil centre is not any more precise than Algorithm 3.2.

3.3.4 Correcting Foveal Displacement

We will now correct for the displacement of the fovea from the optical axis to find the visual axis of the eye.

The matrix $\boldsymbol{F}$ (Equation 2.3, page 19) defines the transformation between the optical and visual axes, but this transformation is defined relative to the eye in its primary position. To apply $\boldsymbol{F}$, we therefore first need to know the rotation matrix $\boldsymbol{R}$ between the primary position and the current position of the eye. $\boldsymbol{R}$ is easy to compute if the direction of the visual axis is known (see Equation 2.5, page 20), but here we have only the optical axis.

To find $\boldsymbol{R}$, let $z_{\text{optical},0}$ and $z_{\text{visual},0}$ be the direction of the optical and visual axes of the eye in the primary position, and let $z_{\text{optical},1}$ be the direction of the optical axis in the eye's current position. (All of these vectors should be unit vectors.) We now wish to find the axis $\boldsymbol{w}$ around which the eye has rotated from the primary position to its current position. From Equation 2.4 (page 19), we know that $z_{\text{visual},0}$ is perpendicular to $\boldsymbol{w}$, i.e. $z_{\text{visual},0} \cdot \boldsymbol{w} = 0$. Furthermore, because $z_{\text{optical},0}$ rotates around $\boldsymbol{w}$ into $z_{\text{optical},1}$, we know that the projection of $z_{\text{optical},0}$ and $z_{\text{optical},1}$ onto $\boldsymbol{w}$ must be the same, i.e. $z_{\text{optical},0} \cdot \boldsymbol{w} = z_{\text{optical},1} \cdot \boldsymbol{w}$. From this, we find $(z_{\text{optical},1} - z_{\text{optical},0}) \cdot \boldsymbol{w} = 0$, i.e. the difference vector $z_{\text{optical},1} - z_{\text{optical},0}$ is perpendicular to $\boldsymbol{w}$. We now have two vectors that are perpendicular to $\boldsymbol{w}$, so we can find $\boldsymbol{w}$ using the cross product: $\boldsymbol{w} = z_{\text{visual},0} \times (z_{\text{optical},1} - z_{\text{optical},0})$.

We know that the rotation matrix $\boldsymbol{R}$ transforms $z_{\text{optical},0}$ to $z_{\text{optical},1}$ and $\boldsymbol{w}$ to itself. Again, as in the derivation of Equation 2.5 (page 20), we wish to find two orthonormal frames such that $\boldsymbol{R}$ transforms one to the other. To find these frames, we apply Gram-Schmidt orthonormalization to obtain the vectors

$$z_{\text{across},0} = \frac{z_{\text{optical},0} - (\boldsymbol{w}^{\mathrm{T}} z_{\text{optical},0})\boldsymbol{w}}{\|z_{\text{optical},0} - (\boldsymbol{w}^{\mathrm{T}} z_{\text{optical},0})\boldsymbol{w}\|_2}, \tag{3.1}$$

$$z_{\text{across},1} = \frac{z_{\text{optical},1} - (\boldsymbol{w}^{\mathrm{T}} z_{\text{optical},1})\boldsymbol{w}}{\|z_{\text{optical},1} - (\boldsymbol{w}^{\mathrm{T}} z_{\text{optical},1})\boldsymbol{w}\|_2}, \tag{3.2}$$

giving us the frames $(\boldsymbol{w}, z_{\text{across},0}, \boldsymbol{w} \times z_{\text{across},0})$ and $(\boldsymbol{w}, z_{\text{across},1}, \boldsymbol{w} \times z_{\text{across},1})$. From this, we find that $\boldsymbol{R}$ is

$$\boldsymbol{R} = \left(\boldsymbol{w},\ z_{\text{across},1},\ \boldsymbol{w} \times z_{\text{across},1}\right) \begin{pmatrix} \boldsymbol{w}^{\mathrm{T}} \\ z_{\text{across},0}^{\mathrm{T}} \\ (\boldsymbol{w} \times z_{\text{across},0})^{\mathrm{T}} \end{pmatrix}. \tag{3.3}$$

We can now finally use $\boldsymbol{R}$ to compute the orientation of the visual axis:

$$z_{\text{visual},1} = \boldsymbol{R}\,\boldsymbol{F}\,\boldsymbol{R}^{-1}\, z_{\text{optical},1}. \tag{3.4}$$

Algorithm 3.3 summarizes the steps for correcting foveal displacement.

3.3.5 Computing the Point of Regard

Now that the visual axis of the eye is known, we can intersect it with the display plane to find the point of regard on the display. Algorithm 3.4 ties together all of the steps of the gaze estimation process: estimating the position of the cornea; locating the pupil centre in space to find the optical axis; correcting for the foveal offset to find the visual axis; and finally intersecting the visual axis with the display plane to obtain the point of regard.

$\boldsymbol{z}_{\text{visual}} = \texttt{correct\_foveal\_displacement}(\boldsymbol{z}_{\text{optical}}, \alpha_{\text{fovea}}, \beta_{\text{fovea}})$

Input:	$\boldsymbol{z}_{\text{optical},1}$	Direction of optical axis
	α_{fovea}	Horizontal angle between visual and optical axis
	β_{fovea}	Vertical angle between visual and optical axis
Constants:	$\boldsymbol{z}_{\text{optical},0}$	Direction of optical axis in primary position
	$\boldsymbol{z}_{\text{visual},0}$	Direction of visual axis in primary position
Output:	$\boldsymbol{z}_{\text{visual},1}$	Direction of visual axis

$$\boldsymbol{w} = \boldsymbol{z}_{\text{visual},0} \times (\boldsymbol{z}_{\text{optical},1} - \boldsymbol{z}_{\text{optical},0})$$

$$\boldsymbol{z}_{\text{across},0} = \frac{\boldsymbol{z}_{\text{optical},0} - (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{z}_{\text{optical},0}) \boldsymbol{w}}{\|\boldsymbol{z}_{\text{optical},0} - (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{z}_{\text{optical},0}) \boldsymbol{w}\|_2}$$

$$\boldsymbol{z}_{\text{across},1} = \frac{\boldsymbol{z}_{\text{optical},1} - (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{z}_{\text{optical},1}) \boldsymbol{w}}{\|\boldsymbol{z}_{\text{optical},1} - (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{z}_{\text{optical},1}) \boldsymbol{w}\|_2}$$

$$\boldsymbol{R} = \Big(\boldsymbol{w},\ \boldsymbol{z}_{\text{across},1},\ \boldsymbol{w} \times \boldsymbol{z}_{\text{across},1}\Big) \begin{pmatrix} \boldsymbol{w}^{\mathrm{T}} \\ \boldsymbol{z}_{\text{across},0}^{\mathrm{T}} \\ (\boldsymbol{w} \times \boldsymbol{z}_{\text{across},0})^{\mathrm{T}} \end{pmatrix}$$

$$\boldsymbol{F} = \boldsymbol{Q}_{\boldsymbol{v}_\alpha}(\alpha_{\text{fovea}}) \cdot \boldsymbol{Q}_{\boldsymbol{v}_\beta}(\beta_{\text{fovea}}) \quad \text{(see Equation 2.3)}$$

$$\boldsymbol{z}_{\text{visual},1} = \boldsymbol{R}\,\boldsymbol{F}\,\boldsymbol{R}^{-1}\,\boldsymbol{z}_{\text{optical},1}$$

Algorithm 3.3 (`correct_foveal_displacement`): Computes direction of visual axis from direction of optical axis and foveal displacement.

3.4 User-Specific Calibration

We now turn to the problem of calibration, i.e. estimating the user-specific values for the eye model parameters from the data obtained during a calibration phase.

During calibration, the user is asked to fixate a number of calibration points whose position relative to the camera is known. For each of the calibration points, we record the position of the pupil and the CRs in the camera image. We do not require the user's head to remain fixed during calibration – but, for that matter, we do not require the user's head to move, either.

Our algorithm does not require a specific pattern of calibration points, but obviously the results tend to become more accurate the more calibration points are used.

The calibration algorithm uses a statistical approach: We find the maximum a posteriori estimate for the user-specific parameters. Assume that we have n calibration targets $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n$ that the user fixates in sequence, and that for each calibration target $\boldsymbol{t}_i$ we observe an image $\boldsymbol{I}_i$. (In particular, of course, we are interested in the position of the corneal reflexes and the pupil in the image, but we do not need to make this ex-

```
$\boldsymbol{g} = \texttt{estimate\_gaze}((\boldsymbol{k}_i), \boldsymbol{p}, r_{\text{cornea}}, r_{\text{pc}}, \alpha_{\text{fovea}}, \beta_{\text{fovea}})$

Input:      $\boldsymbol{k}_i\ (i = 1, 2)$    Positions of corneal reflexes in image
            $\boldsymbol{p}$                  Position of pupil in image
            $r_{\text{cornea}}$               True radius of cornea
            $r_{\text{pc}}$                   Distance between $\boldsymbol{c}_{\text{cornea}}$ and $\boldsymbol{c}_{\text{pupil}}$
            $\alpha_{\text{fovea}}$           Horizontal angle between visual and optical axis
            $\beta_{\text{fovea}}$            Vertical angle between visual and optical axis
Constants:  $n_{\text{air}}$                  Refractive index of air
            $n_{\text{cornea}}$               Refractive index of cornea
            $(\boldsymbol{o}, \boldsymbol{x}_1, \boldsymbol{x}_2)$   Coordinate system of the display plane
Output:     $\boldsymbol{g}$                  Point of regard on display

$\boldsymbol{c}_{\text{cornea}} = \texttt{estimate\_cc}(r_{\text{cornea}}, (\boldsymbol{k}_1, \boldsymbol{k}_2))$
$\boldsymbol{z}_{\text{optical}} = \texttt{find\_optical\_axis}(\boldsymbol{c}_{\text{cornea}}, \boldsymbol{p}, r_{\text{cornea}}, r_{\text{pc}})$
$\boldsymbol{z}_{\text{visual}} = \texttt{correct\_foveal\_displacement}(\boldsymbol{z}_{\text{optical}}, \alpha_{\text{fovea}}, \beta_{\text{fovea}})$
$\boldsymbol{g} = \texttt{intersect\_ray\_plane}(\boldsymbol{c}_{\text{cornea}}, \boldsymbol{z}_{\text{visual}}, (\boldsymbol{o}, \boldsymbol{x}_1, \boldsymbol{x}_2))$
```

Algorithm 3.4 (estimate_gaze): Computes point of regard on the display.

plicit here.) The image that is observed depends on the user-specific parameters, which we will gather into a parameter vector $\boldsymbol{\theta} = (r_{\text{cornea}}, r_{\text{pc}}, \alpha_{\text{fovea}}, \beta_{\text{fovea}})$. The posterior that we wish to maximize is then

$$\left(\prod_{i=1}^{n} P(\boldsymbol{I}_i | \boldsymbol{\theta}, \boldsymbol{t}_i) \right) P(\boldsymbol{\theta}). \tag{3.5}$$

To model $P(\boldsymbol{I} | \boldsymbol{\theta}, \boldsymbol{t})$, we need to introduce the function $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I})$, which computes the gaze position from an image $\boldsymbol{I}$ given the user-specific parameters $\boldsymbol{\theta}$. (In our eye tracker, $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I})$ performs the image analysis from Section 3.2 to obtain the position of the corneal reflexes and the pupil centre, then uses Algorithm 3.4 to compute the gaze position.) The computed gaze position $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I}_i)$ will not be exactly identical to the target $\boldsymbol{t}_i$, for several reasons: (i) The user does not fixate the target exactly; (ii) a certain measurement error is incurred when determining the position of the image features; and (iii) the physical model used in the gaze estimation algorithm is only an approximation of reality. We assume that these combined effects result in a normally distributed error between the computed gaze position $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I}_i)$ and the gaze target $\boldsymbol{t}_i$. This allows us to model the probability of observing an image $\boldsymbol{I}_i$ as follows:

$$P(\boldsymbol{I}_i | \boldsymbol{\theta}, \boldsymbol{t}_i) = \mathcal{N}(\boldsymbol{t}_i - \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I}_i) | \boldsymbol{\mu}_{\text{tracker}}, \boldsymbol{\Sigma}_{\text{tracker}}). \tag{3.6}$$

We assume that $\boldsymbol{\mu}_{\text{tracker}} = \mathbf{0}$ (i.e. there is no systematic gaze estimation error) and that the gaze estimation error is equally distributed in all directions with a standard deviation of σ_{tracker}, i.e. $\boldsymbol{\Sigma}_{\text{tracker}} = \text{diag}(\sigma^2_{\text{tracker}}, \sigma^2_{\text{tracker}})$. We estimate σ_{tracker} using simulations (see Section 3.6.1).

We turn now to the prior $P(\boldsymbol{\theta})$, which we also assume is normally distributed:

$$P(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}). \tag{3.7}$$

We take the parameters of the distribution from the literature. For the mean values $\boldsymbol{\mu}_{\boldsymbol{\theta}} = (\mu_{r_{\text{cornea}}}, \mu_{r_{\text{pc}}}, \mu_{\alpha_{\text{fovea}}}, \mu_{\beta_{\text{fovea}}})$, we take $\mu_{r_{\text{cornea}}} = 7.98$ mm, $\mu_{r_{\text{pc}}} = 4.44$ mm (from [13, Section 1.210]) and $\mu_{\alpha_{\text{fovea}}} = 5\,°$, $\mu_{\beta_{\text{fovea}}} = 2\,°$ (from [4, Chapter 4]).

The covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ is harder to obtain. We would expect some or all of the parameters to be statistically dependent – for example, r_{cornea} and r_{pc} should scale approximately equally with the overall size of the eye – but we have not been able to find any data on this in the literature. Hence, in the absence of better data, we assume $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ to be a diagonal matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \text{diag}(\sigma^2_{r_{\text{cornea}}}, \sigma^2_{r_{\text{pc}}}, \sigma^2_{\alpha_{\text{fovea}}}, \sigma^2_{\beta_{\text{fovea}}})$. Even the standard deviations in this matrix are somewhat hard to determine. From [41], we take $\sigma_{r_{\text{cornea}}} = 0.6$ mm. We were not able to find a corresponding value for $\sigma_{r_{\text{pc}}}$, but we speculate that most of the variation in r_{cornea} and r_{pc} is due to variations in the overall size of the eye, and so we assume that $\sigma_{r_{\text{pc}}}$ should have a similar relative magnitude as $\sigma_{r_{\text{cornea}}}$. We therefore choose $\sigma_{r_{\text{pc}}} = \frac{\mu_{r_{\text{pc}}}}{\mu_{r_{\text{cornea}}}} \sigma_{r_{\text{cornea}}} = 0.33$ mm. Good values for $\sigma_{\alpha_{\text{fovea}}}$ and $\sigma_{\beta_{\text{fovea}}}$ are even harder to obtain; Boff and Lincoln [13, Section 1.210] give a range of $5\,°$ to $7\,°$ for α_{fovea}, while Atchison and Smith [4, page 35] state that "the mean value of angle α is often taken to be about $+5\,°$ horizontally, but is usually in the range $+3$ to $+5\,°$, and is rarely negative. The visual axis is also downwards relative to the optical axis by $2 - 3\,°$." Based on this, we choose to set $\sigma_{\alpha_{\text{fovea}}} = 2\,°$ (allowing negative α_{fovea} in rare cases) and $\sigma_{\beta_{\text{fovea}}} = 1\,°$.

With the statistical model in place, we can maximize the posterior to find the parameter estimates. By computing the negative log posterior and discarding constant terms, we arrive at the potential function

$$\begin{aligned}\phi(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(\frac{\|\boldsymbol{t}_i - \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I}_i)\|_2}{\sigma_{\text{eyetrack}}} \right)^2 + \left(\frac{r_{\text{cornea}} - \mu_{r_{\text{cornea}}}}{\sigma_{r_{\text{cornea}}}} \right)^2 + \left(\frac{r_{\text{pc}} - \mu_{r_{\text{pc}}}}{\sigma_{r_{\text{pc}}}} \right)^2 + \\ \left(\frac{\alpha_{\text{fovea}} - \mu_{\alpha_{\text{fovea}}}}{\sigma_{\alpha_{\text{fovea}}}} \right)^2 + \left(\frac{\beta_{\text{fovea}} - \mu_{\beta_{\text{fovea}}}}{\sigma_{\beta_{\text{fovea}}}} \right)^2 . \end{aligned} \tag{3.8}$$

Because $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{I}_i)$ is a nonlinear function of $\boldsymbol{\theta}$, we find the minimum of the potential function using a nonlinear minimizer (the MATLAB function *fminunc*).

3.5 Recalibration

The results of the gaze estimation algorithm always contain a certain amount of systematic error because the algorithm's model of the system (camera, eye and display) is by necessity only an approximation. There are several factors that may contribute to this systematic error:

- The image analysis algorithms may make systematic errors when measuring the position of the pupil and the corneal reflexes in the image.
- The camera model contains residual errors after camera calibration (see Section 3.1).
- The eye model is only an approximation of the real eye; for example, the cornea is modelled as a spherical surface, whereas in reality it is closer in shape to an ellipsoid [4].
- The estimated eye model parameters contain residual errors.
- The measured position of the display relative to the camera contains errors.

To compensate for these and other sources of systematic error, we ask the user to fixate a number of calibration targets and determine the difference between this known gaze direction and the gaze direction estimated by the system; this information is stored and used to correct subsequent measurements.

Specifically, we construct a bilinear interpolation function that maps the point of regard $\boldsymbol{g}_{\text{est}} = (x_{\text{est}}, y_{\text{est}})$ as computed by the gaze estimation algorithm to the corrected point of regard $\boldsymbol{g}_{\text{corr}} = (x_{\text{corr}}, y_{\text{corr}})$ as follows:

$$\begin{pmatrix} x_{\text{corr}} \\ y_{\text{corr}} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{pmatrix} \begin{pmatrix} 1 \\ x_{\text{est}} \\ y_{\text{est}} \\ x_{\text{est}}\, y_{\text{est}} \end{pmatrix}, \tag{3.9}$$

where the parameters $a_{11}, \ldots, a_{24}$ are chosen using least squares estimation on the calibration data.

This procedure, which we call *recalibration*, is performed individually for each eye, because there may be differences in the systematic error between the two eyes. The results that are obtained for both eyes are then averaged to obtain the final point of regard.

System component	Position
Screen	$x = -0.18, \ldots, 0.18$ $y = 0.108, \ldots, 0.388$ $z = 0$
Lights	$(-0.15, 0.03, 0)$ $(0.15, 0.03, 0)$
Camera	$(0, 0, 0)$
Eye position volume	$(0, 0.388, 0.6) \pm (\frac{w}{2}, \frac{h}{2}, \frac{d}{2})$ $w = 0.27$, $h = 0.2$, $d = 0.2$

Table 3.1: Position of system components in the simulated eye tracker (coordinates in metres). The coordinate system is right-handed, with the screen in the x-y-plane and the z-axis pointing towards the user.

3.6 Results

3.6.1 Simulation

We will begin by testing the gaze estimation algorithms on the simulation framework described in Chapter 2. This will allow us to examine the effect of errors in the measured feature positions on gaze estimation accuracy, and will also be able to compare the eye parameters estimated by the gaze estimation algorithm to the ground-truth values.

Because the eye models used by the simulation framework and the gaze estimation algorithm are very similar, the simulated results are probably slightly optimistic, i.e. they underestimate the gaze error that would be made by a hardware implementation. With this caveat in mind, though, the simulation is a useful tool for assessing the properties of the gaze estimation algorithm.

For the tests in this section, we will not use a recalibration procedure (see Section 3.5) because we are interested in the performance of the gaze estimation algorithm itself, without any correction.

The virtual setup that we use for the tests is intended to be as close as possible to the physical layout of the real eye tracker described in Section 3.1. Table 3.1 gives the precise positions of the individual system components. All positions are given in a right-handed coordinate system, with the screen in the x-y-plane and the z-axis pointing towards the user.

We also define an *eye position volume*, i.e. an area within which we wish to be able to track a user's eye; this eye position volume is also intended to correspond as closely as possible to the eye position volume of the real eye tracker. The volume is box-shaped and is centred around a point 60 cm from the top centre of the screen. The dimensions (width, height, and depth) of the volume are 27 × 20 × 20 cm. Assuming an interocular distance of 7 cm, this eye position volume allows head movements of 20 × 20 × 20 cm if we wish to track both eyes. (However, all simulated results presented in the following are for one eye only.) The simulated camera is oriented with its optical axis pointing towards the centre of the eye position volume.

Visibility of CRs

As the eye rotates, the position of the CRs relative to the border of the cornea changes. Beyond a certain rotation angle, a CR will move off the cornea. When this happens, the reflex either moves onto the sclera or becomes invisible; in both cases, the reflex becomes unsuitable for eye tracking (a scleral reflex does not have the same properties as a corneal reflex because of the different curvature of the sclera).

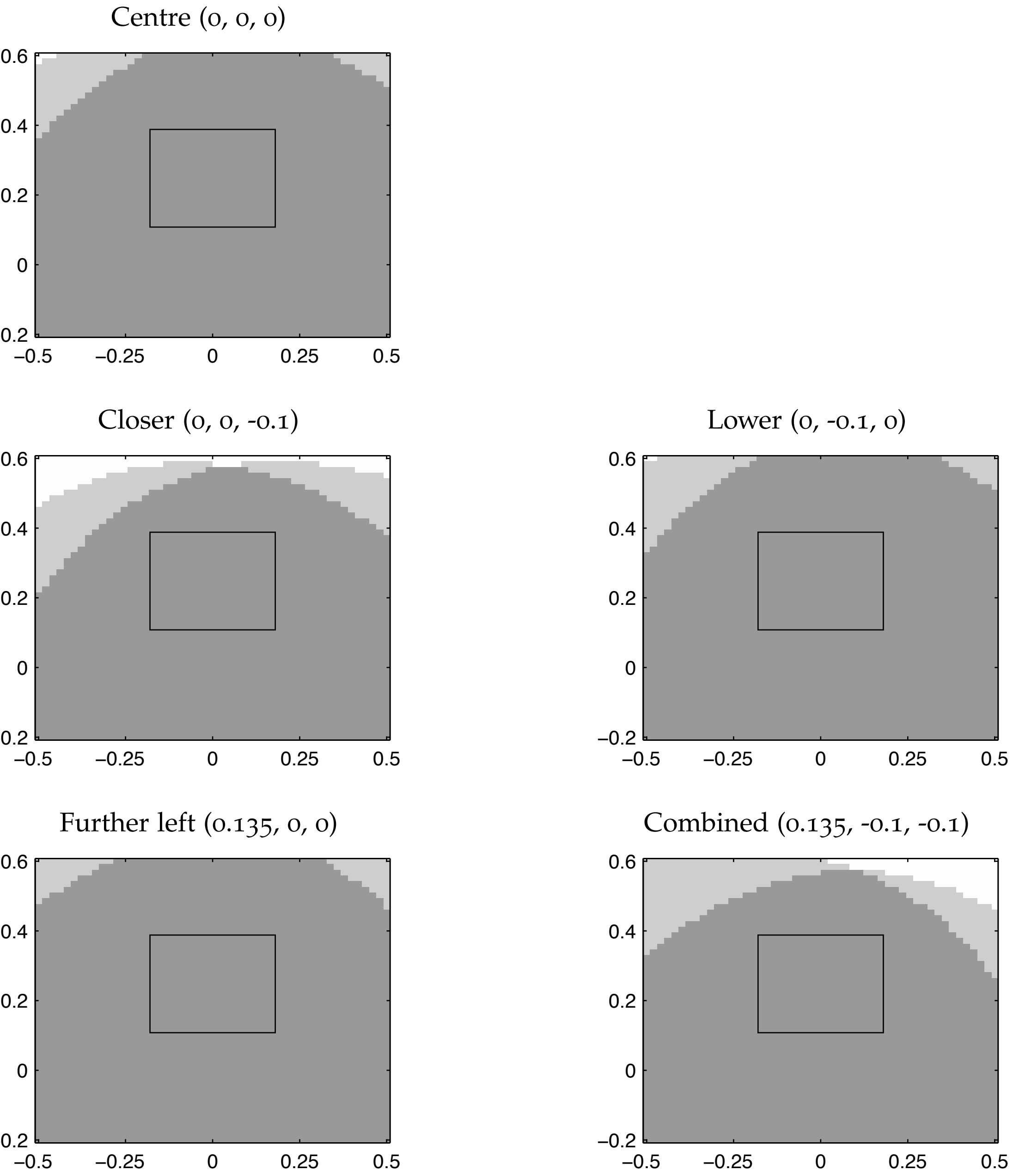
Because of this, we wish to validate that both CRs remain on the cornea for all gaze directions and eye locations the eye tracker is designed for. Consider Figure 3.5, which visualizes the visibility of the CRs as a function of gaze direction, for various eye locations. The x- and y-axis of each plot correspond to the coordinates of the gaze target. For each point in the plot, the shading indicates how many CRs are visible when the eye fixates that location (white: no glint visible; light grey: one glint visible; dark grey: two glints visible). The rectangular inset marks the boundaries of the screen; to ensure that gaze tracking is possible across the whole screen, two glints must be visible for all gaze positions within the screen.

Note first that when the eye is horizontally centred on the screen (topmost plot), the CR visibility regions are not exactly horizontally symmetric. This is caused by the asymmetry in the eye that is due to the offset between the optical and visual axis.

Comparing the plot for the centre of the eye position volume to the other plots, we notice that the greatest change occurs when the eye moves closer to the screen. The region where two CRs are visible shrinks, for two reasons: First, the CRs move further apart on the cornea, and hence the eye can rotate less before one of the CRs moves off the cornea; and second, when the eye is closer to the screen, it needs to rotate by a greater angle to fixate a given location in the screen plane.

When the eye moves parallel to the screen plane, the change in the CR visibility region is not as pronounced. The visibility region moves in the same direction as the eye, but by a smaller amount. This is because there are two effects at work that partially cancel each other out: As the eye moves down, for example, the CRs shift upwards on the cornea, but the eye also has to rotate further upwards to fixate a given location.

Finally, when the eye moves along all three axes to the front bottom left corner of the eye position volume, the CR visibility region still comfortably contains the screen.


Centre (0, 0, 0)

Closer (0, 0, -0.1)

Lower (0, -0.1, 0)

Further left (0.135, 0, 0)

Combined (0.135, -0.1, -0.1)

Figure 3.5: Visibility of CRs as a function of gaze direction; the x- and y-axis of each plot correspond to the coordinates of the gaze target. The plots show CR visibility for different eye positions, whose coordinates are given relative to the centre of the eye position volume (0, 0.388, 0.6). The shading indicates how many CRs are visible (white: no CR visible; light grey: one CR visible; dark grey: two CRs visible). The rectangular inset marks the boundaries of the screen.

Though we have only shown the results for a few selected eye positions, we have validated that both CRs remain visible for a finely spaced grid of positions spanning the eye position volume.

Gaze estimation using true eye parameters

We will now test the gaze estimation algorithm with the eye model parameters set to their ground truth values, i.e. with a "perfect" calibration. This serves two purposes: First, we want to explore how the algorithm performs under ideal conditions, and second, we want to determine σ_{tracker}, the standard deviation of the gaze estimation error, which is needed for the calibration algorithm (see Section 3.4).

We will investigate the effect that different amounts of error in the measured position of the image features have on gaze estimation error. The feature position error was uniformly distributed, and we varied the maximum magnitude of the error. We placed the simulated eye at the centre of the eye position volume and measured the resulting gaze estimation error for a grid of 16 × 16 gaze targets distributed across the simulated screen.

Figure 3.6 shows the results. First, consider the plot for a normal pupil radius of $r_{\text{pupil}} = 3$ mm. As expected, the gaze error increases with increasing feature position error. Interestingly, though, the gaze error does not reduce to zero even if no feature position error is present. The reason for this residual error is an approximation that the gaze estimation algorithm makes: It fits an ellipse to the pupil contour points in the image, then uses the centre of this ellipse as the position of the pupil centre. As explained in Section 3.3.3, this approximation introduces a certain error, and this error causes the gaze error we observe.

To eliminate the error caused by this pupil centre approximation, we can reduce the simulated pupil to a point by setting its radius to zero. The dotted plot in Figure 3.6 shows the results obtained for this setting; a feature position error of zero now results in a gaze error of zero, confirming that the pupil centre approximation was the source of the error. We also see that for feature position errors we might realistically expect in a real implementation (at least 0.2 pixels, say), the pupil centre approximation contributes only a few millimetres to the gaze error, while the greater part of the gaze error is caused by feature position error.

We will now use the data from this second plot to estimate the dependency of σ_{tracker}, the standard deviation of the gaze estimation error, on the feature position error e_{fpe}. A least-squares fit yields $\sigma_{\text{tracker}} = 34 \frac{\text{mm}}{\text{px}} \cdot e_{\text{fpe}}$, and this estimate will be used in the tests of the calibration algorithm.

We choose to estimate σ_{tracker} on the data for $r_{\text{pupil}} = 0$ because the calibration algorithm (see Section 3.4) assumes that there is no systematic gaze estimation error; in other words, we are interested only in the noise-dependent component of the gaze

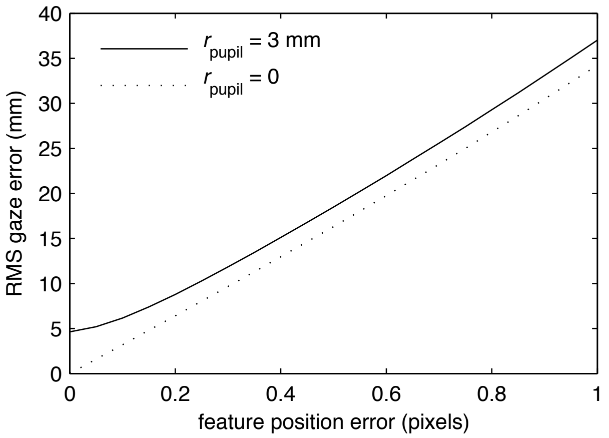


Figure 3.6: Effect of feature position error on gaze estimation error. The eye model parameters were set to ground-truth values.

estimation error. As we will see later, the calibration algorithm can compensate very well for the systematic error by calibrating the eye model parameters to values that are slightly biased relative to their ground truth values.

Effect of inaccuracies in eye model parameters

Before we use the calibration algorithm to estimate the user-specific eye model parameters, we will investigate the effect that errors in these parameters have on gaze estimation. We will keep the parameters used by the gaze estimation algorithm fixed at the population averages from Section 3.4 and change the parameters of the simulated eye to investigate their effect on the gaze error.

Changing α_{fovea} and β_{fovea}, i.e. the angle between the visual and optical axis, has a rather predictable effect: It adds a fixed angular error to the gaze estimate. For this reason, we will only examine the effect of changing r_{cornea} and r_{pc}.

The standard deviation of r_{cornea} is 0.6 mm (see Section 3.4), which corresponds to around 8% of the mean value. Assuming a normal distribution, this means that around 99.7% of the adult population have a corneal curvature that lies within 24% of the mean. Assuming that r_{pc} has a similar distribution, we varied both parameters of the simulated eye from 25% below the population average to 25% above. Figure 3.7 shows the effect of this on the gaze estimation error. (The feature position error was set to zero.)

The top plot shows the effect of varying both parameters in unison, which we believe is reasonable because most of the variation should be due to differences in the size of the eyeball as a whole. At 25% or about three standard deviations below the assumed

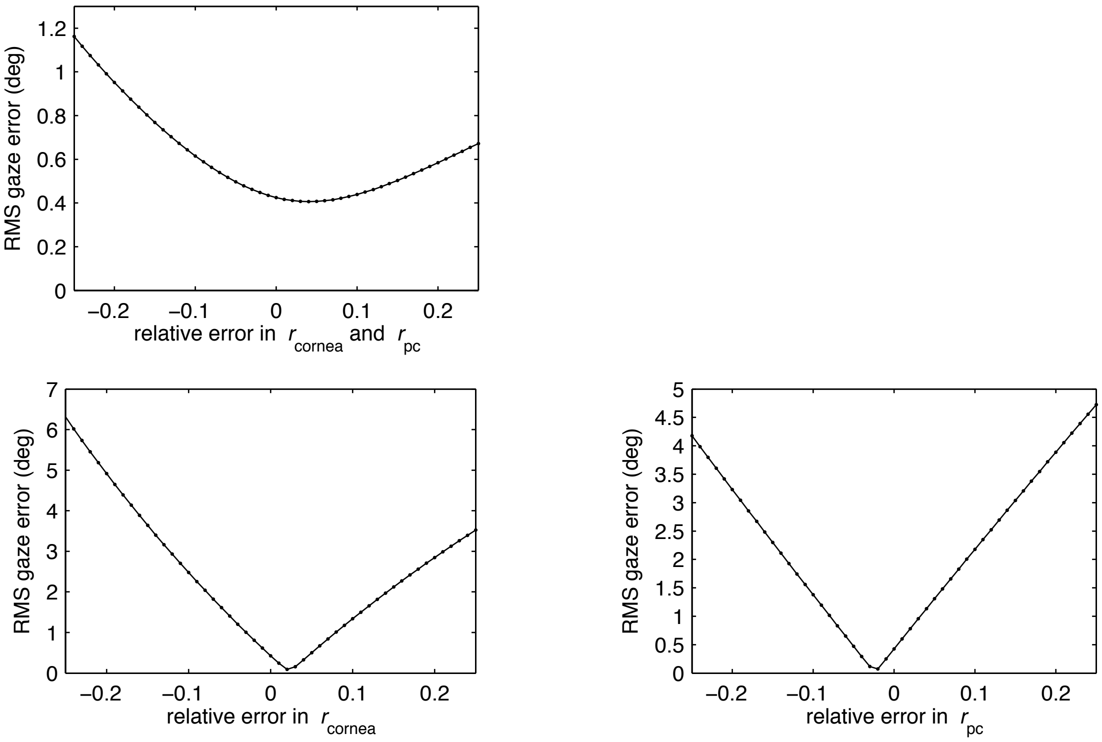


Figure 3.7: Effect of inaccuracies in the eye model parameters. Top: The parameters r_{cornea} and r_{pc} of the simulated eye are varied by the same ratio relative to the values assumed by the gaze estimation algorithm. Bottom: Only r_{cornea} or r_{pc} is varied while the other parameter is fixed at the assumed value.

values for r_{cornea} and r_{pc}, the parameter error accounted for another 0.7 degrees of gaze estimation error in addition to the 0.4 degrees obtained for a relative error of zero (this residual error is caused by the pupil centre approximation).

Because the relative deviation of both parameters will not be exactly the same, we also investigated the effect of varying just one parameter (see the lower row of plots in Figure 3.7).

Here, the effect on gaze estimation error is much stronger, reaching over 6 degrees for an r_{cornea} value 25% below the assumed value. Though it is probably not realistic for r_{cornea} or r_{pc} to change this much without the other parameter changing at all, smaller changes of ±10% still increase the gaze error by over two degrees. This shows that calibrating the eye model parameters accurately can have an appreciable effect on the gaze error.

The plots shows another, perhaps even more interesting effect: The lowest gaze error is not achieved when the actual values of the parameters are identical to the assumed values but when the actual value of r_{cornea} is slightly too high or when the actual value of r_{pc} is slightly too low. At the lowest point in the two plots, the gaze error is less than 0.1 degrees, compared to 0.4 degrees for a relative error of zero. This suggests that a conscious miscalibration of r_{cornea} or r_{pc} can actually compensate for most of the error caused by the pupil centre approximation. The calibration algorithm will take advantage of this fortuitous effect automatically because it seeks to minimize the gaze estimation error. As we will see later, the parameter values determined by the calibration algorithm actually yield smaller errors than the ground truth parameter values. One may argue that this deliberate miscalibration is not entirely sound from a theoretical point of view; however, the actual amount by which the parameters are miscalibrated is fairly small, and the miscalibration does reduce the gaze estimation error, so we feel that we can allow it to occur.

Accuracy of eye model parameter estimation

We will now investigate how accurately the calibration algorithm from Section 3.4 can estimate the user-specific eye model parameters.

We performed the calibration with a rectangular 3×3 grid of calibration targets, with the eye at the centre of the eye position volume. We set each of the parameters of the simulated eye to one standard deviation above the population mean used by the prior in the calibration algorithm (see Section 3.4); this was to avoid the unfair advantage that the calibration algorithm would have had if we had set the eye parameters exactly to the means used in the prior. We varied the amount of feature position error to assess its effect on the accuracy of the estimated parameters. Because the random variations of the feature position error can cause corresponding variations in the parameter estimates, we performed 100 calibrations for each given amount of feature position error.

Figure 3.8 shows the results. The dotted line plots the ground-truth parameter value, the dashed line plots the mean parameter estimate over all runs, and the shaded region indicates a range of two standard deviations above and below this mean. This means that, 95% of the time, the estimated parameter value will fall within this range.

For a feature position error of zero, the estimated parameter values are close but not equal to the ground-truth values. This is because, as discussed above, the gaze estimation error is lower when the parameters are slightly miscalibrated than when they are exactly equal to the ground-truth values.

For all four parameters, the mean estimated values remain quite close to the ground truth as feature position error increases. Somewhat surprisingly, though, the variations in the estimated values for r_{cornea} and r_{pc} become quite large when even small amounts of feature position error are introduced. Figure 3.7 (top) illustrates the reason for this: When r_{cornea} and r_{pc} are close to their true values and change by the same relative

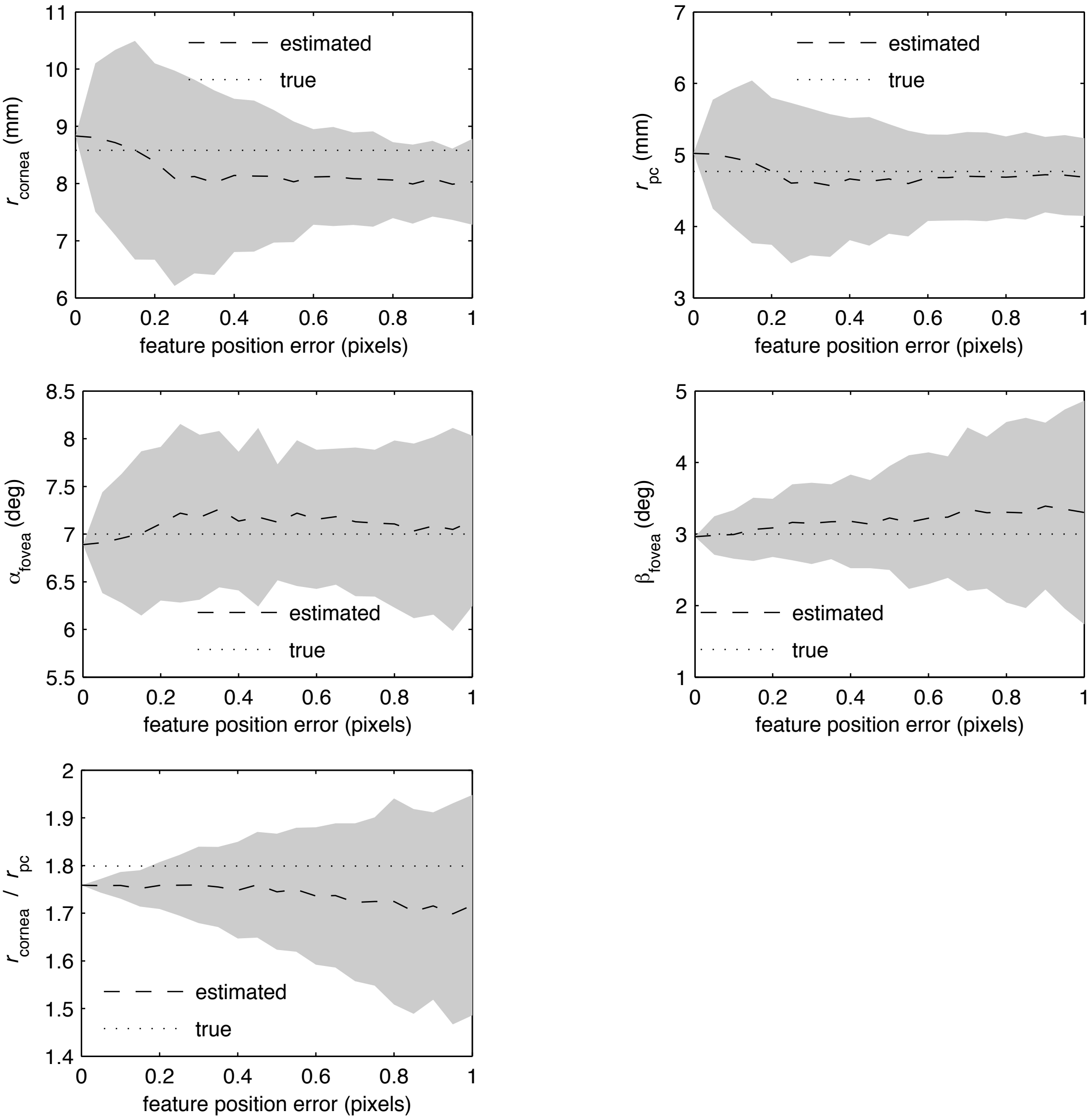


Figure 3.8: Estimated eye model parameter values as a function of feature position error. The dotted line plots the true parameter value; the dashed line plots the mean estimated parameter value, averaged over 100 runs; the shaded region indicates a range of two standard deviations above and below the mean.

amount, the gaze estimation error does not change very much. This means that even a small amount of feature position error can cause a relatively large change in the position of the minimum. If we look at the ratio between r_{cornea} and r_{pc} (Figure 3.8, bottom), we see that, for small feature position error, there is much less variation, confirming that r_{cornea} and r_{pc} change by approximately the same relative amount.

An even more surprising effect is that, for large feature position error, the variation in r_{cornea} and r_{pc} individually is actually smaller than for small feature position error. We hypothesise that this is because the relative influence of the prior becomes stronger as feature position error increases. The ratio between r_{cornea} and r_{pc}, on the other hand, behaves as expected, with the variation increasing in an approximately linear fashion with feature position error.

For α_{fovea}, the variation remains approximately constant, independent of feature position error; we hypothesise that the effects of increasing feature position error and of the prior cancel each other out. For β_{fovea}, finally, the variation increases approximately linearly with feature position error.

As we have already noted, most of the variation in the parameters, particularly for small feature position errors, is caused by the parameters varying jointly, rather than individually. Figure 3.9 (left) illustrates this with a scatter plot of r_{cornea} against r_{pc} (for a feature position error of 0.2 pixels); it is evident that the two parameters are strongly correlated. In fact, this is true for all four parameters. For every setting of the feature position error, we performed a principal component analysis (see e.g. [12]) on the parameter values obtained in the 100 calibrations. (All four parameters were normalized to a standard deviation of one.) Figure 3.9 (right) plots the eigenvalues of the covariance matrix as a function of feature position error. We see that, for small feature position error, almost all of the variation in the data is due to a single principal component, which corresponds to a joint scaling of r_{cornea} and r_{pc} and a corresponding adjustment of α_{fovea} and β_{fovea}.

Overall accuracy

Finally, we will test the overall accuracy of the eye tracker. We will calibrate the eye tracker with the eye at the centre of the eye position volume, then test the accuracy of the gaze estimation obtained using that calibration. Again, we will set each of the parameters of the simulated eye to one standard deviation above the population means used in the prior and, again, we will perform 100 runs for each condition to average out the effect of random variations in feature position error.

Figure 3.10 (left) shows the gaze estimation error as a function of feature position error, with the eye at the same position as for the calibration. Again, the shaded region indicates a range of two standard deviations above and below the mean; for comparison, we also show the gaze estimation error obtained by setting the eye model param-

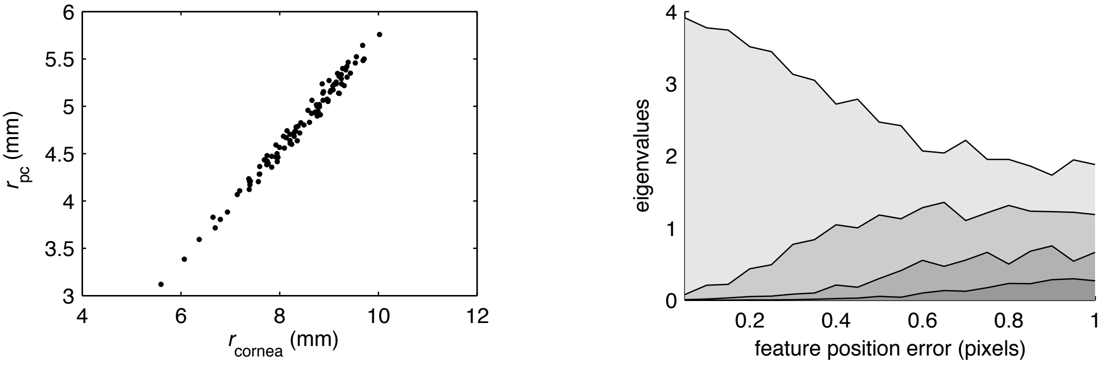


Figure 3.9: Left: Scatter plot of estimated values of r_{cornea} and r_{pc} for a feature position error of 0.2 pixels. Right: Result of a principal component analysis on the estimated parameter values. The plot shows the eigenvalues of the covariance matrix as a function of feature position error.

eters to their ground-truth values. For both calibrated and ground-truth parameter values, gaze error increases approximately linearly with feature position error, and the mean errors for both variants are quite close to each other, though the ground-truth parameter values produce less variation in the gaze error.

We see a notable difference between the two variants for a feature position error of zero: While the ground-truth parameter values produce a residual gaze error of around 0.4 degrees, the gaze error for the calibrated parameters values goes down to almost zero (the residual error is 0.02 degrees). This shows that the intentional slight miscalibration discussed earlier does in fact reduce the gaze error compared to a "perfect" calibration that sets the eye model parameters to their ground truth values.

This is true not only at the calibration position. Figure 3.10 (bottom left) shows the gaze error obtained for a feature position error of zero when the distance of the eye from the screen is varied (calibration was performed at a distance of 0.6 m). At all positions, the error obtained using the calibrated parameter values is lower than the error obtained using the ground-truth parameter values.

Finally, Figure 3.10 (bottom right) shows the result of the same test with a feature position error of 0.5 pixels. The gaze error for both calibrated and ground-truth parameters is around 1.5 degrees across the whole range of eye positions, increasing slightly with the distance of the eye from the camera. This is probably because, as the distance increases, the image of the eye becomes smaller, and so a given amount of feature position error makes a greater difference relative to the size of the eye. The variation in the error is larger for the calibrated parameter values than for the ground-truth parameter values.

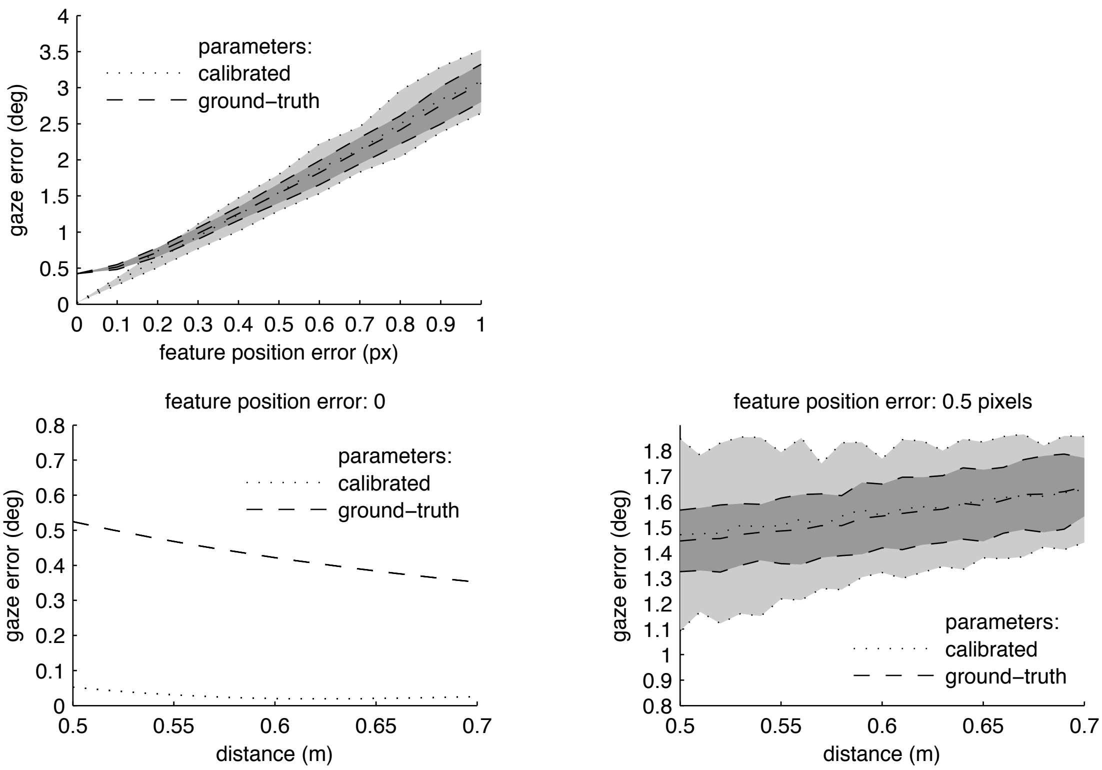


Figure 3.10: Overall accuracy of the simulated eye tracker. Results were averaged over 100 runs; the shaded region indicates a range of two standard deviations above and below the mean. Top: Gaze estimation error at the calibration position as a function of feature position error. Bottom: Gaze estimation error obtained with the eye at different distances from the screen for a feature position error of 0 (left) and 0.5 pixels (right). Calibration was performed at a distance of 0.6 m from the screen.

Discussion of the results

Simulation has not only allowed us to validate that the gaze estimation and calibration algorithms function correctly; it has also given us an idea of the accuracy that needs to be achieved in the image analysis component of the real eye tracker to achieve a certain gaze estimation accuracy. Referring to Figure 3.10 (top), we see that if we want to achieve an accuracy of 1 degree, we need to measure the position of image features to within an accuracy of 0.3 pixels. This is a relatively demanding requirement but one that still seems achievable. For the calibration step, in particular, we know that the eye is (nearly) stationary when fixating a calibration point, so we can reduce the noise in the measured position of the image features by averaging over several frames.

3.6.2 Hardware Implementation

For the hardware implementation of the remote eye tracker, the image analysis algorithms (Section 3.2) and gaze estimation algorithms (Section 3.3) were implemented in C++, running under the Windows XP operating system.

The gaze estimation algorithm requires the position of the illuminators and the display relative to the camera to be known; these positions were measured using a ruler and a protractor.

The eye tracker was tested on four subjects, all of whom were Caucasians and none of whom wore glasses or contact lenses. Each subject calibrated the eye tracker with the head at the centre of the eye position volume using a 5-point calibration pattern; the accuracy of the eye tracking was then measured for different head positions in the eye position volume using a rectangular grid of 9 gaze targets, 5 of which were identical to the calibration points. The grid had a width of 27 cm and a height of 22 cm. For each gaze target, 40 gaze samples were collected, and the root mean square (RMS) error across all samples was computed. No temporal filtering or averaging was applied to the gaze samples. A chin rest was used to position the subject's head accurately, but, of course, no chin rest is needed during actual use.

The eye model parameters were set to population averages because initial experiments had shown that they could not be estimated reliably from the calibration data. We speculate that this is because the relative positions of the system components were not known with sufficient accuracy; this issue is discussed in more detail in Section 3.7.

A bilinear recalibration function (see Section 3.5) was used to compensate for remaining systematic errors in the gaze estimation.

Table 3.2 shows the results of the accuracy measurements; for each head position, the error averaged over all gaze targets and subjects is reported. The lowest error, around one degree, was obtained with the head in the calibration position. Moving the head parallel to the display plane caused the error to increase by 0.2–0.4 degrees, while moving the head forwards and backwards increased the error more strongly, by around 1–1.5 degrees. The main reason for this is that, at the front and back of the eye position volume, the image becomes slightly defocused, so that the position of the corneal reflexes and the pupil cannot be measured as accurately.

The average error over all head positions was 1.57 degrees; other systems (e.g. [2, 122, 127]) report accuracies between 0.5 and 1 degrees, but note that these systems typically use temporal filtering on the gaze samples, which was not used here.

The temporal resolution of the system is constrained by the frame rate of the camera, which runs at 15 frames per second, but the software itself is capable of running at higher frame rates; on the test system, 37.4 ms were required to process a camera frame, which would allow a frame rate of 26 frames per second. Most of the processing time is spent on computing the DOG filter on the complete image; once the eyes have been

Head position		rms gaze error
centre (calibration position)	(0, 0, 0)	1.01°
left	(−10, 0, 0)	1.33°
top	(0, 10, 0)	1.23°
top left	(−10, 10, 0)	1.44°
front	(0, 0, −10)	2.54°
back	(0, 0, 10)	1.88°
Overall		1.57°

Table 3.2: Gaze accuracy for different head positions, averaged over four test subjects. Head position coordinates (in centimetres) are relative to the centre of the eye position volume.

found, one could restrict the processing in subsequent frames to a region of interest (roi), since it is unlikely that the eyes will move much from frame to frame. This strategy would considerably reduce the amount of computation required and allow a corresponding increase in the frame rate (see also the discussion of rois in Section 3.7).

3.7 Avenues for Future Work

While the eye tracker we have presented has satisfactory performance, we have already hinted at several areas in which it could be improved.

We have used simulation to validate that the algorithm for estimating the user-specific eye model parameters works and is reasonably robust to feature position error. However, the current hardware implementation does not estimate these parameters reliably; this is why the results presented in Section 3.6.2 used population averages for these values. We speculate that the reason the parameters cannot be estimated is that the positions of the illuminators and the display relative to the camera were not measured accurately enough. We used a ruler and a protractor, which is obviously not a very accurate way of making these measurements, particularly since the reference point for the measurements – the camera's optical centre – lies inside the camera and its position is not marked on the camera or lens.

We believe a more accurate approach would be to use the camera itself to make the measurements, similar to the method described by Beymer and Flickner [11], as follows: Part of a planar mirror is covered with a chessboard pattern. The mirror is then held in front of the camera, thus reflecting an image of the illuminators and the display back at the camera. From the chessboard pattern on the mirror, we can determine the spatial orientation of the mirror plane; from this and the location of the illuminators in the camera image, we can determine the position of the illuminators relative to the camera. To determine the position of the display, we display calibration marks at known pixel positions on the display. From the position of these calibration marks in the camera image (observed via the mirror), we can likewise deduce their position relative to the camera.

Another obvious deficiency of the current eye tracker is the low temporal resolution of 15 Hz, which is quite low even for HCI (human-computer interaction) applications. The frame rate of 15 Hz is imposed by the USB interface of the camera, so an obvious way to increase it would be to use an interface with higher bandwidth. If one wants to stay with the low-cost consumer-grade USB interface, there is another option: Most current industrial cameras can be configured to read out only part of the pixel array – a *region of interest* (ROI) or *area of interest* (AOI); the smaller the number of pixels read out, the higher the frame rate. In the eye tracker, one could restrict the ROI to a rectangular region containing the eyes. If the head moves, the ROI is shifted accordingly, and a full frame has to be acquired only on the first frame or when tracking of the eyes is lost. In addition to increasing the camera frame rate, defining an ROI in this way also reduces the amount of computation that needs to be carried out in the image analysis step.

The camera that we used (see Section 3.1) does offer the option to define an ROI; unfortunately, we discovered that changing the position of the ROI caused a pause of around half a second during which the camera did not deliver any image data. This made the ROI feature unattractive to us because any head movement would cause a corresponding pause in the eye tracker data stream. However, with a camera that can change the ROI position more quickly, it should be easy to implement a shifting ROI.

On a more general note, most remote eye trackers currently in existence, both commercial and academic, are roughly similar in terms of their performance and other characteristics, and the eye tracker we have presented here is no exception. A contemporary remote eye tracker typically achieves an accuracy of between half a degree and a degree and allows head movements within a box measuring around 20 cm on each side. Most trackers still have difficulty tracking some users, particularly those wearing glasses, and the tracker needs to be calibrated for each individual user. On the hardware side, remote trackers typically use industrial cameras with high-grade glass optics, leading to high manufacturing costs, even at volume.

These characteristics themselves suggest ways in which current eye trackers could be improved, both to make them more useful for current applications and to enable their use in new areas.

Cost One of the reasons that eye trackers have not yet found their way into consumer applications is the cost of making them. The camera (including the lens) accounts for much of the cost, so this is where large savings could be made: The ⅔-inch image sensor on our current eye tracker measures 8.6 × 6.9 mm, whereas web cameras often use ¼-inch sensors, which measure 3.6 × 2.7 mm, i.e. only about a sixth of the area. Similarly, our tracker uses a lens consisting of multiple glass elements, whereas web cameras often use only a single element, made from plastic. Obviously, there is potential for reducing the cost, but equally obviously, lower-quality components do an inferior job. A smaller image sensor has a lower signal-to-noise ratio, and a simple lens has lower resolution and produces more distortion than a more sophisticated one. Higher-quality algorithms may be able to compensate for lower-quality hardware, but only to a certain degree; the challenge is to see how cheap a system can be while still remaining useful for certain applications at least.

Another obvious area for cutting cost is to eliminate the infrared illuminators and use natural illumination (see e.g. [59]); again, though, this makes the image analysis task significantly more difficult.

Tolerance towards glasses Many eye trackers that use active illumination have trouble tracking users who wear glasses because the illumination causes reflections to appear on the glasses. Depending on the orientation of the head, these reflections can obscure the eye entirely. A possible solution to this problem would be to add more illuminators to the system; then, if the system detected that the eyes were being obscured by reflections on the glasses, it could switch to a different set of illuminators, which would change the position of the reflections, making the eyes visible again.

Another consideration with users wearing glasses is that the glasses distort the image of the eye observed through them. This might make it necessary to incorporate an optical model of the glasses into the gaze estimation algorithm, though, as far as we have observed, accuracy still remains tolerable for moderate-strength glasses even if their effect is not modelled.

User-specific calibration Current eye trackers must be calibrated to account for the individual size and shape of the user's eyes; this is usually done by asking the user to fixate a certain number of calibration targets. Because this places a certain burden on the user and requires the user to cooperate, it would be desirable to eliminate the calibration phase. The difficulty, however, is that certain parameters of the eye, particularly the foveal displacement, cannot be measured easily or unobtrusively.

Alternatively, if we know the scene the user is viewing or the task the user is executing, we may be able to infer – with a certain probability – where the user may be looking. For example, if the user is interacting with a graphical user environment using a mouse, the user will often (though not always) look at the mouse cursor when clicking the mouse. In the following chapter, we will show how we can use this information to perform an automatic, unobtrusive calibration.

4 Automatic User-Specific Calibration

As we saw in Chapter 3, eye trackers require user-specific calibration because the size and shape of the eye varies from person to person [41]. In current eye trackers, the calibration phase typically involves the user having to fixate one or more calibration targets. It should be possible to measure some of the relevant parameters – such as the corneal radius – without requiring the user to fixate known targets, but it seems that at least one parameter cannot realistically be measured without knowing where the user is fixating: This parameter is the offset of the fovea from the optical axis. Measuring the position of the fovea directly would require an image of the retina obtained through ophthalmoscopy, which involves shining a bright light into the user's eye and imaging it from a short distance. Such a procedure would be more burdensome than simply requiring the user to fixate a few calibration targets.

Nevertheless, it would of course be appealing if user-specific calibration could be eliminated entirely, particularly in situations with rapidly changing users, such as web usability studies. Also, the less obtrusive the eye tracking process is, the smaller the chance that users will make different eye movements than they ordinarily would because they are aware that they are being tracked.

In this chapter, we will present an automatic calibration technique that requries no explicit calibration phase and can be used to calibrate the system continually while it is in use. The technique is based on the observation that in many settings, we know the visual scene that the user is looking at, and this allows us to make educated guesses as to where the user is looking. These guesses may be wrong much of the time, but if we gather enough of them, we may still be able to exploit them to perform calibration.

The particular setting we will use here is that of a user interacting with a graphical user environment (GUI) using a mouse. When a mouse click occurs, the user will often (though not always) be looking at the mouse cursor (see Section 4.2).

Because the click location coincides with the gaze position only a certain percentage of the time, we cannot use click locations as input to conventional calibration algorithms, which assume that the gaze target is fixated accurately every time. Instead, we use a probabilistic approach that models the distribution of gaze positions relative to the click location as a mixture of two distributions: The first is a Gaussian with low variance and represents the case where the user was in fact looking at the click location. The second is a uniform distribution across the whole screen, representing the case where the user was looking elsewhere. We then use Expectation Maximization (EM) to estimate the maximum likelihood calibration (see Section 4.3).

We performed experiments in a web browsing scenario (see Section 4.4). The average gaze error without calibration was 181 pixels (or 5.2 degrees). Using a conventional calibration with a nine-point calibration pattern, we obtained an average error of 42.0 pixels (or 1.2 degrees); with the automatic calibration, the average error was 49.3 pixels (or 1.4 degrees), 17.4% more than for the conventional calibration. The calibration was based on around 150–220 clicks per user, corresponding to around 15 minutes of web browsing.

4.1 State of the Art

The idea of automatic calibration does not appear to have been investigated much in the existing literature, despite its obvious appeal. The only relevant publication we were able to find concerns the EyeCatcher system [60], a gaze-controlled human-computer interface that uses so-called "EyeCons" as gaze targets, square buttons showing an image of an eye. When the user gazes at an EyeCon, the eye on the button starts to close; if the user maintains fixation until the eye is fully closed, the corresponding user interface item is selected. To make the system robust against miscalibration, an EyeCon is activated even if the gaze falls slightly outside its border. Because of the visually salient nature of the EyeCon as well as its semantically dominant role in the user interface, the system assumes that whenever an EyeCon is selected, the user was looking right at the graphical depiction of the eye; this information is then used for local recalibration.

Several commercially available eye trackers name "automatic calibration" among their product features; see for example [128]. However, this seems to mean simply that the tracker switches from one calibration point to the next automatically.

Several patents exist describing "automatic calibration" or "self-calibration". One approach [108] is based on histograms of gaze positions; from the figures given in the patent, it appears that such histograms usually exhibit a peak at the eye's rest position and fall off towards the sides. A new user may be calibrated by comparing their histogram with that of a previously calibrated reference user and adjusting the calibration until the histograms match.

Another patent [85] claims an "automatic recalibration". However, the approach relies on "attractors with constant or variable [...] size" that may be "stationary or [...] moving" and "attract the attention of the curious observer automatically". Calibration then proceeds on the assumption that the attractors are sufficiently salient to attract the user's gaze. In effect, then, this approach is similar to a traditional calibration

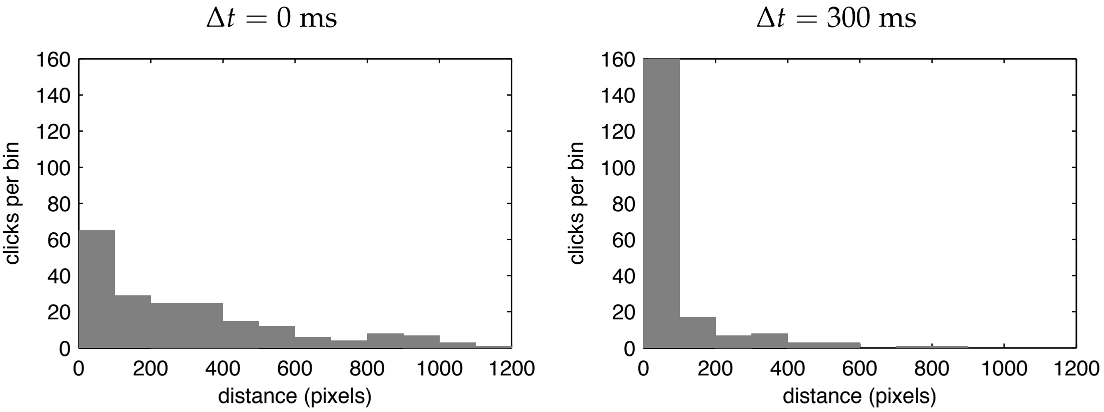


Figure 4.1: Histogram of the distance between click location and gaze position. Left: No time offset was applied, i.e. gaze position was measured at the time of the click. Right: Gaze position was measured $\Delta t = 300$ ms before the click occurred.

procedure as described above, only with more refined gaze targets. The procedure requires a specific visual stimulus and is thus not transparent to the user.

4.2 Motivation

To assess how often and with what degree of accuracy the user fixates the mouse cursor when clicking, we recorded about 25 minutes of eye movement, mouse movement and mouse button presses on a subject who was freely browsing the web, without any specific task. We used an SMI iView X Hi-Speed 1250 eye tracker [121], a fixed-head system with 1250 Hz sampling rate; the resolution of the screen was 1280 × 960 pixels. For this recording, the eye tracker was calibrated.

We extracted clicks from the recorded mouse data; for each click, we determined the corresponding gaze position. Since experienced users may already be shifting their gaze away from the click target by the time the click occurs, we used the position of gaze a certain time offset Δt before the click, with the goal of determining the value for Δt that would maximize the agreement between click location and gaze position.

Figure 4.1 shows histograms of the distance between click location and gaze position. The left plot is for an offset of $\Delta t = 0$, i.e. gaze position was measured at the time that the click occurred. The right plot is for $\Delta t = 300$ ms, i.e. gaze position was measured 300 milliseconds before the click occurred. It is apparent that the latter plot is more "peaked", i.e. that there was better agreement between click and gaze when gaze was measured 300 milliseconds before the click.

To quantify this effect, we computed the entropy of the distance between click and gaze for various values of Δt. Figure 4.2 shows a plot of the entropy as a function of

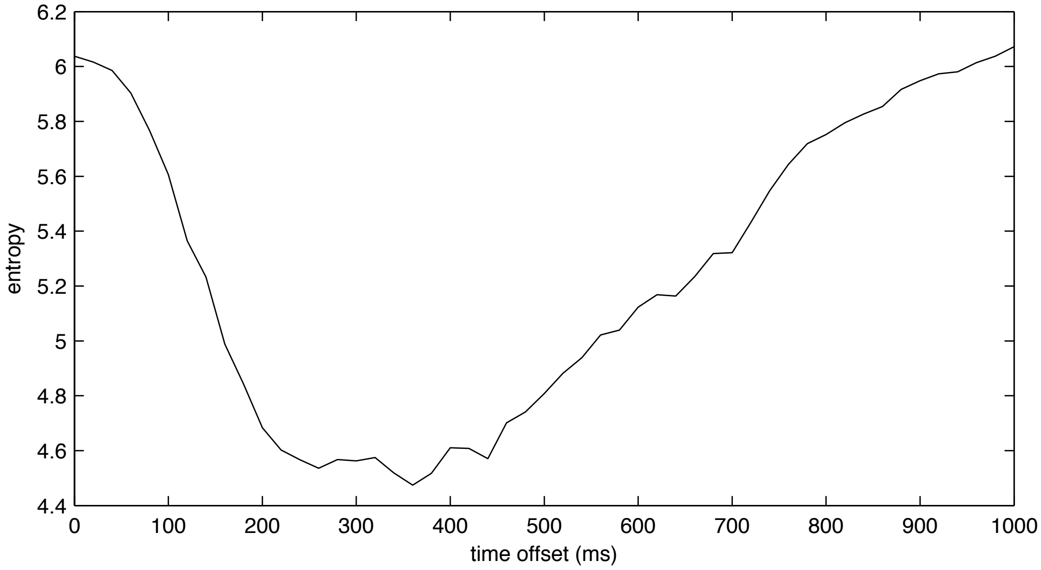


Figure 4.2: Entropy of the distance between click location and gaze position as a function of Δt; gaze is measured at a time offset Δt before the click.

Δt, with a minimum at $\Delta t \approx 360$ ms, indicating that this Δt gave the best agreement between click and gaze. (To estimate the entropy, we used an m_n-spacing estimator [9, Equation (16)] with $m_n = \lfloor \sqrt{n} \rfloor$.)

Figure 4.3 shows a scatter plot of gaze positions relative to the click location, which is at the origin of the plot; the plot uses the optimal offset $\Delta t = 360$ ms determined above. There is a prominent cluster of points around the origin, which corresponds to cases when the user looked at the mouse cursor when clicking. Note that the cluster appears to be somewhat wider than it is tall; one explanation for this could be that in the web browsing scenario, most click targets (such as web links and page tabs) are also wider than tall.

4.3 Method

Our approach to automatic calibration is as follows: We start out with the eye tracker set to a "standard" calibration. In our case, this is simply the calibration obtained for an arbitrarily chosen test user; a more sophisticated approach might be to choose a calibration that yields the smallest average error over a diverse group of test subjects.

We seek a function ϕ that translates the uncalibrated gaze position $\hat{g}$ (measured using the "standard" calibration) into the true gaze position g, i.e. $\phi(\hat{g}) = g$. The function ϕ can be chosen to take different forms, such as a bilinear or biquadratic interpolation function.

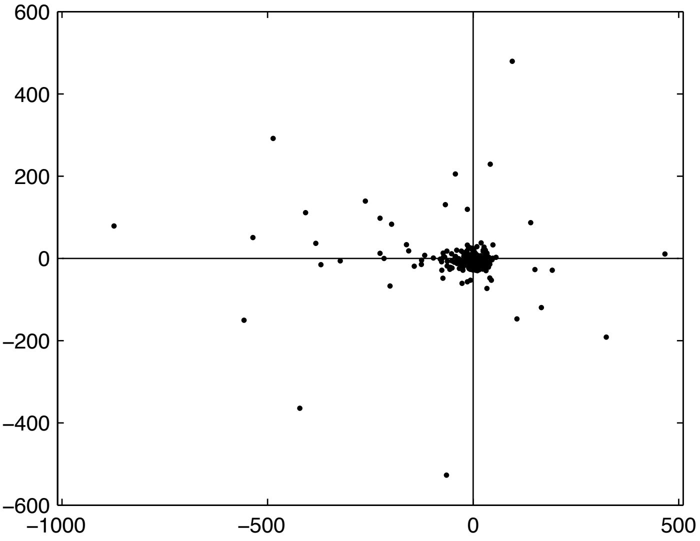


Figure 4.3: Scatter plot of (true) gaze positions relative to click location (which is at the origin of the plot). Gaze was measured at a time offset of $\Delta t = 360$ ms before the click. Coordinates are in pixels.

As the user begins interacting with the system, we start collecting mouse click locations while simultaneously recording gaze data. As more and more click positions are collected, we compute a continually refined calibration, using a procedure that will be described below. In principle, we can continue refining the calibration indefinitely. However, if we wish to limit the amount of computation required, we can choose to stop refining the calibration when it has stabilized, i.e. when the change in the calibration falls below a certain threshold.

While the accuracy of the calibration is low during the initial period of interaction, we can recalibrate the whole of the recorded gaze data using the more accurate calibration available at the end of the session. Thus, if the gaze data is not needed for online, interactive purposes but recorded for later evaluation (as is typical for "diagnostic" applications such as web usability studies), we achieve high gaze accuracy during the whole session. If the calibration for the specific eye tracker is expected to "drift" over time, we can choose to use only the last n clicks for calibration, thus continually adjusting the calibration.

Assume, then, that we have acquired n click locations $\mathbf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$ along with a corresponding gaze recording. Automatic calibration then proceeds as follows.

The first step is to determine the time offset Δt that gives the best agreement between click and gaze, as explained in Section 4.2. Using this time offset, we extract n gaze positions $\hat{\mathbf{G}} = (\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_n)$ from the gaze recording that correspond to the n click locations.

We now seek to find the transformation ϕ between measured and true gaze positions. To do this, we will define a suitable likelihood function

$$P(\hat{\mathbf{G}}|\mathbf{C}, \phi), \tag{4.1}$$

i.e. the likelihood that (uncalibrated) gaze positions $\hat{\mathbf{G}}$ will be measured given click locations $\mathbf{C}$ and the transformation ϕ. We then choose the transformation ϕ that maximizes this likelihood function.

As a basis for defining the likelihood function, we use the conditional probability $P(\boldsymbol{g}|\boldsymbol{c})$, i.e. the conditional probability that the true gaze position is $\boldsymbol{g}$, given that a click occurred at position $\boldsymbol{c}$. Figure 4.3 shows a scatter plot of true gaze positions relative to the click position. Given our observations in Section 4.2, it seems reasonable to model $P(\boldsymbol{g}|\boldsymbol{c})$ as a mixture of two distributions, conditioned by a latent variable z, where $z = 1$ corresponds to the case that the user looked at the click target, and $z = 0$ corresponds to the case that the user was looking elsewhere:

$$P(\boldsymbol{g}|\boldsymbol{c}) = P(z = 1)P(\boldsymbol{g}|\boldsymbol{c}, z = 1) + P(z = 0)P(\boldsymbol{g}|\boldsymbol{c}, z = 0). \tag{4.2}$$

We model the first case (user was looking at the click target, $z = 1$) as a Gaussian distribution, i.e. $P(\boldsymbol{g}|\boldsymbol{c}, z = 1) = \mathcal{N}(\boldsymbol{g} - \boldsymbol{c}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu} = \mathbf{0}$; the covariance matrix $\boldsymbol{\Sigma}$ will be estimated from the data. For the case where the user was not looking at the click target ($z = 0$), we assume a uniform distribution across the screen, i.e. $P(\boldsymbol{g}|\boldsymbol{c}, z = 0) = \frac{1}{w \cdot h}$, where w and h are the width and height of the screen.

Using this model, we obtain the following expression for the likelihood:

$$\begin{aligned} P(\hat{\mathbf{G}}|\mathbf{C}, \phi) &= \prod_{i=1}^{n} P(\hat{\boldsymbol{g}}_i|\boldsymbol{c}_i, \phi) \\ &= \prod_{i=1}^{n} P(\boldsymbol{g}_i = \phi(\hat{\boldsymbol{g}}_i)|\boldsymbol{c}_i) \\ &= \prod_{i=1}^{n} \left(P(z = 1)\mathcal{N}(\phi(\hat{\boldsymbol{g}}_i) - \boldsymbol{c}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) + P(z = 0)\frac{1}{w \cdot h} \right). \end{aligned} \tag{4.3}$$

To find the transformation ϕ and covariance matrix $\boldsymbol{\Sigma}$ that maximize the likelihood function, we use the Expectation Maximization (EM) algorithm (see for example Chapter 9 in [12]). Expectation Maximization is commonly used to find the maximum of likelihood functions, such as this one, that depend on latent variables (in this case, z). In other words, the likelihood function $P(\boldsymbol{X}|\boldsymbol{\theta})$ (where $\boldsymbol{X}$ are the observed variables and $\boldsymbol{\theta}$ are the model parameters to be estimated) is obtained by marginalizing a distribution $P(\boldsymbol{X}|\boldsymbol{\theta}, \boldsymbol{Z})$ over the latent variables $\boldsymbol{Z}$. The Expectation Maximization algorithm alternates between two steps: The E (Expectation) step estimates the posterior distribution $P(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{cur}})$ of the latent variables given the observed data $\boldsymbol{X}$ and a current estimate $\boldsymbol{\theta}_{\text{cur}}$ for the parameters; the M (Maximization) step finds new parameter values $\boldsymbol{\theta}_{\text{new}}$ that maximize the expected log likelihood given the distribution of the latent variables estimated in the E step.

In our case, the parameters $\boldsymbol{\theta}$ comprise the parameters of the calibration function as well as the covariance matrix $\boldsymbol{\Sigma}$. We assume that the calibration function is parameterized by parameters $\boldsymbol{\alpha}$, and we will therefore in the following denote it by $\phi_{\boldsymbol{\alpha}}$ to emphasize that it depends on $\boldsymbol{\alpha}$. The parameters $\boldsymbol{\theta}$ for which we wish to find a maximum-likelihood estimate are then the calibration parameters $\boldsymbol{\alpha}$ as well as the covariance matrix $\boldsymbol{\Sigma}$.

In the M step of the EM algorithm, we need to find the new parameter values $\boldsymbol{\theta}_{\text{new}}$ that maximize the likelihood for a given distribution $P(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{cur}})$ of the latent variables; we find $\boldsymbol{\theta}_{\text{new}}$ by setting the derivative of the log likelihood to zero. We will do this for $\boldsymbol{\alpha}$ and $\boldsymbol{\Sigma}$ individually.

Let us begin with $\boldsymbol{\alpha}$. The derivative of the log likelihood with respect to $\boldsymbol{\alpha}$ is

$$\frac{\partial}{\partial \boldsymbol{\alpha}} \ln P(\hat{\mathbf{G}}|\mathbf{C}, \phi_{\boldsymbol{\alpha}}) = \\ -\sum_{i=1}^{n} \underbrace{\frac{P(\boldsymbol{g}_i = \phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i)|\boldsymbol{c}_i, z=1)P(z=1)}{P(\boldsymbol{g}_i = \phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i)|\boldsymbol{c}_i)}}_{\gamma_i} \left(\phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i) - \boldsymbol{c}_i\right)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \left(\frac{\partial}{\partial \boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i)\right), \quad (4.4)$$

where the γ_i are the so-called *responsibilities*.

A common special case is that $\phi_{\boldsymbol{\alpha}}$ is linear in $\boldsymbol{\alpha}$, i.e. $\phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i) = \boldsymbol{J}(\hat{\boldsymbol{g}}_i)\boldsymbol{\alpha}$, where $\boldsymbol{J}(\hat{\boldsymbol{g}}_i) = \frac{\partial}{\partial \boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}_i)$ is the Jacobian, which depends only on $\hat{\boldsymbol{g}}_i$. In this case, Equation 4.4 becomes

$$\frac{\partial}{\partial \boldsymbol{\alpha}} \ln P(\hat{\mathbf{G}}|\mathbf{C}, \phi_{\boldsymbol{\alpha}}) = -\sum_{i=1}^{n} \gamma_i \left(\boldsymbol{J}(\hat{\boldsymbol{g}}_i)\boldsymbol{\alpha} - \boldsymbol{c}_i\right)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{J}(\hat{\boldsymbol{g}}_i). \quad (4.5)$$

To find the $\boldsymbol{\alpha}$ that maximizes the log likelihood, we set the above expression to zero, obtaining a linear system of equations for the unknown $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha}^{\mathrm{T}} \sum_{i=1}^{n} \gamma_i \boldsymbol{J}(\hat{\boldsymbol{g}}_i)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{J}(\hat{\boldsymbol{g}}_i) = \sum_{i=1}^{n} \gamma_i \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{J}(\hat{\boldsymbol{g}}_i). \quad (4.6)$$

We solve this system to find the updated values for the calibration parameters. The system matrix is nonsingular if at least one of the $\boldsymbol{J}(\hat{\boldsymbol{g}}_i)$ has full rank. In this case, since $\boldsymbol{\Sigma}^{-1}$ is positive definite, at least one of the $\boldsymbol{J}(\hat{\boldsymbol{g}}_i)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{J}(\hat{\boldsymbol{g}}_i)$ is also positive definite, and all of them are positive semidefinite. Hence, because $\gamma_i > 0$ for all i, the system matrix $\sum_i \gamma_i \boldsymbol{J}(\hat{\boldsymbol{g}}_i)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{J}(\hat{\boldsymbol{g}}_i)$ is also positive definite.

As a concrete example, let us choose $\phi_{\boldsymbol{\alpha}}$ as follows:

$$\phi_{\boldsymbol{\alpha}}(\hat{\boldsymbol{g}}) = \begin{pmatrix} \alpha_1 \hat{g}_1 + \alpha_2 \hat{g}_2 + \alpha_3 \hat{g}_1 \hat{g}_2 + \alpha_4 \\ \alpha_5 \hat{g}_1 + \alpha_6 \hat{g}_2 + \alpha_7 \hat{g}_1 \hat{g}_2 + \alpha_8 \end{pmatrix} \quad (4.7)$$

(where $\hat{\boldsymbol{g}} = (\hat{g}_1, \hat{g}_2)^\mathrm{T}$). The corresponding Jacobian is

$$\boldsymbol{J}(\hat{\boldsymbol{g}}) = \begin{pmatrix} \hat{g}_1 & \hat{g}_2 & \hat{g}_1\hat{g}_2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \hat{g}_1 & \hat{g}_2 & \hat{g}_1\hat{g}_2 & 1 \end{pmatrix}, \tag{4.8}$$

and it is obvious that $\boldsymbol{J}(\hat{\boldsymbol{g}})$ always has full rank, independent of $\hat{\boldsymbol{g}}$.

We turn now to the covariance matrix $\boldsymbol{\Sigma}$, for which we use the usual update rule as derived in [12, Chapter 9]:

$$\boldsymbol{\Sigma}_\mathrm{new} = \frac{1}{\sum_{i=1}^n \gamma_i} \sum_{i=1}^n \gamma_i (\phi(\hat{\boldsymbol{g}}_i) - \boldsymbol{c}_i)\ (\phi(\hat{\boldsymbol{g}}_i) - \boldsymbol{c}_i)^\mathrm{T}. \tag{4.9}$$

The rest of the EM algorithm proceeds in the standard way.

4.4 Results

We evaluated the automatic calibration algorithm as follows. Four test subjects with normal or corrected-to-normal vision were asked to perform a fifteen-minute web browsing task in the style of a web usability study. During this task, we recorded the subjects' eye movements using an SMI iView X RED eye tracker [122], a remote eye tracker with a sampling rate of 50 Hz. The eye tracker was set to a "standard" calibration that had been obtained with a user who otherwise did not participate in the experiment. In addition to gaze data, we also recorded mouse movements and mouse button presses. The subjects were seated with their head around 60 cm from a display with a resolution of 1280 × 1024 pixels and a visible area of 38 × 30 cm. One degree of visual angle thus corresponded to around 35 pixels.

The subjects were told that the goal of the experiment was to analyze the eye movements that people make when interacting with web sites. We purposely did not tell subjects that we were recording mouse movements, nor did we tell them that we were interested in the relationship between eye and mouse movements; this was to avoid demand characteristics, an experimental effect where participants modify their behaviour according to the perceived purpose of the experiment [66, 104]. In this case, we were concerned that participants might fixate the mouse cursor more often than they normally would if they knew that the experiment was concerned with the relationship between eye and mouse movements.

After the web browsing task was completed, we asked subjects to fixate a nine-point calibration pattern that would later be used to perform a post-hoc "conventional" calibration of the eye tracker. We then asked subjects to fixate a second, 25-point grid; this was used to compute the calibration error of the "automatic" and the "conventional" calibration as the root mean square error over all fixation targets. Both the automatic and the conventional calibration used the calibration function from Equation 4.7.

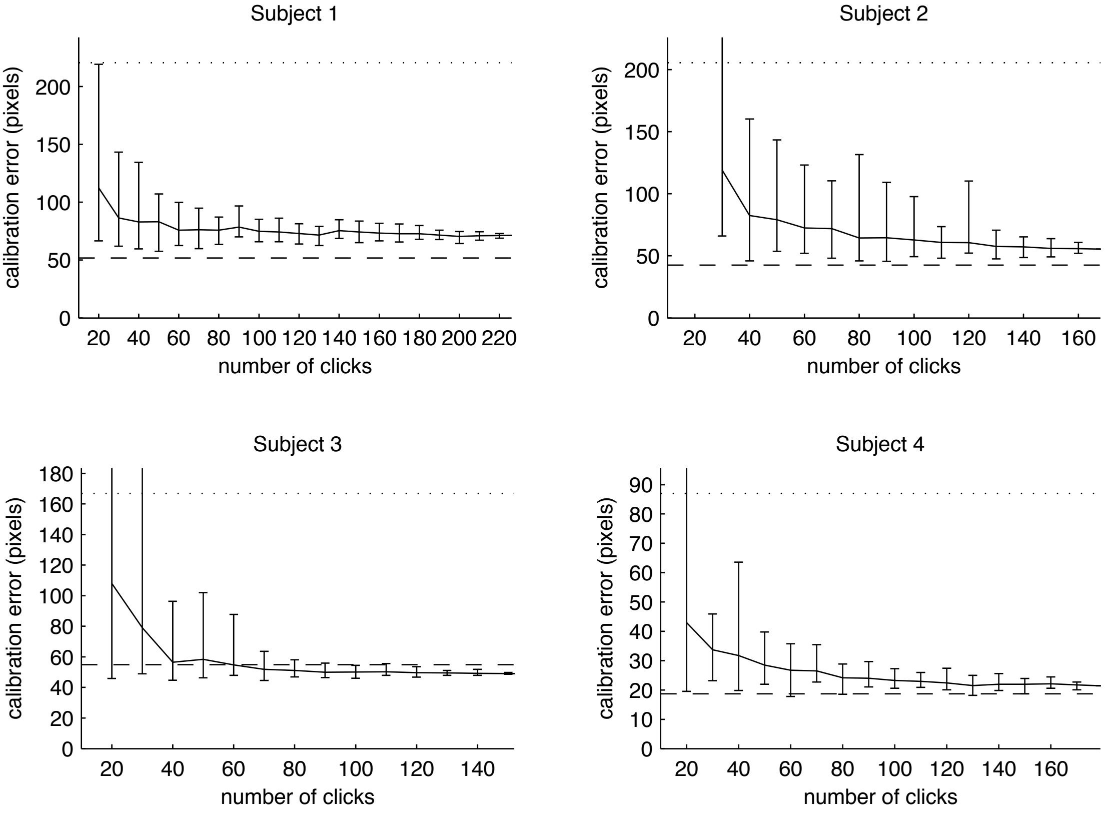


Figure 4.4: Calibration error in pixels as a function of the number of mouse clicks available for the calibration. The continuous line plots the average calibration error, averaged over 20 runs for a given number of clicks; error bars show the minimum and maximum error over all runs. The dashed line shows the calibration error of a conventional calibration, the dotted line shows the error obtained when no calibration at all is performed.

		calibration			
	clicks	none	conventional	automatic	
Subject 1	226	220.5 px	51.8 px	71.4 px	(+37.8%)
Subject 2	168	205.5 px	42.6 px	55.4 px	(+30.1%)
Subject 3	152	166.8 px	54.9 px	49.0 px	(-10.7%)
Subject 4	179	87.0 px	18.7 px	21.4 px	(+14.6%)
Average	182	197.6 px	42.0 px	49.3 px	(+17.4%)

Table 4.1: Calibration error in pixels for uncalibrated gaze data as well as conventional and automatic calibration. The relative difference between the automatic and conventional calibration is given in the last column. For each subject, the complete set of recorded clicks was used; the number of clicks is given in the table.

Figure 4.4 shows the results. For each user, we plot the calibration error of the automatic calibration as a function of the number of clicks available for the calibration. For a given number of clicks, we selected a random subset of that size from all the clicks, then performed an automatic calibration using those clicks. Because the accuracy of the calibration can vary depending on whether a "good" or a "bad" subset of clicks happens to be chosen, we performed 20 runs for each number of clicks. The plots show the average calibration error over all runs using a continuous line, with error bars for the minimum and maximum calibration error. Additionally, we plot the calibration error for the conventional calibration and with no calibration at all (using only the standard calibration that the eye tracker was set to).

It is evident that, for all subjects, the automatic calibration reduced the error considerably compared to the uncalibrated case; in one case (Subject 3), the accuracy of the automatic calibration was comparable to that of the conventional calibration. While acceptable calibrations can sometimes be obtained with around 50 clicks, around 100 clicks are necessary to reliably obtain a good calibration. For the tasks in our experiments, 100 clicks correspond to between 5.3 and 10.9 minutes of web browsing, depending on the subject.

Table 4.1 gives the numerical values for the calibration error achieved per subject when all recorded clicks were used. The relative difference between the conventional and the automatic calibration is also given. This varies between -10.7% for Subject 3 (where the automatic calibration was actually slightly better than the conventional calibration) to +37.8% for Subject 1.

The average error across all subjects was 42.0 pixels (1.2 degrees) for the conventional calibration and 49.3 pixels (1.4 degrees), or 17.4% more, for the automatic calibration.

4.5 Discussion and Outlook

We have shown that in a human-computer interaction setting where a mouse or other pointing device is used, an eye tracker can be calibrated using only the information of where the user clicked the mouse; 100 mouse clicks are sufficient to reliably obtain a good calibration. The calibration error for the automatic calibration is typically slightly higher than for a conventional calibration; for one subject, however, the automatic calibration was actually more accurate than the conventional calibration.

The calibration becomes progressively better over time the more data is recorded; for non-interactive applications where the gaze data is stored and analyzed offline, the whole of the recorded data can be recalibrated using the calibration computed at the end of the session.

An obvious advantage of the approach is that it eliminates the tedious calibration phase – the user can just "sit down and go". This is particularly valuable in settings with rapidly changing users such as web usability studies. Also, the less obtrusive the eye tracking is, the smaller the chance that users' awareness of the fact that they are being eye tracked will cause them to make different eye movements than they normally would. Finally, in public installations without supervision, it can be hard to get users to perform a calibration procedure properly [60, Section 2.3].

The current technique only works in settings where a mouse is used for interaction. If we wish to drop this requirement, we must look for other clues to tell us where the user might be looking. The most general but probably most challenging scenario would be to calibrate the eye tracker based solely on the knowledge of the scene the user is looking at. Algorithms that compute visual saliency (see e.g. [71, 76, 130]) could be used to quantify the probability that the user will look at certain parts of the scene. This probability distribution could then be used in a maximum-likelihood approach similar to that of Section 4.3 to estimate the calibration.

The success of such a scheme would depend on the accuracy with with the saliency algorithm is able to predict where gaze is likely to fall. Apart from the shortcomings of any individual saliency algorithm, we note that top-down, task-driven influences place a principal limit on how accurately gaze can be predicted based on knowledge of the visual stimulus alone. Nevertheless, we believe that such a calibration scheme should still be feasible in many settings, particularly when the visual stimulus contains strong bottom-up cues for eye movements, such as in video games. One may also introduce visual elements into the scene that are designed to attract the user's gaze with high probability (see for example [85]), but such a manipulation of the scene may be undesirable, and it is debatable whether such an approach can still be called "automatic" calibration – after all, the traditional calibration stimulus of a lone fixation target on a uniform background is itself a fairly strong gaze cue.

Part II

Human Activity Tracking

5 The Time-of-Flight (TOF) Camera

The time-of-flight (TOF) camera is a combined range and image sensor that delivers both a range and an intensity measurement at every pixel in the sensor array. It works by emitting modulated light (usually infrared light) and measuring the time taken by the light to reach the object and return to the camera.

There are two main variants in which the modulation of the light and the range measurement can be performed. The first variant uses pulsed light combined with an optical or electronic shutter on the sensor [69, 138]. The closer the object is to the camera, the more light can return to the camera before the shutter closes. To account for different object reflectivities, a second measurement is made without the shutter to determine how much light the object would normally return to the camera.

The second variant, which we will describe in more detail here, uses light modulated by a periodic signal [79, 119]. The sensor, which is synchronized with the light source, integrates over different subintervals of the modulation period; from these measurements, the phase shift between the transmitted and received light (which is proportional to the distance) can be reconstructed.

Both of these variants can be viewed as specific embodiments of a more general principle: That of demodulating a periodic signal by correlating it with another periodic signal.

In both cases, the sensor has a certain *non-ambiguity range* beyond which the range can no longer be reconstructed unambiguously. This is because the light emitted by the camera is modulated by a periodic signal. When the time taken by the light to reach the object and return to the camera becomes greater than one period of the modulation signal, the measurement becomes ambiguous. A typical modulation frequency for a TOF camera is 20 MHz; this corresponds to a non-ambiguity range of 7.5 m, i.e. distances up to a maximum of 7.5 m can be measured unambiguously.

In this chapter, we will summarize the properties and working principle of the TOF camera. For an in-depth treatment of the subject, see the work of Lange [79].

5.1 Introduction

Figure 5.1 shows a few examples of the TOF cameras currently in existence. The MESA SR4000 [90] (Figure 5.1a) is a typical representative of the state of the art. It consists of an array of 24 near-infrared LEDS placed around the camera lens, which is fitted with a bandpass filter matched to the wavelength of the LEDs. Figure 5.2 shows the camera in operation; the image was taken with a camera that is sensitive in the near-infrared spectrum, and the LEDs are clearly visible around lens. The LEDs emit light with a wavelength of 850 nm, modulated with a frequency of 30 MHz; this gives the camera a non-ambiguity range of 5 metres. The total optical power emitted by the LEDs is around 1 W or below, depending on the integration time. The sensor, which is manufactured in a hybrid CCD/CMOS process, has a resolution of 176 × 144 pixels. Depending on the integration time, the camera runs at framerates of up to 54 frames per second. The accuracy of the range measurement depends on the amount of light received from the object, which in turn depends on the distance of the object, its reflectivity, and other factors; in favourable conditions, the root-mean-square (RMS) noise level in the range image is around 5 mm, and the absolute accuracy is around 1 cm.

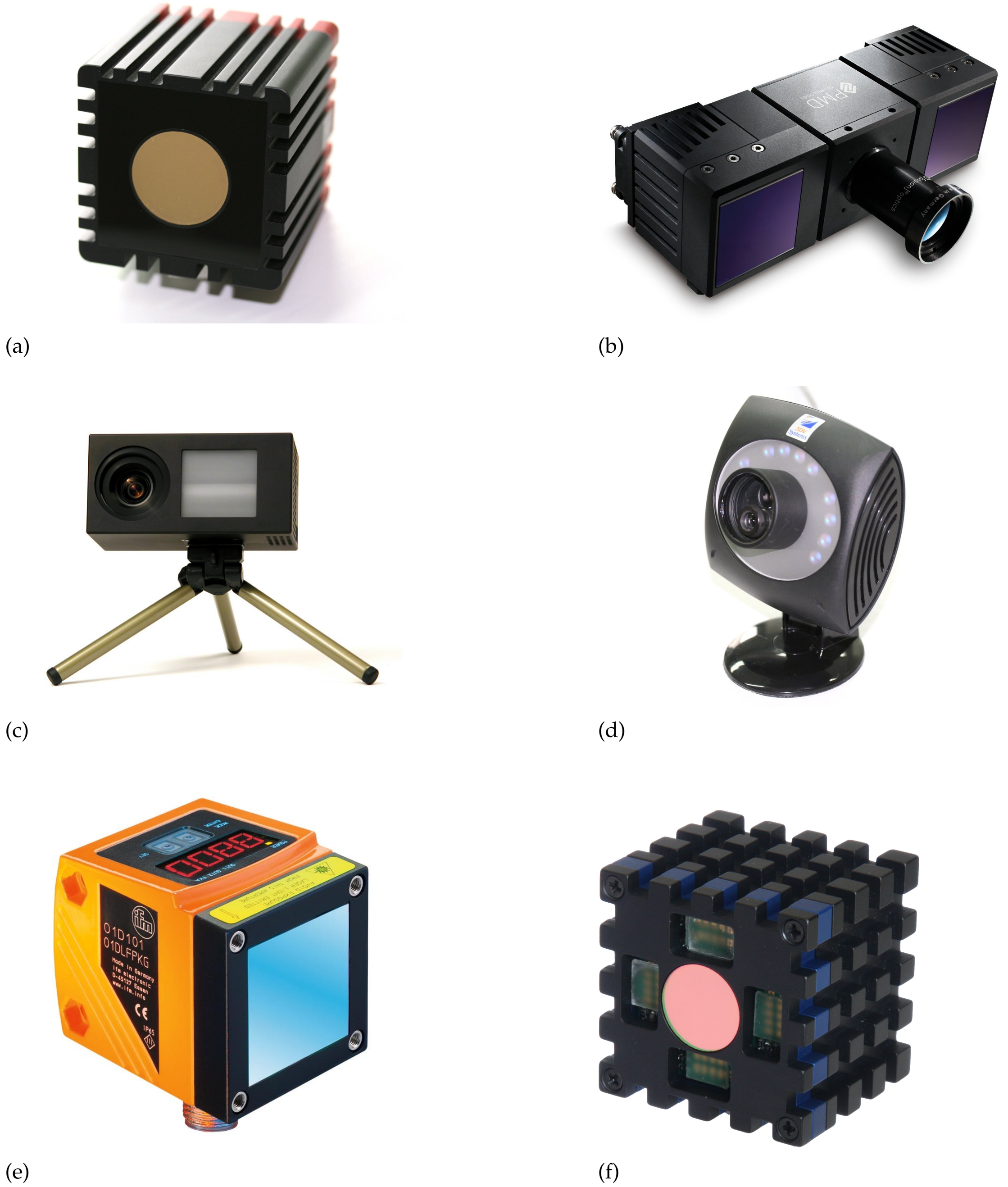
Several other manufacturers offer TOF cameras with slightly different specifications; we will not give a complete overview but cite a few notable examples. The PMDtec CamCube [109] (Figure 5.1b) has a resolution of 204 × 204 pixels; to optimize the measurement setup for different applications, the illumination units can be detached from the camera, and the lens can be exchanged. The Canesta DP200 [25] (Figure 5.1c) has a resolution of 64 × 64 pixels and comes in three variants with different fields of view (30, 55 and 80 degrees). The 3DV Systems ZCam [1] (Figure 5.1d) contains both a TOF sensor and an RGB colour sensor in a single housing, though with separate optics. It is aimed at the gaming market and is expected to sell at a consumer price point. The ifm electronic efector pmd [70] (Figure 5.1e) is a single-pixel TOF sensor for distance measurement in industrial applications; the company also offers the efector pmd3d, a 64 × 48 pixel TOF camera. A prototype camera (Figure 5.1f) developed by CSEM within the EU project ARTTS [3] is distinguished by its small size (40 × 40 × 35 mm) and the fact that it is powered entirely via a USB cable.

5.2 Comparison With Other Range Imaging Methods

Besides TOF cameras, there are a host of other range imaging technologies, with different characteristics – see [73, Volume 1, Chapter 18] and [79, Section 2.1] for an overview. Broadly, these technologies can be subdivided into the following categories:

Triangulation

The methods in this category are based on the principle that the position of a point can be determined by measuring angles towards that point at the ends of a fixed baseline. The baseline and the point to be measured form a triangle, hence the name. In all triangulation-based techniques, the accuracy of the distance measurement increases



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.1: Examples of TOF cameras: (a) MESA SR4000 (image courtesy of Ingrid Brænne), (b) PMDtec CamCube (image courtesy of PMD Technologies, Siegen, Germany), (c) Canesta DP200 development platform (image courtesy of Canesta, Inc.), (d) 3DV Systems ZCam (image courtesy of Ingrid Brænne), (e) ifm electronic efector pmd (single-pixel TOF sensor, image courtesy of ifm electronic GmbH, Essen, Germany), (f) prototype camera developed by CSEM within the ARTTS project (image courtesy of CSEM SA, Neuchâtel, Switzerland).

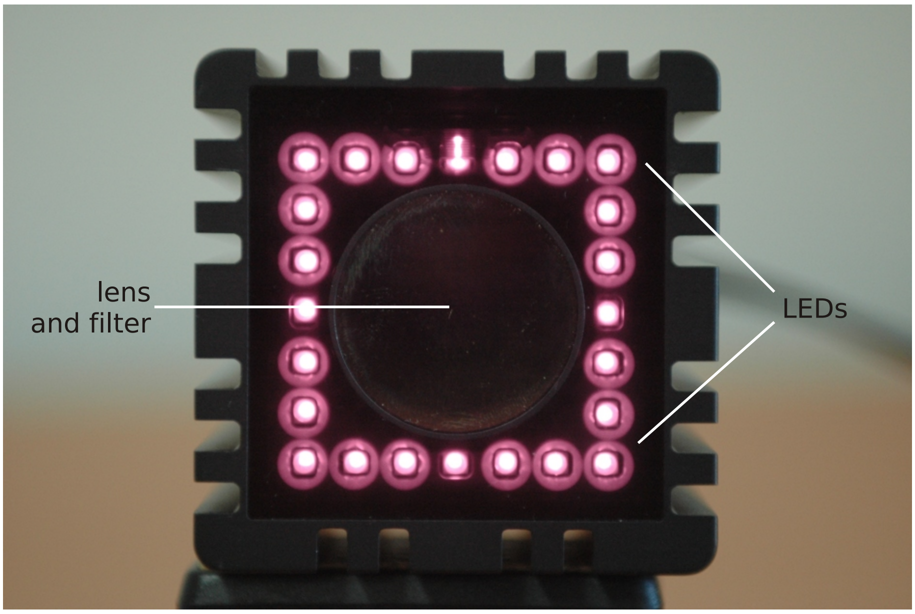


Figure 5.2: MESA SR4000 in operation. The image was taken with a camera that is sensitive in the near-infrared spectrum, making the illuminated LEDs clearly visible.

with the size of the baseline; this can be a disadvantage if it is important that the system should be small.

The basic principle of triangulation can be implemented in a variety of ways:

Stereoscopic methods use two or more cameras; the distance of a point is computed from the difference in the positions in which it appears in the camera images. Stereoscopy works best on textured objects and cannot be used on uniformly coloured surfaces.

Triangulating laser scanners sweep a collimated laser beam or a lightsheet across the scene; on the object to be measured, the beam or lightsheet appears as a dot or stripe, respectively. A camera is used to observe the position of the dot or the shape of the stripe, from which the shape of the object can be reconstructed. This technique allows accurate range measurements, but because the laser must be swept across the scene, the temporal resolution is typically lower than with many other systems. Another disadvantage is that the scanner requires moving parts.

Structured light techniques follow a similar approach. They typically use a projector to project a light pattern, such as a series of parallel stripes, onto the scene, which is observed using a camera. The shape of the object to be measured causes characteristic distortions in the appearance of the pattern, from which the shape can be reconstructed. Like triangulating laser scanners, structured light measurements can achieve high accuracy, but several different patterns are typically used per measurement, limiting the temporal resolution.

Shape from focus is a technique for reconstructing the shape of an object from the variations in its appearance when the focus setting of the lens is changed. (The baseline of the triangulation in this case is the width of the lens aperture.) Because the technique is based on the different amounts of blur caused by defocus, it does not work on uniformly coloured surfaces.

Photometric techniques

Photometric techniques exploit the fact that the intensity with which a surface point appears in the image depends on the orientation of the surface normal towards the light source, the distance between the surface and the light source, and the object's reflectivity. The shape of the object can thus be reconstructed from its appearance in the image. *Photometric stereo* uses several images of the same scene taken under different illumination conditions. It can handle objects with non-uniform reflectivity, but shadows and highlights can cause problems. *Shape from shading* is a variant of photometric stereo that uses only a single image. This obviously requires stronger constraints on the properties of the object; usually, the object is required to have uniform reflectivity.

Interferometry

Interferometry works by illuminating the object with coherent monochromatic light and superposing the reflected light with a reference wavefront from the same light source. Depending on the difference in path lengths, constructive or destructive interference will occur, and this can be used to measure the distance of the object. Interferometry allows very accurate measurements, in the range of fractions of a wavelength, but in the classical setting, absolute range measurements are not possible, and the non-ambiguity range is only half a wavelength. *Multiple-wavelength interferometry* overcomes this limitation by using light of two different but very similar wavelengths. This creates a beat signal with a much longer wavelength, allowing unambiguous range measurements up to the 10 metre range.

Time-of-flight

Like time-of-flight cameras, time-of-flight laser scanners also measure the time taken by light to travel to the object and back. However, TOF laser scanners can measure the distance of only one point at a time; to measure the entire scene, the light beam has to be swept across it, which increases the mechanical complexity and reduces the temporal resolution of the system.

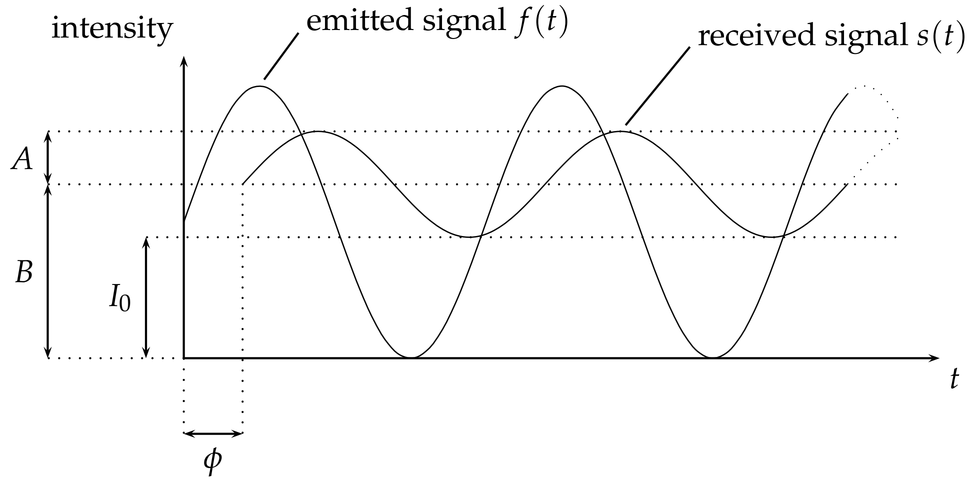


Figure 5.3: Working principle of time-of-flight cameras based on the phase-difference method. ϕ: Phase difference between the emitted and received signal. A: Amplitude of received signal. B: Offset (or DC component) of received signal. I_0: Background illumination component.

5.3 Working Principle

The majority of TOF cameras currently on the market use light whose intensity is modulated by a periodic signal. The time taken by the light to reach the object and return to the camera leads to a phase difference between the emitted and received light; by measuring this phase difference, the distance can be determined.

Figure 5.3 illustrates this working principle. The received light is delayed by a phase difference ϕ relative to the emitted light. Also, because each pixel receives only a small fraction of the light emitted by the camera, the amplitude A of the received signal is smaller than that of the emitted signal; the precise value of A depends on various factors including the object's reflectivity and its distance from the camera. Finally, because background light is usually present in the scene, the received signal contains a certain constant component I_0; the total DC component of the received signal is $B = I_0 + A$.

As shown in [79], these parameters of the received signal $s(t)$ can be reconstructed by sampling it at four points in the modulation period, yielding samples $A_0, \ldots, A_3$, with $A_i = s(\frac{i}{4} T_{\mathrm{mod}})$, where T_{mod} is the modulation period. Based on these samples, we can reconstruct the parameters of the received signal as follows:

$$\text{Phase} \quad \phi = \operatorname{atan}\left(\frac{A_0 - A_2}{A_1 - A_3}\right), \tag{5.1}$$

$$\text{Amplitude} \quad A = \frac{\sqrt{(A_0 - A_2)^2 + (A_1 - A_3)^2}}{2}, \tag{5.2}$$

$$\text{Offset} \quad B = \frac{A_0 + A_1 + A_2 + A_3}{4}. \tag{5.3}$$

For a signal with a modulation frequency of f_{mod}, a phase difference of ϕ corresponds to a time delay of $\frac{\phi}{2\pi f_{\text{mod}}}$ and thus a distance travelled by the light of $\frac{\phi\, c}{2\pi f_{\text{mod}}}$ (where c is the speed of light). Because the light has to travel to the object and back, the distance of the object from the camera is

$$R = \frac{\phi\, c}{4\pi f_{\text{mod}}}. \tag{5.4}$$

In practice, of course, it is not possible to sample the received signal at a single point in time. Rather, one must integrate the signal over a certain time window Δt; physically, this corresponds to collecting the photoelectrons generated during this time window. A natural way of doing this is to provide four "charge buckets" (i.e. potential wells) per pixel for the four measurements $A_0, \dots, A_3$; the pixel switches cyclically between these charge buckets and is synchronized to the illumination source so that one cycle through the four buckets corresponds to one modulation interval. The length of the integration window is thus $\Delta t = \frac{T_{\text{mod}}}{4}$. This type of pixel (known as a *4-tap pixel*) is a straightforward implementation of the measurement principle. However, the four charge buckets occupy a relatively large part of the pixel area, which reduces the *fill factor*, i.e. the fraction of the pixel area that is sensitive to light.

An alternative is to use only two charge buckets, each with an integration time of $\Delta t = \frac{T_{\text{mod}}}{2}$. The measurement then has to be done in two phases: In the first phase, we measure A_0 and A_2, and in the second phase, we measure A_1 and A_3. This type of pixel is known as a *2-tap pixel*.

The 2-tap pixel has a higher fill factor, but its longer integration window also brings a disadvantage: Because we are measuring the average amplitude of the signal over the whole integration window, any change that happens in the signal during the integration window is effectively lost. The longer the integration window, the more pronounced this effect is. Mathematically, we can model the effect of integrating over a time period Δt by convolving the signal with a rect function $h(t) = \frac{1}{\Delta t}\,\operatorname{rect}(\frac{t}{\Delta t})$. The corresponding transfer function $H(f) = \operatorname{sinc}(\pi f \Delta t)$ (with $\operatorname{sinc}(x) = \frac{\sin(x)}{x}$) has a low-pass characteristic. At the modulation frequency $f_{\text{mod}} = \frac{1}{T_{\text{mod}}}$, we obtain $H(f_{\text{mod}}) = \operatorname{sinc}(\pi \frac{\Delta t}{T_{\text{mod}}})$. For $\Delta t < T_{\text{mod}}$, this transfer function leaves the phase of the signal unchanged, so it does not interfere with the phase measurement, but it attenuates the signal by a certain factor, and this attenuation increases with the length of the integration window Δt. The effect of this is that the measured amplitude A is smaller than the actual amplitude of the signal A_{sig} that reaches the pixel. The ratio between the measured amplitude and the actual amplitude is called the *demodulation contrast*, defined as

$$c_{\text{demod}} = \frac{A}{A_{\text{sig}}}. \tag{5.5}$$

Integrating over a time window of Δt leads to a demodulation contrast of $c_{\text{demod}} = \operatorname{sinc}(\pi \frac{\Delta t}{T_{\text{mod}}})$, as derived above. (In the physical implementation, other effects reduce the demodulation contrast further, but we will not deal with these here.) For a 2-tap pixel, with $\Delta t = \frac{T_{\text{mod}}}{2}$, we obtain a demodulation contrast of $c_{\text{demod}} = \frac{2}{\pi} \approx 0.64$; for a 4-tap pixel, with $\Delta t = \frac{T_{\text{mod}}}{4}$, we obtain $c_{\text{demod}} = \frac{\sqrt{8}}{\pi} \approx 0.90$. From this point of view, the 4-tap pixel appears to be superior, but the higer fill factor of the 2-tap pixel more than makes up for the difference in demodulation contrast, and so the majority of TOF cameras on the market use 2-tap pixels.

An actual TOF camera will only be able to implement an approximation to the measurement principle presented here: The signal emitted by the active illumination will not be exactly sinusoidal, the timing of the integration intervals will not be perfect, and so on. These deviations from the idealized measurement principle will cause a systematic error; due to manufacturing variations, this error is different for every pixel. A wide variety of calibration schemes that can eliminate most types of systematic error have been devised. We will not describe these calibration schemes in detail here but refer simply to the survey given in [77].

5.4 Accuracy

How accurate the TOF range measurement is depends on the signal-to-noise ratio of the measurements $A_0, \ldots, A_3$. An important factor that influences the signal-to-noise ratio is the amount of light: The more light, the better the signal-to-noise ratio. One way of improving the accuracy is thus to increase the strength of the illumination, but this is not always possible, for reasons of both power consumption and eye safety.

For this reason, we will examine the factors that govern measurement accuracy given a certain amount of available light. There are a number of effects that contribute to the noise present in the measurements $A_0, \ldots, A_3$, but while most of these can, at least in theory, be reduced by improving the technology, one noise source, photon shot noise, is a principal physical limitation. Photon shot noise occurs because a light source does not emit a steady stream of photons at regular intervals; rather, photons are emitted probabilistically at irregular intervals according to a Poisson distribution. Because the standard deviation of a Poisson-distributed random variable is the square root of the mean, the standard deviations of the measurements $A_0, \ldots, A_3$ are $\sqrt{A_0}, \ldots, \sqrt{A_3}$. From this, one can show [79] that the standard deviation of the phase measurement is

$$\Delta\phi = \frac{\sqrt{B}}{A\sqrt{2}} \tag{5.6}$$

and, hence, the standard deviation of the range measurement is

$$\Delta R = \frac{c}{4\pi f_{\text{mod}}} \cdot \frac{\sqrt{B}}{c_{\text{demod}} A_{\text{sig}} \sqrt{2}}. \tag{5.7}$$

From this, we see that several factors influence measurement accuracy. First, increasing the amplitude of the modulated light or increasing the demodulation contrast decreases the measurement error. Second, increasing the amount of background light (which in turn increases B, the mean amount of light reaching the pixel) increases the measurement error. Finally, increasing the modulation frequency decreases the measurement error.

Equation 5.7 is a principal physical limitation on the measurement accuracy; the TOF cameras currently on the market are already close to reaching this physical limit [24].

5.5 Limitations

Like any measurement technology, TOF cameras have a number of idiosyncrasies and limitations that need to be kept in mind when using them in applications.

Multiple reflections The TOF measurement principle is based on the assumption that all of the modulated light arriving at a pixel comes from the same object and that it travelled directly from the camera to the object and back. There are two basic ways in which this assumption can be violated: First, the object being imaged may be illuminated not only directly but also indirectly through light from the active illumination that is reflected by other objects in the scene (which need not even be in the field of view of the camera); this indirect illumination has taken a longer path, leading to a longer time of flight and, hence, an incorrect range measurement. Second, unwanted reflections inside the lens and camera body can cause stray light from other objects to reach a pixel in addition to the light that was intended for that pixel; again, this causes an error in the measured distance. While the latter effect (multiple reflections inside the camera) can be minimized using suitable nonreflective coatings, the phenomenon of multiple reflections outside the camera is a principal problem for current time-of-flight cameras. If possible, the scene to be measured should be arranged in a way that minimizes indirect reflections.

Flying pixels Pixels that straddle the border between two objects at different ranges will receive light from both objects. This results in a measured range that lies somewhere between the ranges of the two objects, depending on the relative amount of light contributed by them. When the objects move, this relative contribution changes, causing the pixel to appear to fly towards or away from the camera; this effect is therefore referred to as the *flying pixel* effect.

Motion artefacts In a 1-tap or 2-tap pixel, which are used in the majority of TOF cameras currently available, the four measurements $A_0, \ldots, A_3$ (see Section 5.3) are not obtained simultaneously; a 2-tap pixel, for example, may first measure A_0 and A_2 and then A_1 and A_3. If the object moves between these measurements, they will not be consistent, and the result will be an error in the range measurement. Note that the effect of this is not just an averaging of the range value over time; the range value that is computed can be greater than the largest range value or less than the smallest range value seen by the pixel in the given time interval.

Interference between multiple cameras If multiple TOF cameras are used in close proximity, the modulated light signals emitted by them will interfere with each other. One way of dealing with this is to operate the cameras at different modulation frequencies. The demodulation technique used for the phase measurement responds only to a narrow band of frequencies around the modulation frequency, and the light emitted by the other cameras at different modulation frequencies will only show up as an increased background illumination component. An alternative approach to using several cameras in the same environment is to modulate the light with a pseudo-noise signal that is different for each camera [63].

6 Image Improvement Using the Shading Constraint

6.1 Introduction

Whereas the strength of the TOF camera is that it delivers both a range map and an intensity image with high temporal resolution in a compact solid-state device, the signal-to-noise ratio of the range map is not as good as with some other ranging technologies. In this chapter, we will explore a technique for increasing this signal-to-noise ratio, based on the insight that the range and intensity measurements are not independent but are linked by the *shading constraint*: Assuming that the reflectance properties of the object surface as well as the positions and illumination properties of the light sources are known, we can deduce the intensity image that should be observed for a given range map. In practice, a general reflectance model (such as Lambertian reflectance) will provide an acceptable approximation to the properties of a wide range of objects.

In theory, the shading constraint can be used to reconstruct the range map from an intensity image alone; this idea has been exploited in a wide range of *shape from shading* (SfS) algorithms (see [38, 143] for surveys). A principal limitation of these algorithms, however, is that they cannot determine whether intensity changes are caused by the object's shape or by changes in the object's reflectivity (or *albedo*). Because of this, the object is usually assumed to have constant albedo; this limits the applicability of SfS methods.

The range map measured by the TOF camera places a strong additional constraint on the shape of the object, allowing ambiguities that may exist in the pure SfS setting [39] to be resolved and enabling the albedo of the surface to be estimated, both globally for an entire object as well as locally for objects whose albedo varies across the surface.

Besides the shading constraint, there are also other ways of fusing range and intensity data. A number of authors exploit the fact that an edge in the intensity data often co-occurs with an edge in the range data. Nadabar and Jain [98] use a Markov random field (MRF) to identify different types of edges. Diebel and Thrun [33] use edge

Parts of this chapter are joint work with others. I developed the concept of enforcing the shading constraint in a probabilistic framework; Martin Haker and I contributed approximately equally to refining and implementing that concept. The work described here has previously been published in [17, 18].

strengths estimated on a high-resolution colour image to increase the resolution of a low-resolution depth map.

The idea of integrating the shading constraint with other range information has been investigated by a number of researchers. Most of this work focuses on the integration of SfS with stereo. Many approaches to this problem (see for example the work of Thompson [126], Fua and Leclerc [47], and Hartt and Carlotto [61]) use an objective function that depends directly on the two or more images obtained from a multi-camera setup; for this reason, they do not generalize to settings where a range map has been obtained in some other way than through stereo. Samaras et al. [116] combine stereo with SfS by using stereo in textured areas and SfS in untextured areas, but they do not perform a fusion of stereo and SfS at the same location. Haines and Wilson [51, 52] fuse stereo and SfS in a probabilistic approach based on a disparity map and the shading observed in one of the stereo images. Because there is a one-to-one correspondence between disparity and range, the approach could also be used with range maps obtained by arbitrary means. However, since colour is used to segment areas of different albedo, the approach is not suitable for use with TOF cameras, which typically only deliver a grayscale image.

There are several other approaches that combine shading with a range map obtained by arbitrary means; stereo may be used, but it is not essential to the formulation of the algorithm. Leclerc and Bobick [80] use a stereo range map to initialize an iterative SfS method. Cryer et al. [31] use a heuristic that combines low-frequency components from the stereo range map with high-frequency components from the SfS range map. Mostafa et al. [96] use a neural network to interpolate the difference between the SfS result and a more coarsely sampled range map from a range sensor; the SfS result is corrected using this error estimate. These approaches allow range maps from arbitrary sources to be used, but they are all somewhat ad-hoc.

Our approach to improving the accuracy of the range map using the shading constraint is based on a probabilistic model of the image formation process. We obtain a maximum a posteriori estimate for the range map using a numerical minimization technique. The approach incorporates the range map measured by the TOF camera and the information from the shading constraint in a single theoretically well-founded framework. As we will see, the signal-to-noise ratio as well as the subjective quality of the range map is improved substantially by the algorithm.

The main shortcoming of the current approach is that it does not deal well with discontinuities in the range map (so-called jump edges) and that it takes several minutes to process a range map, making it unsuitable for interactive applications. In the discussion (Section 6.4), we will address potential ways to deal with these shortcomings.

6.2 Method

6.2.1 Probabilistic Image Formation Model

Our approach to image improvement is based on the following model for the probability $P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}})$ of observing a range map $\boldsymbol{X}^{\mathrm{R}}$ and an intensity image $\boldsymbol{X}^{\mathrm{I}}$:

$$P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}}) = P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A})\ P(\boldsymbol{R})\ P(\boldsymbol{A}). \tag{6.1}$$

$P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A})$ is the conditional probability of observing a range map $\boldsymbol{X}^{\mathrm{R}}$ and an intensity image $\boldsymbol{X}^{\mathrm{I}}$ given that the true range map describing the shape of the imaged object is $\boldsymbol{R}$ and that the parameters of the reflectance model are $\boldsymbol{A}$. (Typically, $\boldsymbol{A}$ is the albedo of the object – we will discuss this in more detail below.) $P(\boldsymbol{R})$ is a prior on the range map, $P(\boldsymbol{A})$ is a prior on the reflectance model parameters. We then seek to find the range map $\boldsymbol{R}$ and parameters $\boldsymbol{A}$ that maximize Equation 6.1 to obtain a maximum a posteriori estimate.

The conditional probability $P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A})$ is based on the following model of image formation: First of all, we assume that $P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A})$ can be written as follows:

$$P(\boldsymbol{X}^{\mathrm{R}}, \boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A}) = P(\boldsymbol{X}^{\mathrm{R}} | \boldsymbol{R}, \boldsymbol{A})\ P(\boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A}). \tag{6.2}$$

In other words, the observations $\boldsymbol{X}^{\mathrm{R}}$ and $\boldsymbol{X}^{\mathrm{I}}$ are conditionally independent given $\boldsymbol{R}$ and $\boldsymbol{A}$.

We now assume that the observed range map $\boldsymbol{X}^{\mathrm{R}}$ is simply the true range map $\boldsymbol{R}$ with additive Gaussian noise, i.e.

$$P(\boldsymbol{X}^{\mathrm{R}} | \boldsymbol{R}, \boldsymbol{A}) = \mathcal{N}(\boldsymbol{X}^{\mathrm{R}} - \boldsymbol{R} | \mu = 0, \sigma_{\mathrm{R}}(\boldsymbol{R}, \boldsymbol{A})). \tag{6.3}$$

Note that the standard deviation σ_{R} is not constant but can vary per pixel as a function of range and albedo. As we saw in Section 5.4, the noise in the range measurement of a TOF camera depends on the amount of light that returns to the camera; we will discuss how we set σ_{R} in Section 6.2.4.

The observed intensity image depends on the range map through the reflectance model, which we express as a function $\boldsymbol{I}(\boldsymbol{R}, \boldsymbol{A})$ that takes the range map $\boldsymbol{R}$ and parameters $\boldsymbol{A}$ and yields the corresponding image. For the results presented in this chapter, we will use the Lambertian reflectance model (see Section 6.2.2); in this case, the parameter vector $\boldsymbol{A}$ contains the albedo of the object, which may vary from pixel to pixel. Again, the observed intensity image is corrupted by Gaussian noise, i.e.

$$P(\boldsymbol{X}^{\mathrm{I}} | \boldsymbol{R}, \boldsymbol{A}) = \mathcal{N}(\boldsymbol{X}^{\mathrm{I}} - \boldsymbol{I}(\boldsymbol{R}, \boldsymbol{A}) | \mu = 0, \sigma_{\mathrm{I}}). \tag{6.4}$$

For the range map prior $P(\boldsymbol{R})$, we use the shape prior introduced by Diebel et al. [34], which favours surfaces with smoothly changing surface normals. We tessellate the range map into triangles (see Section 6.2.3) and compute the surface normal $\boldsymbol{n}_j$ for each triangle. The shape prior is then given by the energy function

$$E^{\mathrm{R}}(\boldsymbol{R}) = w_{\mathrm{R}} \sum_{\substack{\text{triangles } j,k \\ \text{adjacent}}} \|\boldsymbol{n}_j - \boldsymbol{n}_k\|_2, \tag{6.5}$$

which implies the distribution $P(\boldsymbol{R}) = \frac{1}{Z}\exp(-E^{\mathrm{R}}(\boldsymbol{R}))$, where Z is a normalization constant. w_{R} is a constant that controls the dispersion of the distribution.

Finally, we need to define the prior $P(\boldsymbol{A})$ for the parameters $\boldsymbol{A}$ of the reflectance model. In the Lambertian reflectance model, these are the albedo values at each pixel location. We will investigate several alternatives for the prior $P(\boldsymbol{A})$: (i) "Fixed albedo": A single albedo value, specified beforehand, is used for all pixels. (ii) "Global albedo": The same global albedo is used for all pixels, but its value is allowed to vary; we assume a uniform distribution for this global albedo. (iii) "Local albedo": Each pixel location may have a different albedo, and the prior $P(\boldsymbol{A})$ favours smooth albedo changes. In this latter case, we use an energy function

$$E^{\mathrm{A}}(\boldsymbol{A}) = w_{\mathrm{A}} \sum_{\substack{\text{pixels } j,k \\ \text{adjacent}}} |a_j - a_k|, \tag{6.6}$$

which implies the prior $P(\boldsymbol{A}) = \frac{1}{Z}\exp(-E^{\mathrm{A}}(\boldsymbol{A}))$, in analogy to the shape prior defined above.

As usual, we take the negative logarithm of the posterior and eliminate constant additive terms to obtain an energy function

$$\begin{aligned} E(\boldsymbol{R}, \boldsymbol{A}) = & \sum_j \frac{(X_j^{\mathrm{R}} - R_j)^2}{2\,\sigma_{\mathrm{R}}^2} + \\ & \sum_j \frac{(X_j^{\mathrm{I}} - I_j(\boldsymbol{R}, \boldsymbol{A}))^2}{2\,\sigma_{\mathrm{I}}^2} + \\ & E^{\mathrm{R}}(\boldsymbol{R}) + \\ & E^{\mathrm{A}}(\boldsymbol{A}), \end{aligned} \tag{6.7}$$

where the index j runs over all pixels. (For the "fixed albedo" and "global albedo" models, the term $E^{\mathrm{A}}(\boldsymbol{A})$ is omitted.) Note that all the terms in the energy function are unitless due to multiplication or division by the constants σ_{R}, σ_{I}, w_{R} and w_{A}.

We minimize $E(\boldsymbol{R}, \boldsymbol{A})$ using the Polak-Ribière variant of the nonlinear conjugate gradient algorithm (see e.g. [111]). As the starting point for the minimization, we use the observed range map $\boldsymbol{X}^{\mathrm{R}}$, smoothed using a median filter, and an albedo guess (see Section 6.2.4). The gradient of $E(\boldsymbol{R}, \boldsymbol{A})$ is computed numerically using a finite differences approximation. The parameters σ_{R}, σ_{I}, w_{R} and w_{A} should be set to reflect the noise characteristics of the sensor and the statistical properties of the scene.

6.2.2 Lambertian Reflectance Model

For the results presented in this chapter, we will use the Lambertian model of diffuse reflection [129], and we will see that we obtain satisfactory results even on surfaces, such as skin, that do not fulfill the assumption of Lambertian reflectance exactly. Highly glossy surfaces, however, will require a different reflectance model to achieve satisfactory results.

Under the Lambertian reflectance model, the intensity I of a pixel in the image is

$$I = a\frac{\boldsymbol{n} \cdot \boldsymbol{l}}{r^2}, \tag{6.8}$$

where $\boldsymbol{n}$ is the surface normal, $\boldsymbol{l}$ is a unit vector that points from the surface point towards the light source, r is the distance of the light source from the surface point, and a is a constant that describes the combined effect of the albedo of the surface, the intensity of the light source, and properties of the camera such as aperture and exposure time. We will refer to a simply as the albedo because albedo is the only factor affecting a that is not constant across the scene.

In the case of images captured with a TOF camera, the reflectance model is particularly easy to evaluate because the light source is colocated with the camera. Hence, r is just the distance of the surface point from the camera (i.e. its range value), and l is the unit vector from the surface point to the camera.

6.2.3 Discretization

The pixel array of the TOF camera implies a discretization of the range map and intensity image. Though each pixel measures the average range and intensity over a certain area, we will treat the measured values as if they were samples taken at point locations on the object surface.

Particular care needs to be taken when computing the surface normals that are used by the reflectance model and the shape prior. An obvious way of computing the normals (and one that we tried initially) is to take the cross product of two tangent vectors $\boldsymbol{p}(i+1,j) - \boldsymbol{p}(i-1,j)$ and $\boldsymbol{p}(i,j+1) - \boldsymbol{p}(i,j-1)$ (where $\boldsymbol{p}(i,j)$ is the spatial position of the surface point corresponding to pixel (i,j)). However, this approach can cause undesirable solutions to be computed. For example, consider a "corrugated" surface where $\boldsymbol{p}(i,j) = (i,j,0)$ for odd i and $\boldsymbol{p}(i,j) = (i,j,1)$ for even i. If the normals are computed as described above, every pixel will have the same normal, and the surface will appear to be completely flat. As a consequence, the shading of the surface will be wrong, and the shape prior will fail to penalize the surface even though it is in fact highly uneven. We have observed in practice that this effect can cause folds or "checkerboard" patterns to appear in the surface reconstruction.

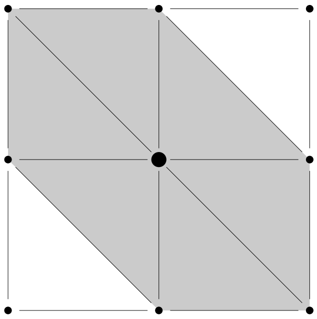


Figure 6.1: The intensity of a pixel is computed by averaging over the intensity of all triangles that are adjacent to it. Triangles that are adjacent to the central pixel are shaded in gray.

For this reason, we use a different approach where the range map is tessellated into an array of triangles; we then compute surface normals not per pixel but per triangle. This ensures that the normals that are computed correspond to a physically consistent surface. Of the many different possible triangular tessellations, we choose one where all diagonals between two pixels run in the same direction.

The shading of the surface is obtained by first shading every triangle, then averaging over all triangles that are adjacent to a pixel to obtain the shading for that pixel. (All triangles have the same projected area in the image, hence they are all weighted equally. In addition, because the triangles are small relative to their distance from the camera, we assume that the shading is constant across each triangle.) The intensity I_j of pixel j is then

$$I_j = \frac{a_j \sum_{k \in N_j} \boldsymbol{n}_k \cdot \boldsymbol{l}_j}{R_j^2 \, |N_j|}, \tag{6.9}$$

where a_j is the albedo of the pixel, R_j is the range value of the pixel, $\boldsymbol{l}_j$ is the unit vector from the surface point to the light source, $\boldsymbol{n}_k$ is the surface normal of triangle k, and N_j is the set of triangles that are adjacent to pixel j; see Figure 6.1 for an illustration of the triangles that are adjacent to a pixel.

The shape prior is evaluated at every edge between two triangles. The direction in which the diagonals run introduces an asymmetry into the tessellation and hence into the shape prior, and we have found in testing that this asymmetry can cause directional artefacts. To avoid these artefacts, we evaluate the shape prior for both possible directions of the diagonals and take the average of the two results.

To make sure that we evaluate the shape prior exactly once for each edge in the tessellation, we iterate over all pixels and evaluate the shape prior for the edges that lie below and to the right or below and to the left of the pixel, depending on the direction of the diagonal; see Figure 6.2 for an illustration of this.

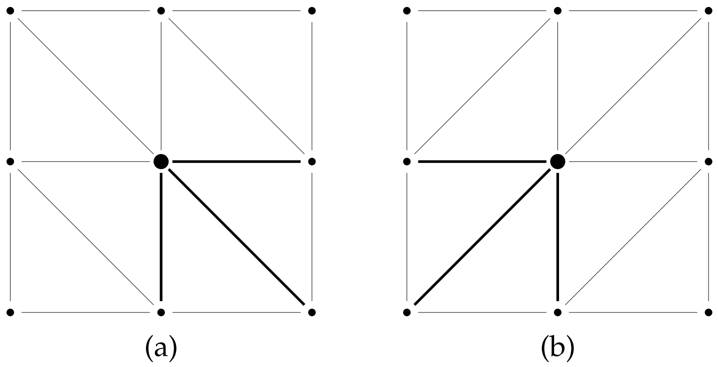


Figure 6.2: To avoid directional artefacts, the shape prior is evaluated over two different tessellations (a) and (b), with the diagonals running in opposite directions. For each pixel, the shape prior is evaluated for the edges shown in bold.

6.2.4 Implementation considerations

In a TOF camera, the standard deviation of the range measurement σ_R is not constant but depends on the amount of light arriving at each pixel: The more light, the lower σ_R. For this reason, we set the value of σ_R per pixel based on the intensity at that pixel. To determine the functional relationship between intensity and σ_R, we recorded a sequence of images of a static scene, from which we were able to estimate the standard deviation of the range measurements at each pixel. We then fit a power law regression function to the measured data. We also used the recorded image sequence to estimate the standard deviation of intensity σ_I. Since σ_I is approximately constant for all pixels (for a given integration time and illumination intensity), we estimated a single fixed value for σ_I.

Another implementation issue is that a TOF camera's illumination is typically stronger in the centre of the field of view and falls off towards the periphery. To measure this effect, we recorded an image (averaged over 100 frames) of a planar wall with constant reflectivity and fit a plane to the measured points. We then used the intensity image of this plane as computed by the reflectance model (which assumes homogeneous illumination) and the measured intensity image to compute a correction factor for each pixel. We can use this correction factor to compensate for the illumination inhomogeneity in new images as follows:

$$\boldsymbol{X}^\mathrm{I}_\mathrm{corrected}(i,j) = \boldsymbol{X}^\mathrm{I}(i,j) \cdot \frac{\boldsymbol{X}^\mathrm{I}_\mathrm{p}(i,j)}{\boldsymbol{X}^\mathrm{I}_\mathrm{a}(i,j)}, \tag{6.10}$$

where $\boldsymbol{X}^\mathrm{I}_\mathrm{p}$ is the intensity image of the plane predicted by the reflectance model, $\boldsymbol{X}^\mathrm{I}_\mathrm{a}$ is the actual intensity image of the plane, $\boldsymbol{X}^\mathrm{I}$ is a new image, and $\boldsymbol{X}^\mathrm{I}_\mathrm{corrected}$ is the corrected version of that image.

Finally, we need to find a suitable starting point for the minimization. To obtain the initial range map, we apply a 5 × 5 median filter to the measured range map. To obtain an initial albedo estimate, we observe that it is likely that the brightest pixel in the image corresponds to a part of the surface that is oriented face-on to the light source. In this case, the surface normal $\boldsymbol{n}$ and the vector to the light source $\boldsymbol{l}$ in the reflectance model (Equation 6.8) are identical, and the equation reduces to $I = \frac{a}{r^2}$. We can rearrange for a and insert the measured intensity I and range value r into the equation to obtain the albedo estimate.

6.3 Results

6.3.1 Synthetic Data

To evaluate the algorithm on data with known ground truth, we used synthesized range maps and intensity images of two test objects (see the leftmost column of Figure 6.3): "wave" (a rotationally symmetric sinusoid) and "corner" (two planar regions meeting at a sharp edge).

We obtained simulated intensity images for these objects by using the reflectance model to shade the ground truth range map, then adding Gaussian noise to the generated intensity image. The measured range map was simulated by adding Gaussian noise to the ground truth range map. To ease evaluation, the standard deviation of the range noise was the same for all pixels, i.e. it did not depend on intensity.

The parameters σ_I and σ_R of the probabilistic model were set to the actual standard deviations of the noise that was added to the intensity image and the range map. The parameters w_R and w_A were set to $w_R = 1$ and $w_A = 50$ for all tests.

Figure 6.3 shows the results obtained on the synthetic test objects shaded with a constant albedo value of 0.2 for all pixels. Since albedo was the same for all pixels, we used the "global albedo" variant of the algorithm; the initial albedo for the minimization was set to 0.4. The range noise had a standard deviation of 20 mm; for comparison, the "wave" object has a depth of 100 mm, and the "corner" object has a depth of 120 mm. The intensity noise had a standard deviation of 0.003; this compares to a maximum intensity of 0.22 in the "wave" images and 0.19 in the "corner" image.

On these test data, the algorithm reduces the RMS error by a factor of over 4 for the "wave" object and around 8 for the "corner" object. To compare these results wtih that of a standard denoising filter, we also applied a 5 × 5 median filter to the range map (fourth column in Figure 6.3). The RMS error in the median-filtered range maps is higher by more than a factor of 2, and the median-filtered result is visually less pleasing than that of the "global albedo" algorithm.

We will now evaluate the algorithm on the same objects but with varying albedo. Accordingly, we will use the "local albedo" algortihm, but will also run the "global

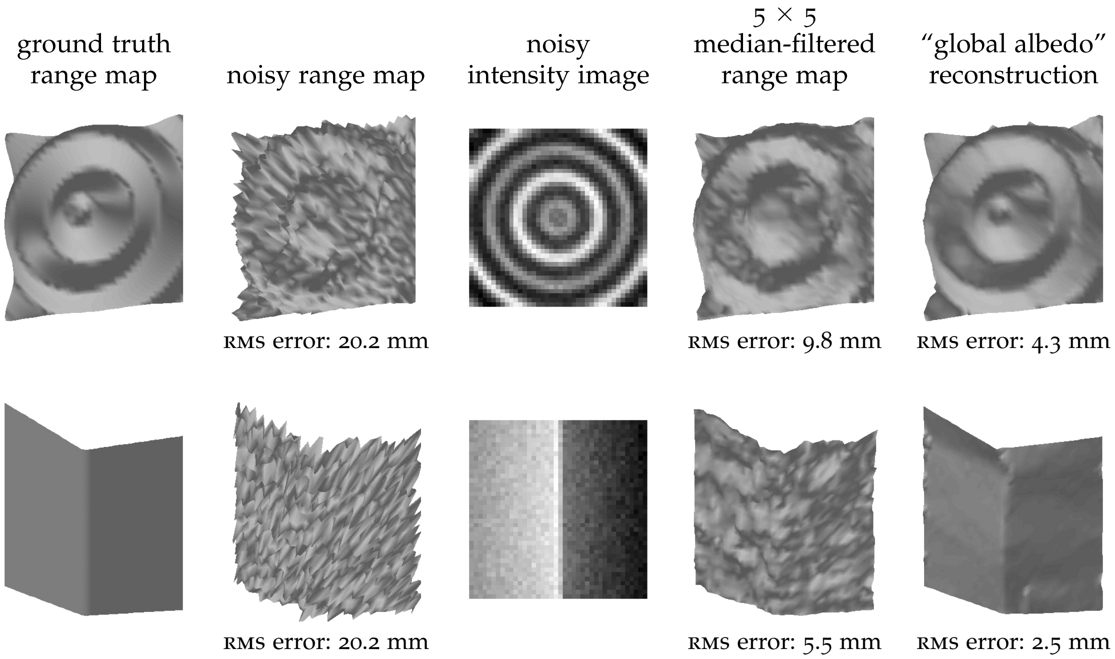


Figure 6.3: Reconstruction results for two synthetic test objects ("wave", top, and "corner", bottom) using the "global albedo" algorithm. The Gaussian noise in the range map had a standard deviation of 20 mm; for comparison, the "wave" object has a depth of 100 mm, and the "corner" object has a depth of 120 mm. The Gaussian noise in the intensity image had a standard deviation of 0.003; the maximum intensity is 0.22 in the "wave" image and 0.19 in the "corner" image.

albedo" variant for comparison. Because this setting is more challenging, we reduce the range noise to a standard deviation of 5 mm; all other parameters remain as before. Figure 6.4 (top) shows the results on the "wave" object with albedo set to 0.2 on the left half of the object and 0.4 on the right half. (The initial albedo for the minimization was set to 0.3.) Not surprisingly, the "global albedo" algorithm fails to produce a satisfactory result; in constrast, the "local albedo" algorithm reconstructs the surface well, and the estimated albedo matches the ground truth almost perfectly. The middle row in Figure 6.4 shows a test with the same object but with albedo varying continuously from 0.2 at the left to 0.4 at the right. Albedo is overestimated slightly on the left side of the image, and the result is not quite as good as in the first case but still satisfactory. Finally, we show a case where the albedo estimation does not work properly (Figure 6.4, bottom): the "corner" object with albedo varying continuously between 0.2 at the top and 0.4 at the bottom. Here, the result of the "local albedo" algorithm is not satisfactory and, in fact, its RMS error is higher than that of the "global albedo" algorithm. We suspect the reason for the poor performance may be that the range map does not contain enough detail for the algorithm to "latch onto".

Finally, we will evaluate the effect of varying amounts of noise σ_{I} in the intensity image and the contribution that the individual components of the probabilistic model make to the result. We compare the results of using only the shading constraint $P(\boldsymbol{X}^{\mathrm{I}}|\boldsymbol{R},\boldsymbol{A})$, only the shape prior $P(\boldsymbol{R})$, or both. The term $P(\boldsymbol{X}^{\mathrm{R}}|\boldsymbol{R},\boldsymbol{A})$, which constrains the solution to lie near the measured range map, will be used in all cases, and because we will use an object with constant albedo, the term $P(\boldsymbol{A})$ will be omitted.

We need to take special consideration to test the case $\sigma_{\mathrm{I}} = 0$ (no intensity noise), since σ_{I} appears in the denominator of the shading term in the energy function (Equation 6.7). Note that for $\sigma_{\mathrm{I}} \to 0$, the shading term dominates all the other terms; hence, if $\sigma_{\mathrm{I}} = 0$, we omit all other terms from the energy function. We can then avoid scaling by $\frac{1}{{\sigma_{\mathrm{I}}}^2}$.

Figure 6.5 shows the results of the tests. The error for the reconstruction obtained using only the shape prior (along with the measured range map $P(\boldsymbol{X}^{\mathrm{R}}|\boldsymbol{R},\boldsymbol{A})$) is, of course, constant for all σ_{I} because it does not use the intensity image. The shape prior reduces the RMS error in the range map by around a factor of 2.

For the reconstructions obtained using only the shading constraint (along with the measured range map), the error in the reconstruction generally increases with the noise σ_{I} in the intensity image. However, it is notable that, even for $\sigma_{\mathrm{I}} = 0$, the range map is not reconstructed perfectly. Though, in this case, the ground truth range map is obviously the minimum of the energy function, the algorithm appears to get stuck in a local minimum. Recall that in the case $\sigma_{\mathrm{I}} = 0$, the shading term dominates all other terms in the energy function. As σ_{I} increases, the energy function begins taking the measured range map into account, and this in fact leads to an initial slight reduction

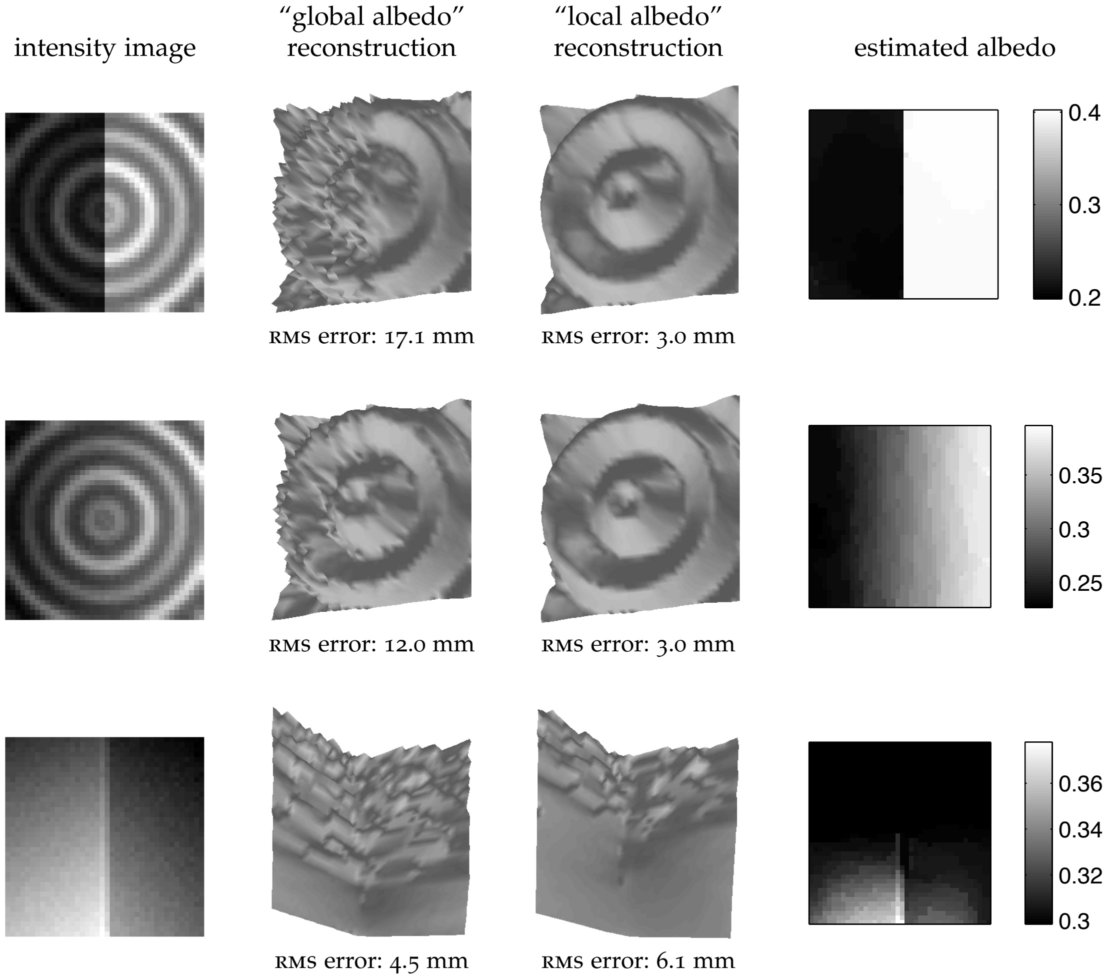


Figure 6.4: Reconstruction results on synthetic objects with varying albedo. Top: "Wave" object with an albedo of 0.2 on the left half of the object and 0.4 on the right half. Middle: "Wave" object with albedo varying continuously from 0.2 at the left to 0.4 at the right. Bottom: "Corner" object with albedo varying continuously from 0.2 at the top to 0.4 at the bottom. In all cases, the noise in the range map had a standard deviation of 5 mm, and the noise in the intensity image had a standard deviation of 0.003.

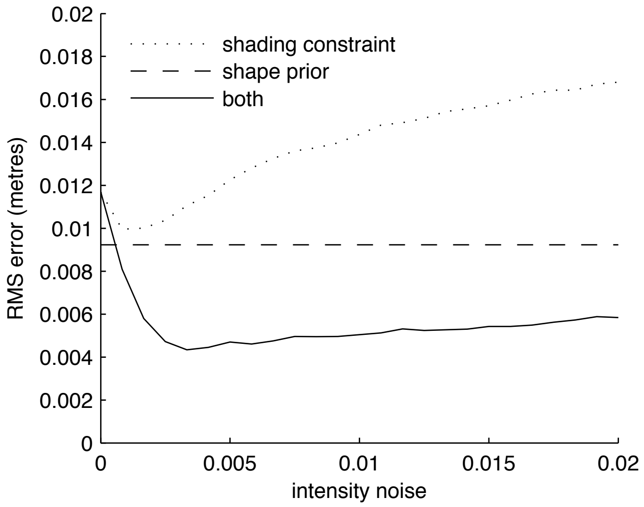


Figure 6.5: Reconstruction error on the "wave" object as a function of noise in the intensity image, for different probabilistic models. (For comparison, the maximum intensity in the image was 0.22.) Range noise was fixed at a standard deviation of 20 mm.

in the reconstruction error; we speculate that the additional constraint imposed by the measured range map makes it easier to minimize the energy function.

This effect is even more prononunced for the full model, which combines the shading constraint, the shape prior, and the measured range map. For $\sigma_I = 0$, the shading term again dominates all other terms in the energy function, and so we obtain the same result as for the shading constraint alone. As σ_I begins to increase, the reconstruction error decreases markedly as the shape prior and the measured range map come into play. After a certain point, the reconstruction error begins increasing again; for $\sigma_I \to \infty$, the reconstruction error will tend to that of the shape prior because the shading term in Equation 6.7 tends to zero. Note that, except for very small σ_I, the combined model yields better results than either the shading constraint or the shape prior alone.

6.3.2 Real-World Data

We will now test the algorithm on real images recorded using a TOF camera, the SwissRanger SR3000 [100], which has a resolution of 176 × 144 pixels. Because the algorithm in its present form cannot deal well with range discontinuities, we manually segmented the objects from the background and computed the reconstruction only on the object. The range and intensity noise were estimated as described in Section 6.2.4; the parameters w_R and w_A were again set to $w_R = 1$ and $w_A = 50$.

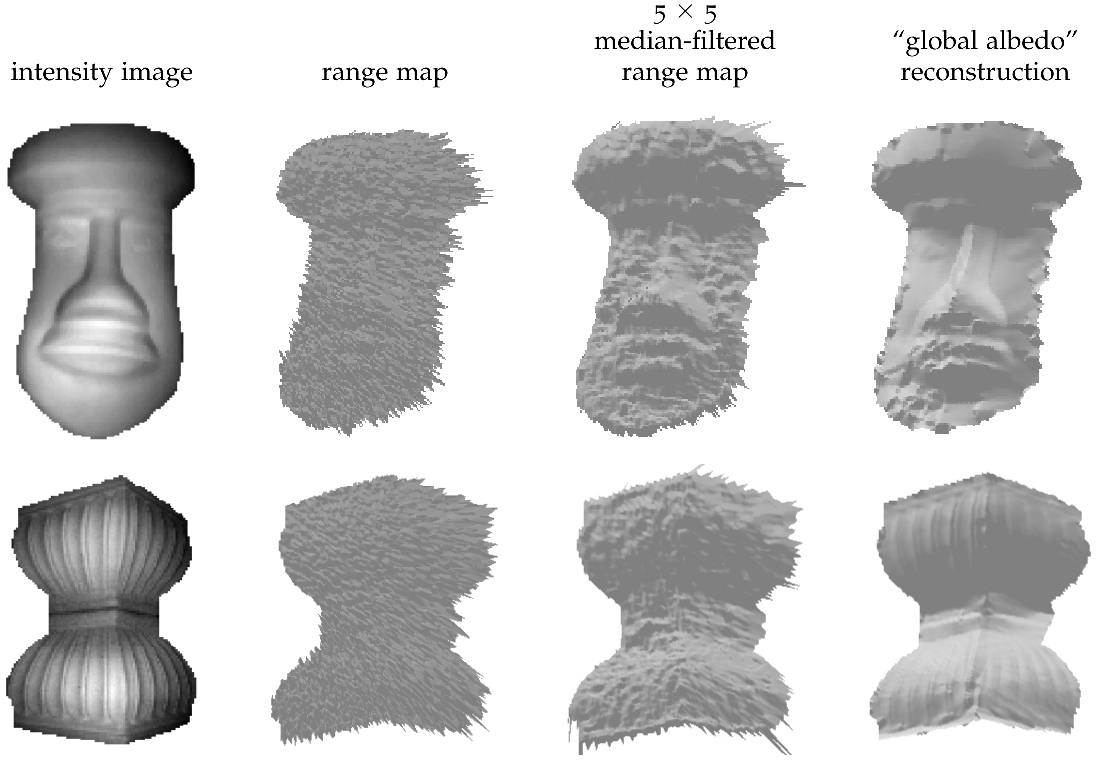

intensity image
range map
5 × 5 median-filtered range map
"global albedo" reconstruction

Figure 6.6: Surface reconstructions of terracotta objects. The renderings of the range maps are rotated 30 degrees around the vertical axis.

For a first test, we will use two objects made of unglazed terracotta, which has approximately Labertian reflectance and approximately constant albedo over the whole surface. Accordingly, the "global albedo" algorithm will be used. Figure 6.6 shows the reconstruction results; again, a 5 × 5 median-filtered version is also shown. It is apparent that the algorithm reduces the amount of noise substantially, yielding a visually pleasing result. Note, in particular, that the reconstructions contain detail, such as the ridges in the second object, that can only be inferred from the intensity image and could not be revealed by a filter operating on the range map alone.

We now test the algorithm on a more challenging object, a human face. Human skin does not fulfill the assumption of Lambertian reflectance nearly as well as terracotta; in addition, while albedo is approximately constant across much of the face, there are regions, such as the lips and eyebrows, where albedo changes noticeably. Figure 6.7 shows the results of the "global albedo" and "local albedo" algorithms on the face; again, a median-filtered range map is shown for comparison. While the "global albedo" reconstruction contains noticeable artefacts due to the non-uniform albedo, the "local albedo" reconstruction is much more smooth and true to the shape of the face. It is apparent that the algorithm can produce satisfactory results even in cases such as this one where the assumption of Lambertian reflectance is not fulfilled exactly. The figure also shows the result of texturing the "local albedo" reconstruction with the intensity image to illustrate the quality of textured models that can be obtained for applications such as virtual avatars.

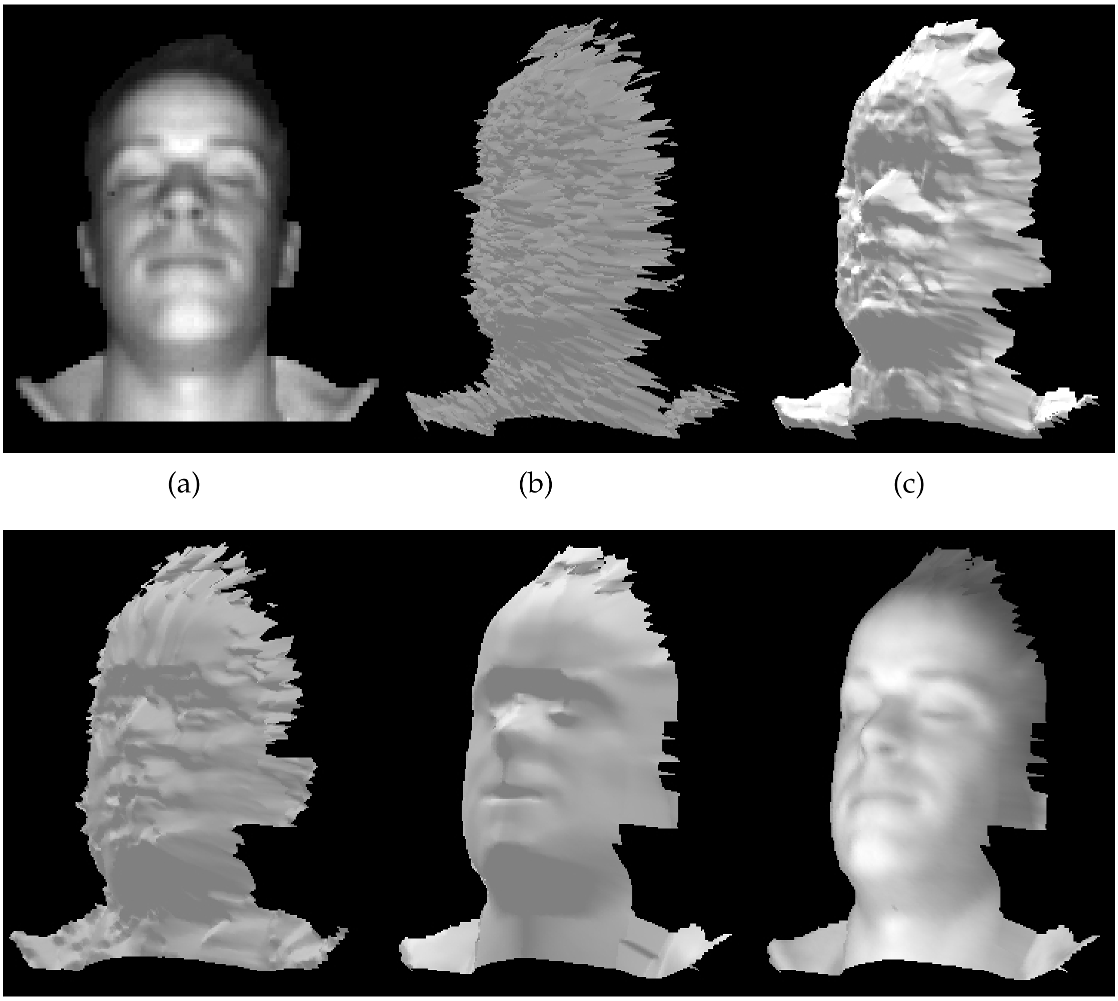
Finally, Figure 6.8 shows the results of the algorithm on the upper body of a person. Note how the shading constraint allows the cloth folds to be reconstructed faithfully. This example also illustrates the limitations of the algorithm: The head is reconstructed less well than in the previous example; we believe this is because there is too much albedo variation in an area of only a few pixels. The lowest part of the body is not reconstructed well either, and this is probably due to the low reflectivity of the material in this region, which leads to a large amount of noise in both the range map and the intensity image.

6.4 Discussion

We have shown that the shading information contained in the intensity image can be used to improve the quality of the range map substantially, in terms of both quantitative measures (RMS error) and subjective quality.

In many ways, the data obtained from a TOF camera are ideally suited for applying the shading constraint: The range map provides a rough "scaffolding" onto which the fine details obtained using the shading constraint can be placed. In this way, we avoid the ambiguities that are present in the traditional shape-from-shading (SfS) setting, where shape is estimated from an intensity image alone. Indeed, the additional constraints provided by the range map allow us to estimate the albedo of the surface within certain limits. A further advantage over the traditional SfS setting is that the position of the light source is known and that the TOF camera suppresses all other ambient illumination.

The current algorithm does, however, have its limitations. On current hardware, our implementation requries several minutes to process a single image. Though the implementation has not been optimized for speed, we do not think it is realistic that the algorithm can be accelerated enough to run at camera frame rates, even if a graphics processing unit (GPU) is used to carry out computations. For this reason, we will unfortunately not be able to use the shading constraint for the interactive applications that we will describe in the remainder of this thesis. Irrespective of the speed of the algorithm, though, we have shown that the range and intensity data combined contain detailed shape information; we hope that future work will find more efficient ways of extracting this information from the data.



(a) (b) (c)

(d) (e) (f)

Figure 6.7: Surface reconstruction of a human face. (a) Manually segmented intensity image, (b) measured range map, (c) 5 × 5 median-filtered range map, (d) “global albedo” reconstruction, (e) “local albedo” reconstruction, (f) “local albedo” reconstruction textured with intensity image. The renderings of the range maps are rotated 30 degrees around the vertical axis.

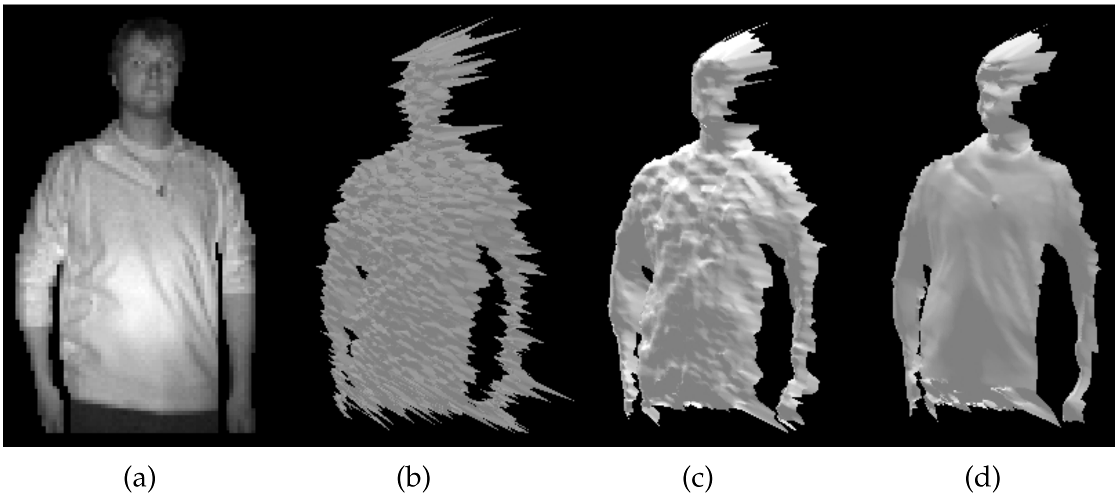


Figure 6.8: Surface reconstruction of a person's upper body. (a) Manually segmented intensity image, (b) measured range map, (c) 5 × 5 median-filtered range map, (d) "local albedo" reconstruction. The renderings of the range maps are rotated 30 degrees around the vertical axis.

Another limitation of the algorithm is that it does not cope well with range discontinuities (so-called *jump edges*). This requires the object of interest to be segmented from the background – either manually, as we have done here, or automatically. The reason why jump edges present a difficulty is that triangles that straddle a jump edge are almost perpendicular to the incoming light and are hence shaded with very low intensity. This disagrees with the observed intensity images, so, to compensate, the algorithm must either flatten the edge or assign very high albedo values to the corresponding pixels. A solution to this problem would be to ignore any triangle that straddles a jump edge. Jump edges could be identified either by searching for large jumps in the measured range values or by incorporating jump edges into the probabilistic model, as in the work of Nadabar and Jain [98].

Of course, other range sensors, such as laser range scanners, still provide far better accuracy than TOF camera range maps post-processed using our algorithm. The strength of the TOF camera, however, lies in its potential to be manufactured at low cost and in its high temporal resolution – though the algorithm is currently too slow for interactive applications, it can still be used to post-process high-frame-rate image sequences. The enhanced accuracy that is achieved using the shading constraint should open up new applications fields for the TOF camera.

7 Facial Feature Tracking

In this chapter, we will explore how the TOF camera can be used for tracking facial features, and we will demonstrate the resulting tracker in an interactive application. The detector for the facial features will be based on geometric features that describe the local properties of the object surface – whether it is planar, convex, concave, and so forth. Since the intensity image can be interpreted as a height field, where height is proportional to intensity, the geometric features can be computed on the intensity image as well. We will combine the geometric features with a very simple threshold-based classifier to obtain a robust detector for a prominent facial feature: the nose.

We will also examine how the TOF camera can be used to deal with the problem of scale changes. The effect of scale is a fundamental problem in computer vision: As the distance of an object from the camera changes, so does its apparent size in the image. Various approaches can be used to detect objects independently of their scale. For example, a scale-sensitive object detector can be made scale-invariant by applying it to a sequence of scaled versions of the image (an *image pyramid*). Alternatively, one can define features in such a way that they are inherently invariant to scale and then detect objects based on these features.

The TOF camera offers another possibility: Because it senses the physical shape of the object, we can compute features on the surface of the object (in spatial coordinates) instead of the image (in image coordinates). If the actual size of the physical object does not change (or not much), we can sidestep the problem of scale variations entirely.

Computing features on the object surface introduces an additional difficulty, however. Whereas the image is sampled on a regular pixel grid, the corresponding grid on the object surface is irregular because the spacing of the grid points increases with the distance from the camera. This irregular sampling makes it more difficult to compute features because many techniques, such as convolution with a filter kernel, assume that the data are sampled on a regular grid.

We deal with this problem by using the nonequispaced fast Fourier transform (NFFT), a generalization of the fast Fourier transform (FFT) that can deal with functions sampled at arbitrary locations. We will use the NFFT to transform the range map to the

Parts of this chapter are joint work with others. Martin Haker and I contributed approximately equally to the facial feature detector and tracker. I came up with the idea of using the NFFT to achieve scale-invariance, and Martin Haker and I contributed approximately equally to refining and implementing that idea. Parts of the work described here have previously been published in [16, 53, 54].

frequency domain, compute the features there, and then transform back to the spatial domain. As we will see, the resulting features are indeed scale-invariant, and, on a dataset containing scale changes, they have higher detection rates than the same features computed on the image in the conventional way.

Unfortunately, though, the scale-invariant features cannot be computed fast enough to allow interactive applications on current hardware. We will therefore use the conventional scale-variant features for the interactive applications. Nevertheless, our results demonstrate the potential that the data recorded by the TOF camera have for dealing with scale variations.

7.1 Geometric Features

The features we will use in the facial feature detector are the so-called *generalized eccentricites*, introduced in [7], which, as we will see, describe the local properties of a surface. To motivate the definition of the generalized eccentricities, we will start with a classical descriptor of surface properties, the Gaussian curvature [10]. If the surface can be described as a *height field* or *Monge patch*, i.e. if the surface points (x, y, z) can be described by a function f of x and y, with $z = f(x, y)$, then the Gaussian curvature K is given by

$$K = \frac{f_{xx} f_{yy} - f_{xy}^2}{1 + f_x^2 + f_y^2}. \tag{7.1}$$

The Gaussian curvature is non-zero only for points where the surface is non-planar in two directions; this is equivalent to saying that the surface cannot be deformed into a plane without introducing distortions. Intuitively, one can say that surfaces with non-zero Gaussian curvature cannot be formed from a sheet of paper without creasing or tearing the paper. Shapes that *can* be formed from a sheet of paper, such as a cylinder, have a Gaussian curvature of zero.

There are two basic types of shape with non-zero Gaussian curvature; these two types can be distinguished by the sign of the curvature K. Positive curvature ($K > 0$) means that the surface is curved in the same sense (either inwards or outwards) in both directions. Negative curvature ($K < 0$) means that one direction has a different sense of curvature than the other direction; this type of shape is also called a *saddle point*.

The property of non-zero Gaussian curvature is interesting because it means that the function f is intrinsically two-dimensional at this location, i.e. it cannot be described locally by a function of less than two variables. It can be shown [7, 97] that f is described fully by its values in the intrinsically two-dimensional regions.

To identify the points with non-zero Gaussian curvature and to distinguish between the two basic types of surface that can occur at these points (according to the sign of

K), we only need the numerator of Equation 7.1, which we will call D:

$$D = f_{xx} f_{yy} - f_{xy}^2. \tag{7.2}$$

The name D is intended to express that this quantity is the determinant of the Hessian $H = \left(\begin{smallmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{smallmatrix}\right)$ of f. We can rewrite D in an alternative form as follows:

$$\begin{aligned} D &= \frac{1}{4}(f_{xx} + f_{yy})^2 - \frac{1}{4}(f_{xx} - f_{yy})^2 - f_{xy}^2 \\ &= \frac{1}{4}\left((\Delta f)^2 - \varepsilon^2\right), \end{aligned} \tag{7.3}$$

where $\Delta f = f_{xx} + f_{yy}$ is the Laplacian and ε is the eccentricity, defined as

$$\varepsilon^2 = (f_{xx} - f_{yy})^2 + 4 f_{xy}^2. \tag{7.4}$$

Note that we have non-zero Gaussian curvature if the Laplacian and the eccentricity are not equal.

We are now ready to define the generalized eccentricities ε_n $(n = 0, 1, 2, \ldots)$, of which, as we will see, both the Laplacian and the eccentricity are special cases:

$$\varepsilon_n^2 = (c_n * f)^2 + (s_n * f)^2. \tag{7.5}$$

The $*$ is the convolution operator, and c_n and s_n are filter kernels corresponding to transfer functions C_n and S_n. These transfer functions are best described in terms of polar coordinates ρ (spatial frequency) and θ (orientation):

$$C_n(\rho, \theta) = i^n A(\rho) \cos(n\theta), \tag{7.6}$$

$$S_n(\rho, \theta) = i^n A(\rho) \sin(n\theta). \tag{7.7}$$

$A(\rho)$, the radial filter tuning function, is typically a power function in ρ. The filters c_n and s_n can be interpreted as orientation-sensitive derivative operators, where the exponent in $A(\rho)$ determines the order of the derivative and n determines the number of directions in which the operator is sensitive.

For certain choices of $A(\rho)$ and certain n, the generalized eccentricities are equivalent to various well-known derivative operators. For $A(\rho) = 2\pi\rho$, ε_1 is equal to the norm of the gradient $\|\nabla f\|_2$. For $A(\rho) = (2\pi\rho)^2$, ε_0 is equal to the absolute value of the Laplacian $|\Delta f|$, and ε_2 is equal to the eccentricity ε.

Before we go any further, let us briefly prove these equivalences. First of all, we note that the Cartesian coordinates corresponding to ρ and θ are

$$\begin{aligned} \xi_x &= \rho \cos(\theta), \\ \xi_y &= \rho \sin(\theta). \end{aligned} \tag{7.8}$$

(We use ξ_x and ξ_y for the Cartesian spatial frequencies instead of the more usual f_x and f_y to avoid confusion with the partial derivatives of f.) We also note that the transfer functions corresponding to the partial derivative operators in x- and y-direction are $D_x = i2\pi\xi_x$ and $D_y = i2\pi\xi_y$, respectively.

Let us now prove that ε_1 with $A(\rho) = 2\pi\rho$ is the norm of the gradient. We have

$$\begin{aligned} C_1 &= i\,2\pi\rho\cos(\theta) = i\,2\pi\xi_x = D_x, \\ S_1 &= i\,2\pi\rho\sin(\theta) = i\,2\pi\xi_y = D_y, \end{aligned} \tag{7.9}$$

i.e. $c_1 * f$ and $s_1 * f$ are just the partial derivatives f_x and f_y. Hence

$$\varepsilon_1^2 = f_x^2 + f_y^2 = \|\nabla f\|_2^2, \tag{7.10}$$

as claimed.

Turning to ε_0 with $A(\rho) = (2\pi\rho)^2$, we have

$$\begin{aligned} C_0 &= (2\pi\rho)^2 = (2\pi)^2\,(\xi_x^2 + \xi_y^2) \\ &= (i\,2\pi\xi_x)^2 + (i\,2\pi\xi_y)^2 \\ &= D_x^2 + D_y^2, \\ S_0 &= 0. \end{aligned} \tag{7.11}$$

Since D_x^2 and D_y^2 are the transfer functions for the second-order partial derivatives f_{xx} and f_{yy}, we have

$$\varepsilon_0^2 = (c_0 * f)^2 = (f_{xx} + f_{yy})^2 = (\Delta f)^2, \tag{7.12}$$

as claimed.

Finally, we turn to ε_2 with $A(\rho) = (2\pi\rho)^2$. We have

$$\begin{aligned} C_2 &= (2\pi\rho)^2\cos(2\theta) \\ &= (2\pi\rho)^2\,(\cos(\theta)^2 - \sin(\theta)^2) \\ &= (2\pi\rho)^2\left(\left(\frac{\xi_x}{\rho}\right)^2 - \left(\frac{\xi_y}{\rho}\right)^2\right) \\ &= (i\,2\pi\xi_x)^2 - (i\,2\pi\xi_y)^2 \\ &= D_x^2 - D_y^2, \\ S_2 &= (2\pi\rho)^2\sin(2\theta) \\ &= (2\pi\rho)^2\,2\cos(\theta)\sin(\theta) \\ &= (2\pi\rho)^2\,2\,\frac{\xi_x}{\rho}\cdot\frac{\xi_y}{\rho} \\ &= 2\,(i\,2\pi\xi_x)(i\,2\pi\xi_y) \\ &= 2D_xD_y. \end{aligned} \tag{7.13}$$

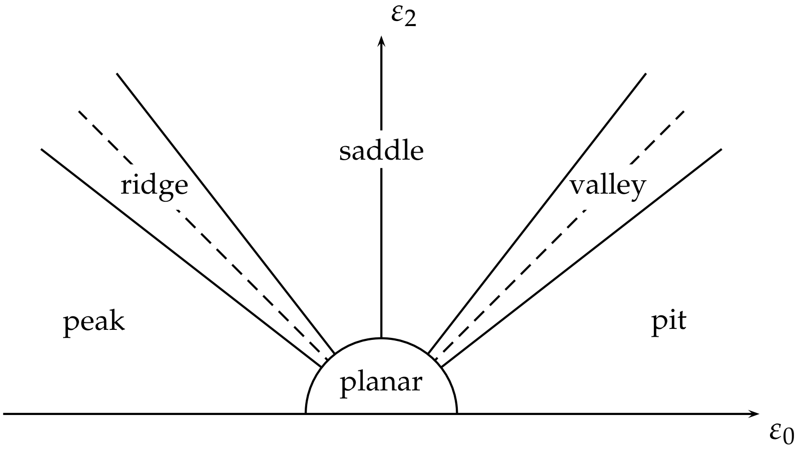


Figure 7.1: Six surface types can be distinguished in the half-plane defined by ε_0 and ε_2 (redrawn from [7]).

Hence

$$\varepsilon_2^2 = (f_{xx} - f_{yy})^2 + (2f_{xy})^2 = \varepsilon^2, \tag{7.14}$$

as claimed.

We have already seen that we can distinguish different surface types based on the sign of $D = \frac{1}{4}\left((\Delta f)^2 - \varepsilon^2\right) = \frac{1}{4}(\varepsilon_0^2 - \varepsilon_2^2)$ (with $A(\rho) = (2\pi\rho)^2$). We can achieve a finer distinction if we consider the two-dimensional coordinate system defined by ε_0 and ε_2. For this purpose, it is useful to define ε_0 to have not just the magnitude but also the sign of the Laplacian, i.e. $\varepsilon_0 = c_0 * f$. With this definition, we can distinguish six different surface types, which correspond to different regions in the coordinate system defined by ε_0 and ε_2 (see Figure 7.1). Regions with a Gaussian curvature of zero can be subdivided into three types: *planar* (no curvature in any direction), and *valley* and *ridge* (curved in one direction). Regions with a Gaussian curvature unequal to zero can also be subdivided into three different types: *saddle* (different sense of curvature in the two directions), and *pit* and *peak* (same sense of curvature in both directions).

Because the generalized eccentricities describe properties of surfaces, it is intuitive that they should be well suited for detecting features in range maps. However, they can also be applied to intensity images if we interpret an image as a height field $(x, y, I(x, y))$, where each point in the image is associated with a height proportional to its intensity $I(x, y)$. We will use this interpretation to compute geometric features on the intensity image as well as on the range map. The use of the geometrical interpretation of images for encoding, labelling and reconstruction is investigated in depth in [7].

7.2 Feature Computation in Spatial Coordinates

7.2.1 Image Coordinates Versus Spatial Coordinates

We saw in the previous section how the generalized eccentricities can be used to determine the surface properties of a height field or Monge patch $(x, y, f(x, y))$. The most obvious way of defining this Monge patch is to let x and y be pixel coordinates and to define $f(x, y)$ as the intensity or range corresponding to that pixel.

We can now evaluate the generalized eccentricities on this Monge patch as follows. Recall that the generalized eccentricities are defined using transfer functions C_n and S_n (see Equations 7.6 and 7.7). The obvious way of applying the filter kernels c_n and s_n corresponding to these transfer functions is to transform $f(x, y)$ to the frequency domain, multiply by the desired transfer function, and transform back. Because $f(x, y)$ is sampled on a regular grid of pixel coordinates, the transform to the frequency domain and back can be carried out efficiently using the fast Fourier transform (FFT). (We will refer to features computed in this way as *image-coordinate features*.)

This approach of computing the generalized eccentricities in image coordinates does, however, have an important disadvantage: If the distance of the object from the camera changes, its apparent size in the image also changes, and this, in turn, affects the values of the geometric features.

In the case of the range data, there is an obvious alternative: Because the range map encodes the true spatial shape of the object, we can define the Monge patch in terms of spatial coordinates, i.e. x and y are spatial coordinates parallel to the image plane, and $f(x, y)$ is the coordinate perpendicular to the image plane. We obtain these spatial coordinates of the object surface pixels by undoing the perspective transform of the camera. If the feature values are now computed in spatial coordinates instead of in image coordinates, they will no longer be influenced by the distance of the object from the camera.

Computing the features in spatial coordinates introduces an additional difficulty, however: The sampling points at which the values of $f(x, y)$ are known now form an irregular grid because the spacing between two grid points is proportional to their distance from the camera; see Figure 7.2 for an illustration of this. This means that the FFT, which requires $f(x, y)$ to be sampled on a regular grid, can no longer be used. One way of dealing with this would be to resample $f(x, y)$ on a regular grid, to compute the generalized eccentricities on this regular grid, and then to interpolate between the computed values to obtain the generalized eccentricities on the original irregular grid.

There is, however, a more elegant alternative: The nonequispaced fast Fourier transform (NFFT) is a generalization of the FFT that can be applied to functions sampled on arbitrary grids. We can use the NFFT to transform the irregularly sampled Monge

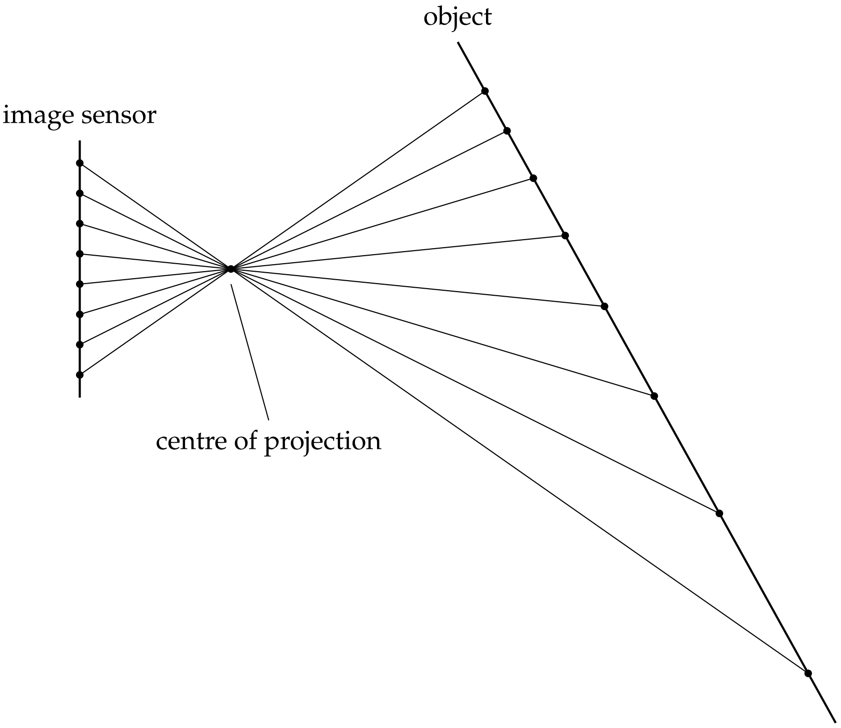


Figure 7.2: Whereas the sampling grid on the image sensor is regular, the sampling grid on the object surface is irregular because the distance between two grid points depends on their distance from the camera.

patch to the frequency domain, apply the transfer functions as before, then transform back. We will refer to features computed in this way as *spatial-coordinate features*.

7.2.2 The Nonequispaced Fast Fourier Transform (NFFT)

The nonequispaced fast Fourier transform (NFFT) [110] is an approximative algorithm for the fast evaluation of sums of the form

$$f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k} \boldsymbol{x}_j}, \tag{7.15}$$

where the $\boldsymbol{x}_j \in [-\frac{1}{2}, \frac{1}{2})^d$, $j = 1, \ldots, M$ are arbitrary nodes in the spatial domain, the $\boldsymbol{k}$ are frequencies on an equispaced grid $I_{\boldsymbol{N}}$, and the $\hat{f}_{\boldsymbol{k}}$ are the corresponding Fourier coefficients. The equispaced frequency grid $I_{\boldsymbol{N}}$ is defined as

$$I_{\boldsymbol{N}} := \left\{ \boldsymbol{k} = (k_t)_{t=1,\ldots,d} \in \mathbb{Z}^d : -\frac{N_t}{2} \leq k_t < \frac{N_t}{2}, t = 1, \ldots, d \right\}, \tag{7.16}$$

where $\boldsymbol{N} = (N_1, \ldots, N_d)$ is the so-called *multibandlimit*, which specifies the band limit (i.e. the number of spectral coefficients) along each direction. (Note that all N_t must be even.)

Equation 7.15, which we will call the nonequispaced discrete Fourier transform (NDFT), describes the transform from the frequency domain to the spatial domain. Note at this point that the nomenclature for the nonequispaced setting is exactly the reverse of that for the equispaced setting. The classical equispaced discrete Fourier transform (DFT) transforms from the spatial domain to the frequency domain, and the inverse DFT transforms from the frequency domain to the spatial domain. In the nonequispaced setting, however, the NDFT transforms from the frequency domain to the spatial domain, and the inverse transform (which we will cover in more detail in a moment) transforms from the spatial domain to the frequency domain.

While the direct evaluation of the sum in the NDFT requires $O(M \cdot |I_{\boldsymbol{N}}|)$ operations, the NFFT algorithm reduces the asymptotic complexity to $O\left(|I_{\boldsymbol{N}}| \log(|I_{\boldsymbol{N}}|) + M \log(\frac{1}{\varepsilon})\right)$, where ε is the desired accuracy.

In the more familiar case of the equispaced DFT, the matrix that describes the transform is unitary, and so the same algorithm – typically the fast Fourier transform (FFT) – can be used for both the forward and the inverse transform. In the nonequispaced setting, however, the matrix that describes the transform need not even be square, and even if it is, it is not guaranteed to be regular. In many cases, therefore, a true inverse transform does not exist. Instead, one can compute a least-squares solution (if the solution is overdetermined) or apply a regularization condition (if the solution is underdetermined) [78]. In both cases, the solution is found by combining the forward transform (the NFFT) with an iterative minimization scheme such as the conjugate gradient method.

In our application, we have $M \leq |I_{\boldsymbol{N}}|$, and so the solution is typically underdetermined. The regularization that we apply yields the solution with minimal energy.

7.2.3 Using the NFFT for Feature Computation

To compute features in spatial coordinates using the NFFT, we first invert the camera projection to obtain the Cartesian coordinates x, y, z of each pixel, where the coordinate system is defined so that the x- and y-axes are parallel to the image plane and the z-axis is perpendicular to it. We interpret these points as a Monge patch $z(x, y)$, sampled on an irregular x-y-grid; the grid points define the nodes $\boldsymbol{x}_j$ for the NFFT.

Because the NFFT requires the nodes to lie within the range $[-\frac{1}{2}, \frac{1}{2}) \times [-\frac{1}{2}, \frac{1}{2})$, as noted in the previous section, the x-y-coordinates need to be scaled suitably. A scaling in the spatial domain corresponds to an equal but opposite scaling in the frequency domain, so the particular scaling factor that is chosen affects the way we must interpret the Fourier coefficients in the frequency domain. For this reason, it is important to use the same scaling across all images. To define the scaling factor, we use the concept of an *equivalence range*: For an object at the equivalence range, the NFFT produces the same spectrum as the FFT. To find the scaling that corresponds to a given equivalence range e, we take a plane that is parallel to the image plane at a distance e from the camera and intersect this plane with the camera's field of view, yielding a rectangle. We then set the scaling factor so that the longer side of the rectangle is scaled to the interval $[-\frac{1}{2}, \frac{1}{2}]$.

Because the x-y-extent of the camera's field of view increases with the distance from the camera, points beyond the equivalence range may have x-y-coordinates outside the range $[-\frac{1}{2}, \frac{1}{2}) \times [-\frac{1}{2}, \frac{1}{2})$; these points are discarded. This implies that the equivalence range needs to be defined in such a way that the objects of interest fit inside this clipping volume. In addition, we shift the centroid of the objects to $x = 0$, $y = 0$ to ensure that they lie centrally within the clipping volume.

Before the range map is passed to the NFFT, we segment the foreground object using adaptive range and intensity thresholds and discard the background pixels. There are several reasons for doing this: (i) Steep edges between the foreground and background can lead to ringing artefacts. (ii) The grid nodes in the background are spaced further apart, and the greater the spacing, the lower the frequency where aliasing sets in. (iii) Passing fewer points to the NFFT reduces the amount of computation that has to be carried out.

The final preprocessing step we perform is to subtract a constant offset from the z values to bring the maximum z value to zero. This is done because, as explained in the previous section, the transform from the spatial domain to the frequency domain is typically underdetermined and the NFFT computes a minimal-energy solution in this case. This implies that the background region, where there are no grid nodes, is implicitly set to zero. By bringing the maximum z value to zero, we avoid steep edges between the foreground and background, which could cause ringing artefacts.

7.3 Facial Feature Detection and Tracking

We will now use the generalized eccentricities ε_0 and ε_2, evaluated on both the range map and the intensity image, to classify each pixel as either "nose" or "non-nose".

Because the feature space spanned by ε_0 and ε_2 has a radial structure (see Figure 7.1), we convert these features to polar coordinates $r = \sqrt{{\varepsilon_0}^2 + {\varepsilon_2}^2}$ and $\phi = \text{atan2}(\varepsilon_2, \varepsilon_0)$. For each pixel, we obtain a feature vector $\boldsymbol{F} = (F_1, \dots, F_4)$ that contains the values of r and ϕ for both the range and intensity data.

On a set of training images, we hand-label the positions of the nose and compute the features at these locations. We then use these training data to train a very simple classifier, as follows. For each feature F_j, we compute the minimum and maximum values $F_{\mathrm{min}\,j}$ and $F_{\mathrm{max}\,j}$ of that feature across all of the labelled nose pixels. In this way, we obtain vectors $\boldsymbol{F}_{\mathrm{min}}$ and $\boldsymbol{F}_{\mathrm{max}}$ that define an axis-aligned bounding box in feature space (the *classification box*). A pixel is classified as "nose" if its feature values fall within the box and "non-nose" otherwise.

We can control the tradeoff between false-positive rate and false-negative rate by scaling the box around its centre to obtain new classification box limits

$$\begin{aligned} \hat{\boldsymbol{F}}_{\mathrm{min}} &= \boldsymbol{F}_{\mathrm{centre}} - \alpha \boldsymbol{F}_{\mathrm{halfwidth}}, \\ \hat{\boldsymbol{F}}_{\mathrm{max}} &= \boldsymbol{F}_{\mathrm{centre}} + \alpha \boldsymbol{F}_{\mathrm{halfwidth}}, \end{aligned} \tag{7.17}$$

where $\boldsymbol{F}_{\mathrm{centre}} = \frac{\boldsymbol{F}_{\mathrm{min}} + \boldsymbol{F}_{\mathrm{max}}}{2}$ and $\boldsymbol{F}_{\mathrm{halfwidth}} = \frac{\boldsymbol{F}_{\mathrm{max}} - \boldsymbol{F}_{\mathrm{max}}}{2}$. The parameter $\alpha \in [-1, \infty)$ controls the scaling of the box.

This very simple classifier can be evaluated quickly and yields good results in our application. Nevertheless, we would expect a more sophisticated classifier, such as a support vector machine (svm), to further improve the classification performance.

Once we have raw detections, we want to use these detections to track the position of the nose from frame to frame.

Typically, the facial feature detector detects a cluster of several "nose" pixels around the tip of the nose. We therefore begin by finding all connected components of "nose" pixels in the current frame. If there are several connected components, we choose the one that is closest to the position of the nose in the previous frame.

To identify a single pixel within the connected component as the position of the nose, we search for the pixel whose feature vector is closest to the centre of the classification box $\boldsymbol{F}_{\mathrm{centre}}$; in a sense, this is the most "prototypical" nose pixel.

We then perform a subpixel refinement of this nose position by computing a weighted centroid of the pixels in a certain neighbourhood around the initial pixel; the weight of a pixel decreases with the distance of its feature vector from the centre of the classification box:

$$\begin{aligned} \boldsymbol{x}_{\mathrm{nose}} &= \frac{\sum_{\boldsymbol{x} \in N} w_{\boldsymbol{x}} \boldsymbol{x}}{\sum_{\boldsymbol{x} \in N} w_{\boldsymbol{x}}} \quad \text{with} \\ w_{\boldsymbol{x}} &= \mathrm{e}^{-\frac{\|\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{F}_{\mathrm{centre}}\|_2{}^2}{\|\boldsymbol{F}_{\mathrm{halfwidth}}\|_2{}^2}}, \end{aligned} \tag{7.18}$$

where N is a neighbourhood of pixels around the initial nose position, $\boldsymbol{F}(\boldsymbol{x})$ is the feature vector for pixel $\boldsymbol{x}$, and $\boldsymbol{x}_{\mathrm{nose}}$ is the subpixel-refined nose position.

Finally, we track the position of the nose from frame to frame using a Kalman filter [87], with constant-speed unaccelerated motion as the underlying dynamic system model.

7.4 Results

The facial feature detector and tracker was implemented in MATLAB. To evaluate the NFFT, we used the NFFT library (version 3.1.0) [75], which is written in C; the MEX interface was used to call this code from MATLAB. The NFFT library itself used the FFTW library (version 3.2.1) [46] to evaluate the FFTs that occur in the NFFT algorithm.

We also implemented an interactive version of the facial feature tracker in C++. Because the spatial-coordinate image features cannot be computed quickly enough for interactive applications, this version used only the image-coordinate features. The C++ implementation also used the FFTW library.

The radial filter tuning function was set to $A(\rho) = (\pi\rho)^2 \cdot e^{\frac{-\pi\rho^2}{\sigma^2}}$ with $\sigma = 0.33$ for the intensity features and $\sigma = 0.15$ for the range features.

We will first evaluate the performance of the facial feature detector on image-coordinate features. Because these features are scale-variant, we will use a dataset that does not contain scale changes. (We will examine later how the image-coordinate features react to scale changes.)

We used a SwissRanger SR3000 camera [100] to acquire images of 13 subjects; for each subject, nine images with different head orientations were taken. In all images, the distance between the subject's face and the camera was approximately 60 cm. The position of the tip of the nose was hand-labelled in the images.

We trained the detector on the data from three of the subjects and tested it on the remaining ten subjects. We count detection rates and false-positive rates on a per-image basis, as follows: If at least one pixel within a five-pixel radius of the labelled nose location is classified as "nose", we count this as a detection; otherwise, it is a false negative. Similarly, if one or more pixels outside the five-pixel radius are erroneously classified as "nose", we count this as a false positive. Note that an image can simultaneously produce a false negative and a false positive if the true nose is not detected and pixels in a different region are erroneously classified as "nose" instead.

As is common when evaluating classifiers, we will plot detection rate against false-positive rate to obtain a receiver operating characteristic (ROC) curve [43]. Note, however, that because we define detection rate and false-positive rate on a per-image basis, these curves do not share all of the characteristics of standard ROC curves. However, we believe that a per-image evaluation is more useful than a per-pixel evaluation because what we are really interested in is how often the detector identifies the nose correctly in a whole image.

Figure 7.3 shows the ROC curves that we obtain. We compare the result of using only the range features, only the intensity features, or both. As the plot shows, the range features yield a more accurate classification than the intensity features, with an equal error rate (EER) of 27.4% on the range features compared to 42.2% on the intensity

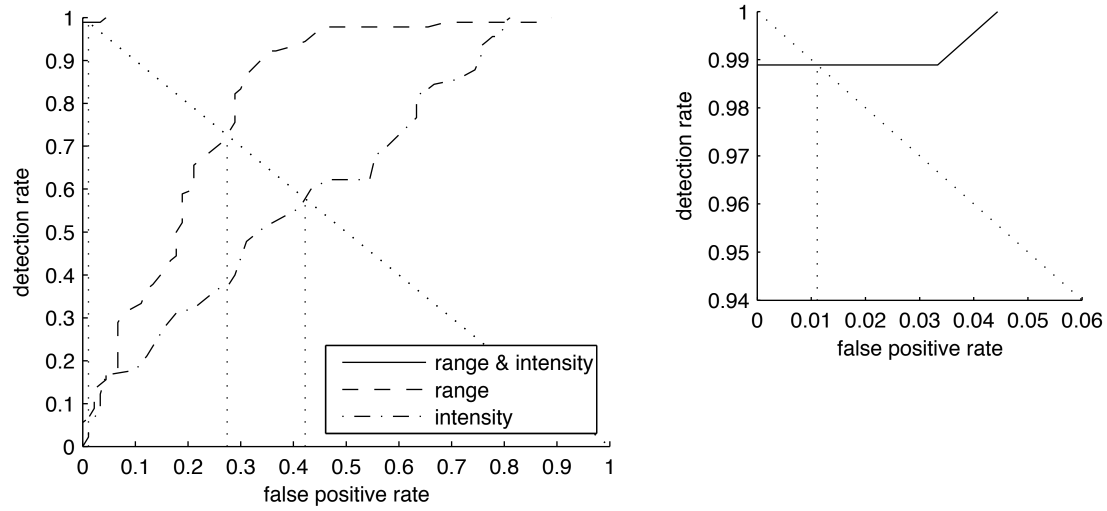


Figure 7.3: ROC curve for the facial feature detector based on image-coordinate features, using either range features, intensity features, or both. A closeup of the top left corner of the plot is shown on the right.

features. However, the combination of range and intensity features yields markedly better results than either type of feature alone, with an EER of 1.1%. Figure 7.4 shows some examples of detection results obtained using the combined range and intensity features on the test set. (Our results here are slightly better than the earlier results reported in [53] because we have optimized the radial filter tuning function since then.)

We will now compare the scale-variant image-coordinate features used so far with the scale-invariant spatial-coordinate features. To verify that the spatial-coordinate features are scale-invariant, we computed them on synthesized range maps of a sphere at various distances from the camera. Figure 7.5 shows the value of ε_0 at the apex of the sphere as a function of distance; the feature was computed both in image coordinates and in spatial coordinates. (ε_2 is not shown because it is identically zero for objects that have the same curvature in all directions.) While the image-coordinate version of the feature changes noticeably with distance, the spatial-coordinate version remains almost constant. Note also that at the equivalence range, which was set to $e = 0.5$ m for this test, the values of the image-coordinate and spatial-coordinate versions of ε_0 are the same.

We will now compare the performance of the image-coordinate and spatial-coordinate features on the nose detection task. We used the same training set as in the previous test (three subjects at a distance of 60 cm from the camera), but a different test set. To evaluate how well the features cope with scale changes, we recorded a test

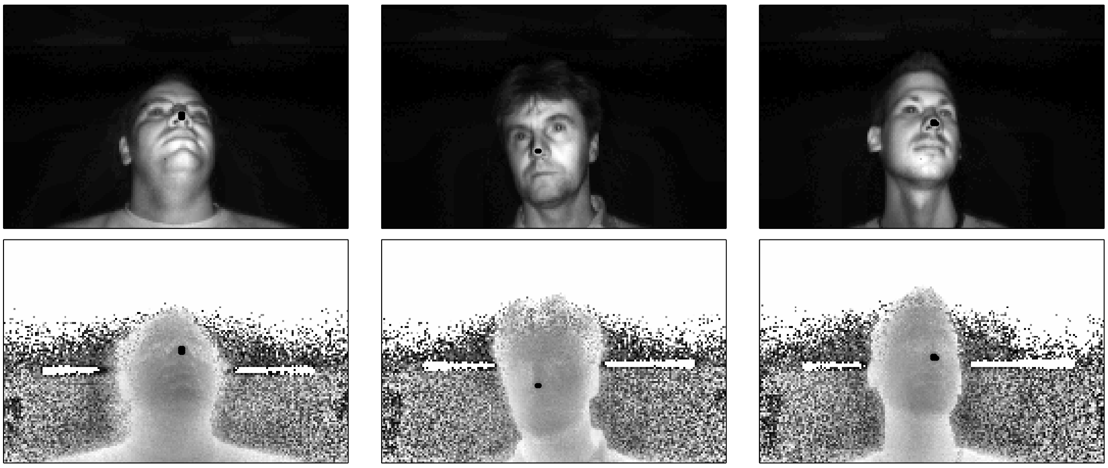


Figure 7.4: Sample detection results of the facial feature detector based on image-coordinate features, using both intensity and range features (top: intensity image, bottom: range map). The pixels marked in black indicate detected nose pixels.

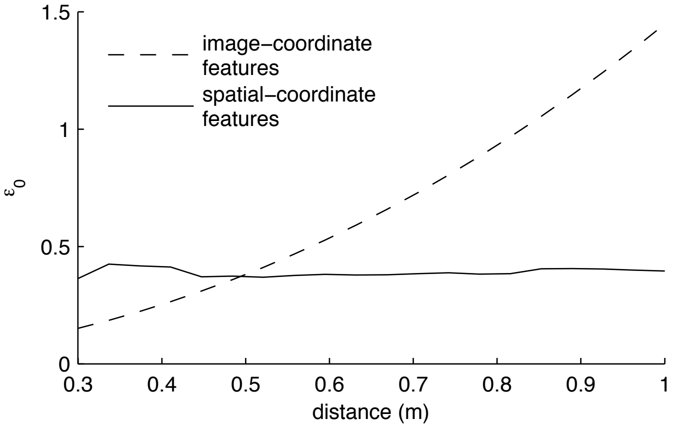


Figure 7.5: Generalized eccentricity ε_0 as a function of distance for a synthesized range map of a sphere. When computed in image coordinates, the feature is sensitive to scale, whereas in spatial coordinates, the feature value stays almost constant.

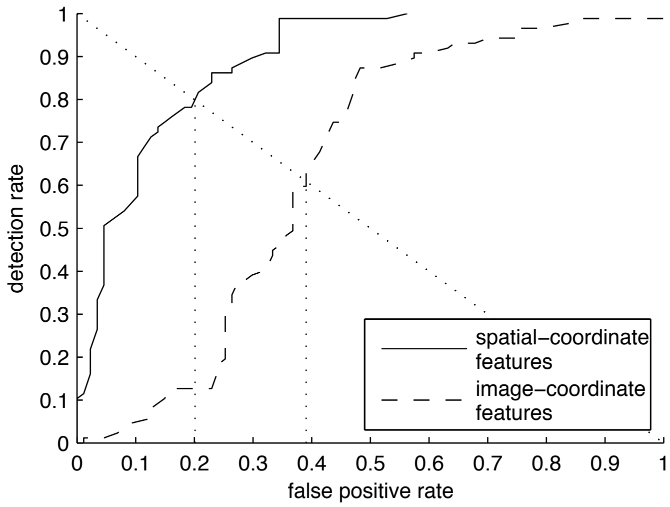


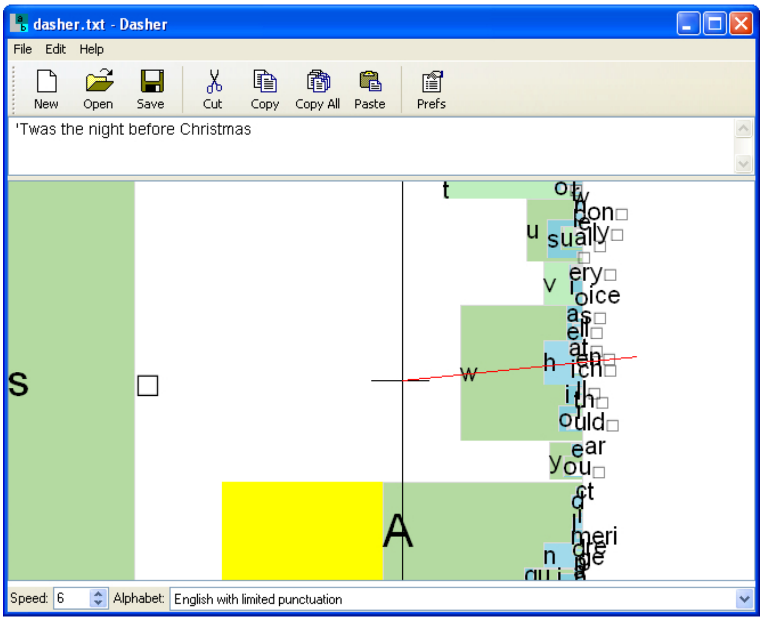
Figure 7.6: ROC curve of the facial feature detector on a dataset containing scale changes. Only the range features were used, computed either in image coordinates or in spatial coordinates.

set consisting of 87 images of a single subject (who was not identical to any of the subjects from the training set) at varying distances of 35 cm to 70 cm from the camera.

Because only the range features can be computed in spatial coordinates, we first examine classification performance on range features alone. Figure 7.6 shows the ROC curves for spatial-coordinate range features versus image-coordinate range features. The performance of the spatial-coordinate features is clearly superior, with an EER of 20.1% compared to 39.1% for the image-coordinate features.

Though the intensity features must be computed in image coordinates and are thus not scale-invariant, they still boost the performance of the detector when combined with the range features. The EER for intensity features combined with image-coordinate range features is 4.6%; when spatial-coordinate range features are used, the EER drops to 0%, i.e. the detector does not make any errors on the test set – a larger test set would be required to measure the EER with better precision. (The ROC curves for these combined range and intensity detectors are omitted because they do not show much meaningful information.) Note that the EER for the intensity features alone was 78.2%, so again it is only the combination of the intensity and range features that produces the good classification results.

While the spatial-coordinate features yield superior classification performance, they do require more computation than the image-coordinate features. On a 2.66 GHz Intel E6750 CPU, our MATLAB implementation requires 0.1 s to compute the image-coordinate features, versus 5 s for the spatial-coordinate features. This difference is mainly due to the NFFT, which has the same asymptotic time complexity as the FFT



'Twas the night before Christmas

Speed: 6

Alphabet: English with limited punctuation

Figure 7.7: Screenshot of Dasher [136], an alternative text input tool that can be controlled using a variety of input devices including eye trackers, head switches, or the facial feature tracker.

but with a larger constant factor. This means that while the spatial-coordinate features are an attractive choice for offline applications, they are not suitable for interactive applications on current hardware.

We have implemented a C++ version of the facial feature tracker [16] that uses the image-coordinate features and runs at camera frame rates (up to 75 frames per second, or 13.3 ms per frame, on a 2.66 GHz Intel E6750). We have evaluated the usefulness of this tracker for interaction tasks by using it to control Dasher, an alternative text input tool [136]. In Dasher, text is written continuously by "steering" towards the letters to be input (see Figure 7.7). A wide variety of input devices can be used with Dasher, including mice, eye trackers, and head switches. When controlling Dasher using the nose tracker, we have achieved a rate of 12 words per minute (wpm) [16]; for comparion, subjects using eye trackers achieve rates between 15 and 25 wpm [136].

7.5 Discussion

There are several conclusions we can draw from the results presented in this chapter. First, we have seen that the combination of range and intensity data yields a substantial improvement in performance on the facial feature detection task compared to either type of data alone. We will see a similar effect for face detection in Chapter 8, and we see this as evidence that the combination of range and intensity makes the TOF camera a powerful tool for a variety of object detection tasks. Within the context of the European Network of Excellence COGAIN [27], we have used the facial feature tracker to control Dasher, an alternative text input tool.

Second, we have explored how features can be computed in spatial coordinates using the nonequispaced fast Fourier transform (NFFT). This eliminates the typical problem of scale variations that occur in image coordinates when the distance of the object from the camera changes. Working in spatial coordinates avoids the need for dealing with scale variations in some other way, for example with an image pyramid that contains multiple versions of the same image at different scales. However, using the scale-invariant spatial-coordinate features is not just a question of convenience; an approach that deals with scale variations after the fact, using an image pyramid or other means, is potentially more prone to false detections because it cannot distinguish between a large, distant false positive and a closer, smaller true detection that has the same apparent size. In contrast, an approach based on spatial-coordinate features will only produce detections that match the true physical scale of the object to be detected. As we have seen, a facial feature detector based on spatial-coordinate features does indeed have better detection performance than one based on image-coordinate features.

The disadvantage of our current approach to computing spatial-coordinate features is that it requires more computation than the image-coordinate features and is thus not suitable for interactive applications on current hardware. It is a question for future research to investigate whether a similar result can be achieved with less computation. Since the distance to each pixel is known, this could be used to compensate for the effect of distance on the value of image-coordinate features. Alternatively, the distance value could be passed to the classifier as an additional feature; a suitable classifier might be able to learn the relationship between distance and feature values from the training data.

Another open question is how the concept of computing features in spatial coordinates could be applied not only to range features but also to intensity features. Whereas the object surface can be defined as a function $z(x, y)$ on the x-y-plane, intensity should properly be defined as a function on the object surface, an irregularly shaped two-dimensional manifold. Moreover, intensity itself decreases as distance increases because there is less and less light arriving per surface area. To obtain features that are distance-invariant, this effect also needs to be compensated for.

On a general level, we believe we have shown how the TOF camera can deal with some of the variations that occur in conventional cameras because of the nonlinearities introduced by the perspective projection. The range map measured by the TOF camera effectively allows this perspective projection to be inverted, thereby eliminating the corresponding nonlinearities.

8 Face Detection

In this chapter, we will examine how a time-of-flight camera can be used for face detection. We will extend the well-known face detection algorithm of Viola and Jones [132] to TOF images; as we will show in the results section, the detector trained on the combined range and intensity data not only has a higher detection rate than detectors trained on either type of data alone, but it also requires fewer features and therefore has a shorter running time.

The Viola-Jones face detector is computationally very efficient while at the same time achieving good detection rates. This is due to three important characteristics: (i) The detector is based on image features that can be evaluated quickly and in constant time, independent of the size of the feature; (ii) the detector selects a set of highly discriminative image features using the *AdaBoost* algorithm; (iii) the detector is structured into a cascade of progressively more sophisticated stages. Since most candidate regions in an image are very dissimilar to a face, the early stages of the cascade can discard these regions with little computation; the later stages of the cascade, which require more computation, need to process only a small proportion of candidate regions.

The attractive properties of the Viola-Jones face detector have motivated a large number of researchers to extend this work in various ways, including the use of different features, modifications to the AdaBoost algorithm, and the application to different types of object detection tasks (see e.g. [6, 22, 84]). The algorithm has also already been applied to TOF data [58]. However, this previous work does not extend the Viola-Jones detector itself to use range features; instead, a standard Viola-Jones detector trained on images from a conventional camera is used to find candidate faces in the TOF image; a final detector stage then computes the average distance of each candidate from the range map and rejects candidates whose size does not match the expected size of a face at this distance.

In contrast, the approach we will use is to include range features as well as intensity features in the set of features used by the detector. As we will show in the results section, the features chosen by the resulting detector consist of an approximately equal number of range and intensity features; the detector has a higher detection rate and shorter running time than detectors trained on the same training samples, but using

Parts of this chapter are joint work with Kolja Riemer, who wrote most of the code as part of his diploma thesis [113], which he conducted under my supervision. A revised version of the work described here has been published in [19].

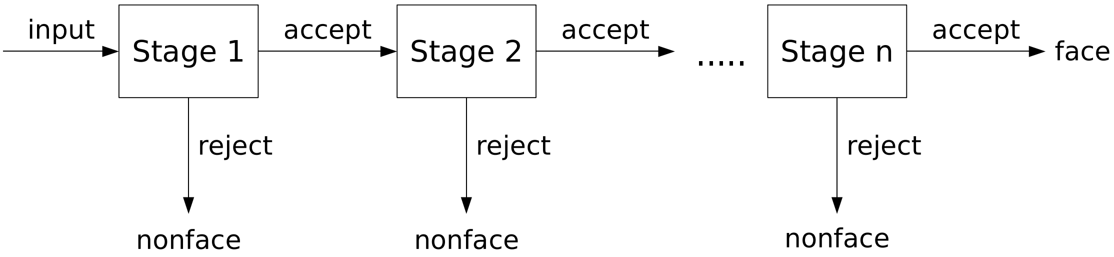


Figure 8.1: Cascade structure of the Viola-Jones face detector.

either the range or the intensity information alone. This underlines the results from the facial feature detection task in Chapter 7, where the combination of range and intensity also yielded better detection performance than either type of data alone.

8.1 Method

We use the basic face detection method of Viola and Jones [132] (which we will summarize briefly) but extend the set of features used to both range and intensity features. Since the method was first described, a number of authors have made improvements to the method (see e.g. [6, 22, 84]), but we use the original algorithm here because we are initially more interested in the difference made by using range data rather than in absolute performance.

The Viola-Jones face detector consists of a cascade of stages that typically become more sophisticated as one progresses through the cascade (see Figure 8.1). The idea is that the overwhelming majority of subregions in an image are nonfaces, and that most of these subregions are "easy", i.e. they can be identified as nonfaces with little computation. Thus, the first stage of the detector contains a computationally efficient classifier that can immediately reject most subregions as being nonfaces; no further processing is carried out on these subregions. Only a small fraction of subregions (both true faces and "hard" nonfaces) are passed on to the next stage for further processing. This next stage performs more computation and, by doing so, can again reject most of the subregions as being nonfaces, passing only a small fraction of subregions on to the next stage, and so on. In this way, the average effort per subregion is kept low because the overwhelming majority of subregions are rejected in the first few stages.

If the detection rate and false-positive rate of the i-th stage (on the input it receives from the previous stage) are d_i and f_i, then the overall detection and false-positive rates of an n-stage cascade are $D = \prod_{i=1}^{n} d_i$ and $F = \prod_{i=1}^{n} f_i$, respectively. A common approach is to train each stage to achieve the same detection rate d and false-positive

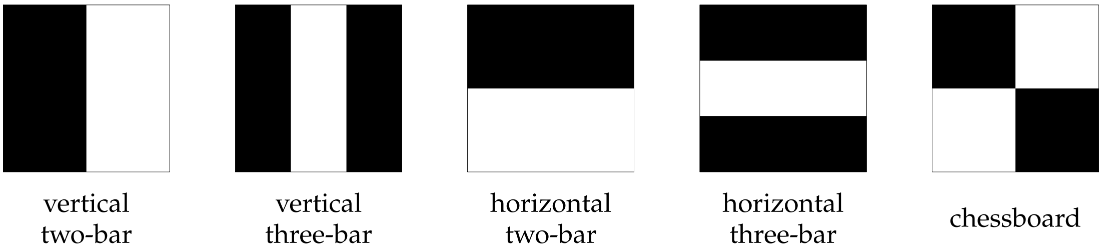


Figure 8.2: Haar-like features used by the Viola-Jones face detector. The feature value is obtained by summing the pixels in the white rectangle(s), then subtracting the sum of pixels in the black rectangle(s).

rate f on its respective input; this results in overall detection and false-positive rates of $D = d^n$ and $F = f^n$.

Each cascade stage is a boosted classifier trained using the AdaBoost algorithm [45]; a boosted classifier combines several *weak classifiers* (each of which performs only slightly better than chance) into a *strong classifier* (which performs substantially better than the individual weak classifiers). The weak classifiers in the Viola-Jones algorithm are obtained by applying a threshold to an image feature.

The image features, finally, are composed of adjacent rectangles (see Figure 8.2); the pixels within each rectangle are summed together, and the resulting values are added or subtracted to obtain the final feature value. For example, the value of the "vertical two-bar" feature is obtained by summing the pixels in the white rectangle and subtracting the sum of pixels in the black rectangle. These features (which are often called *Haar-like features* because of their similarity to Haar wavelets) have the advantage that they can be evaluated in constant time, independent of their size, using a data structure known as an *integral image*.

The feature set for training the detector is obtained by scaling these features to all possible widths and heights and translating them to all possible positions in the image. Also, each feature (at each size and position) may be evaluated either on the range data or on the intensity data.

Training of the cascade now proceeds as follows. We begin with a training set of face and nonface image patches of constant size. These are used to train the first cascade stage to the desired detection and false-positive rate (evaluated on a validation set). Now, because the next stage will never see those nonface patches that the first stage rejects, we discard all nonface samples rejected by the first stage from the training and validation set, keeping only the false positives. To bring the training and validation set back to their original sizes, we generate new nonface samples by scanning the cascade that has been trained so far across a set of images not containing faces and adding those subregions that the cascade erroneously classifies as faces to the training or validation set until both have been replenished. We continue adding stages to the cascade in this way until the false-positive rate of the cascade reaches a set target.

Detection proceeds by scanning the cascade across the input image in steps of a certain size. To be able to detect faces of different sizes, the subwindow processed by the detector, along with the features contained in it, is progressively scaled up by a certain factor until it reaches the size of the complete image.

8.2 Results

The training data for the face detector were recorded using a SwissRanger SR3000 camera [100]. The training set consists of 1310 images (with a resolution of 176 × 144 pixels) showing faces of 17 different persons, in different orientations and with different facial expressions, as well as 4980 images not containing faces. Each face image was labelled by hand with a square bounding box containing the face; some background was included in the bounding box, since previous researchers had reported that this yielded slightly better results than a more tightly cropped bounding box (see the discussion in [132, Section 5.1]).

The images were split up into a training, validation, and test set, containing 70%, 23%, and 7% of the images, respectively. (The training set is used to select the best weak classifiers for each cascade stage, the validation set is used to evaluate whether the stage has reached its goal detection rate and false-positive rate, and the test set is used to test the final cascade after training is completed.) Face images were cropped to the face bounding box and resized to 24 × 24 pixels (see Figure 8.3 for examples). To increase the number of face images in each set, we added versions of each image that were rotated left and right by 3 degrees. After this step, a mirrored version of each face image (including the rotated ones) was also added to the set. The nonface images were full frames of 176 × 144 pixels; to generate examples for training the first cascade stage, subimages of 24 × 24 pixels were cut out of the nonface images. For the second and subsequent stages, new negative examples were generated by scanning the cascade trained so far across the nonface images and collecting false positives (see also Section 8.1).

In all, there were 5412 faces and 3486 nonface images in the training set, 1752 faces and 1145 nonface images in the validation set, and 534 faces and 349 nonface images in the test set. The data set is publicly available at `www.artts.eu/publications/3d_tof_db`, and a competition for time-of-flight face detection based on this dataset has been announced (see `www.artts.eu/events/competition`).

We trained a detector on the combined range and intensity data as well as on the range and intensity data alone. The target detection rate and false-positive rate for each stage were set to $d = 0.995$ and $f = 0.4$, respectively; the target false positive rate

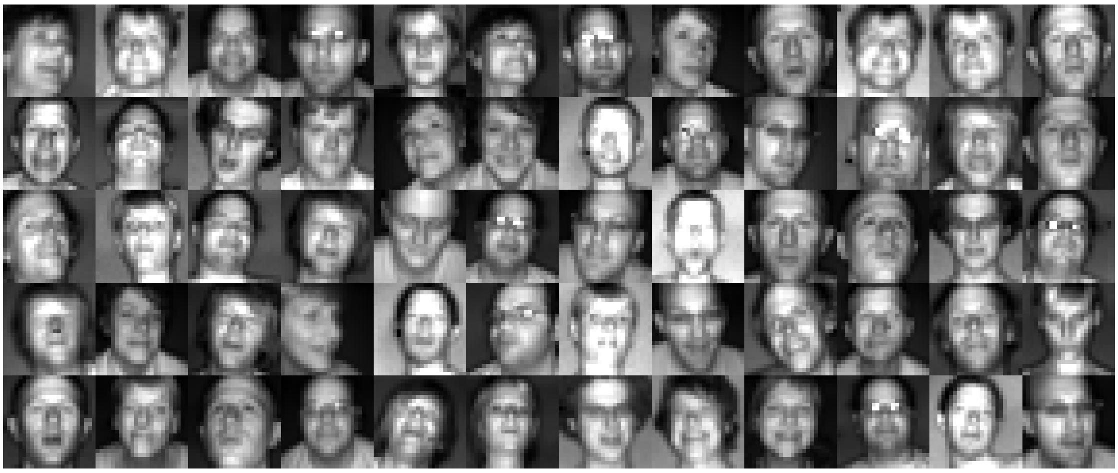


Figure 8.3: Examples of intensity images from the face training set.

for the complete detector was set to 10^{-8}. The range-and-intensity detector as well as the range-only detector were successfully trained to this target rate. Training of the intensity-only detector did not reach the target rate; training was stopped manually when the detector had added over 1500 features to the cascade stage it was training and the false-positive rate of the stage had stagnated without reaching the goal rate. (This is a typical sign that the detector can no longer generalize from the training to the validation set.) We trained another intensity detector with a lower detection rate per stage of $d = 0.99$; training for this detector did complete, but its performance was consistently worse than that of the detector with $d = 0.995$ whose training was aborted. For this reason, we will only use the latter detector in the tests that follow. In addition, we also attempted a training run for all three detectors with $d = 0.997$; the range-and-intensity detector was the only detector that completed the training, and we will show the results for this detector along with the results for all three detectors with $d = 0.995$.

Figure 8.4 shows ROC curves (computed as in [22]) for the detectors. The range-and-intensity detector with $d = 0.997$ outperformed all other detectors across the whole range of false-positive rates. Comparing the three detectors with $d = 0.995$, we find that for false positive rates above $1.5 \cdot 10^{-6}$, the intensity-only detector achieves a slightly higher detection rate than the range-and-intensity detector. Below this point, the range-and-intensity detector achieves better detection rates. Both detectors are markedly better than the range-only detector over the whole range of false-positive rates shown.

All detectors achieve good detection rates even for a false-positive rate of zero. This is an indication that our test set is relatively "easy" compared to, for instance, the MIT+CMU test set [115], on which the Viola-Jones algorithm produces slightly higher

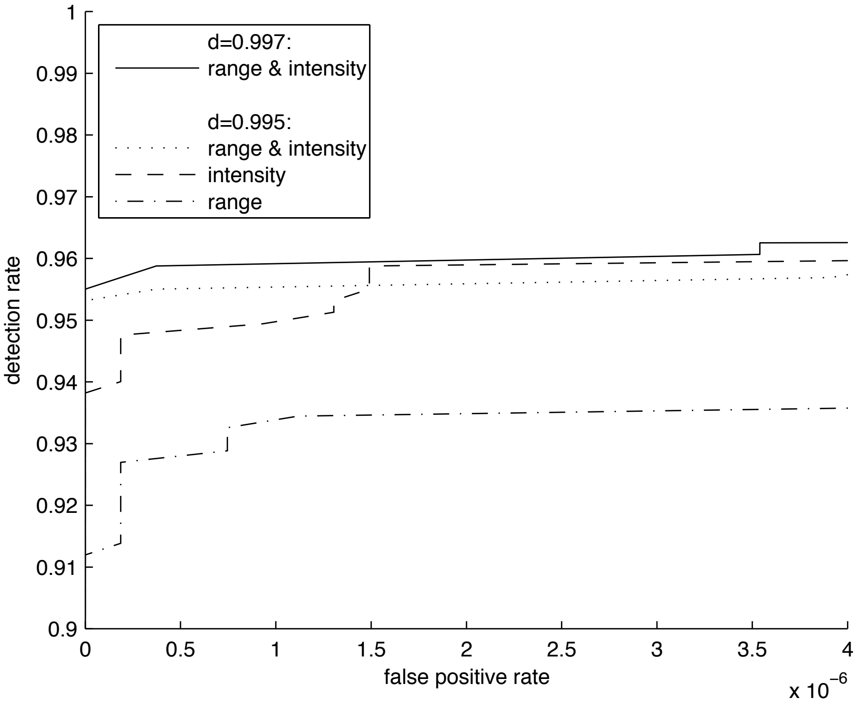


Figure 8.4: ROC curves for the detectors trained on the combined range and intensity data as well as both types of data separately.

error rates [132]. Whereas the MIT+CMU test set contains images from a variety of sources, including text and line drawings, our test set consists solely of images taken with a single camera. Also, because of the active illumination, the lighting is the same across all images. We believe these factors combine to make the test set "easier".

We will now examine the cascade structure of the detectors, i.e. the number of features used in each stage together with the detection rate and false-positive rate for each stage.

We first compare the three detectors with $d = 0.995$ (Table 8.1). The first thing that is noticeable is that the first stage of the range-and-intensity detector achieves a false-positive rate of zero on the validation set, i.e. the false-positive rate was too small to measure on the validation set. When this first stage was run on the set of full-frame test images, its false positive rate was 0.05%. In other words, the first stage already eliminates 99.95% of nonfaces. For comparison, the first stages of the other two detectors had false-positive rates of 18.9% (intensity) and 7.1% (range).

To understand why the first stage of the range-and-intensity detector has such good performance, consider Figure 8.5, which shows the features used by this stage: A vertical three-bar range feature and a horizontal three-bar intensity feature. From the sample training image underlayed under the features, it is evident that the range fea-

		detection rate		false-positive rate		number of features		
	Stage	individual	cumulative	individual	cumulative	total	intensity	range
range & intensity	1	1.000	1.000	0.000	1.0e-02	2	1	1
	2	0.995	0.995	0.165	1.6e-03	2	1	1
	3	0.995	0.991	0.138	2.3e-04	2	1	1
	4	0.995	0.986	0.394	8.9e-05	7	4	3
	5	0.995	0.981	0.290	2.6e-05	7	3	4
	6	0.995	0.976	0.385	1.0e-05	12	5	7
	7	0.995	0.971	0.334	3.3e-06	12	8	4
	8	0.995	0.966	0.381	1.3e-06	15	9	6
	9	0.995	0.961	0.380	4.8e-07	20	11	9
	10	0.995	0.956	0.380	1.8e-07	22	13	9
	11	0.995	0.951	0.368	6.8e-08	27	16	11
	12	0.995	0.946	0.365	2.5e-08	44	26	18
	13	0.995	0.941	0.391	9.6e-09	49	26	23
intensity	1	0.999	0.999	0.189	1.9e-01	3	3	0
	2	0.995	0.994	0.367	6.9e-02	15	15	0
	3	0.995	0.989	0.340	2.4e-02	11	11	0
	4	0.998	0.986	0.340	8.0e-03	6	6	0
	5	0.997	0.983	0.391	3.1e-03	10	10	0
	6	0.995	0.978	0.389	1.2e-03	17	17	0
	7	0.995	0.973	0.376	4.6e-04	25	25	0
	8	0.995	0.968	0.384	1.8e-04	23	23	0
	9	0.995	0.963	0.386	6.8e-05	40	40	0
	10	0.995	0.958	0.389	2.6e-05	59	59	0
	11	0.995	0.953	0.392	1.0e-05	62	62	0
	12	0.995	0.948	0.400	4.1e-06	107	107	0
	13	0.995	0.943	0.396	1.6e-06	198	198	0
	14	0.995	0.938	0.399	6.5e-07	117	117	0
	15	0.995	0.934	0.390	2.5e-07	223	223	0
range	1	0.997	0.997	0.071	7.1e-02	2	0	2
	2	0.995	0.992	0.285	2.0e-02	3	0	3
	3	1.000	0.992	0.325	6.5e-03	3	0	3
	4	0.995	0.987	0.373	2.4e-03	11	0	11
	5	0.995	0.983	0.271	6.6e-04	17	0	17
	6	0.995	0.978	0.374	2.5e-04	18	0	18
	7	0.996	0.974	0.381	9.4e-05	10	0	10
	8	0.995	0.969	0.388	3.6e-05	26	0	26
	9	0.995	0.964	0.377	1.4e-05	29	0	29
	10	0.995	0.959	0.396	5.4e-06	56	0	56
	11	0.995	0.954	0.370	2.0e-06	42	0	42
	12	0.995	0.949	0.381	7.7e-07	66	0	66
	13	0.995	0.944	0.396	3.0e-07	114	0	114
	14	0.995	0.940	0.382	1.2e-07	81	0	81
	15	0.995	0.935	0.377	4.4e-08	111	0	111
	16	0.995	0.930	0.380	1.7e-08	139	0	139

Table 8.1: Cascade structure of the detectors trained for a per-stage detection rate of $d = 0.995$.

		detection rate		false-positive rate		number of features		
	Stage	individual	cumulative	individual	cumulative	total	intensity	range
range & intensity	1	1.000	1.000	0.000	1.0e-02	2	1	1
	2	0.998	0.998	0.241	2.4e-03	3	1	2
	3	1.000	0.998	0.299	7.2e-04	3	2	1
	4	1.000	0.998	0.385	2.8e-04	5	4	1
	5	0.998	0.996	0.373	1.0e-04	6	3	3
	7	0.997	0.993	0.343	3.6e-05	7	4	3
	7	0.997	0.989	0.365	1.3e-05	8	3	5
	8	0.997	0.986	0.385	5.0e-06	18	10	8
	9	0.997	0.982	0.340	1.7e-06	15	8	7
	10	0.997	0.979	0.348	5.9e-07	24	13	11
	11	0.997	0.976	0.364	2.1e-07	21	11	10
	12	0.997	0.972	0.390	8.4e-08	26	16	10
	13	0.997	0.969	0.350	2.9e-08	35	20	15
	14	0.997	0.966	0.394	1.2e-08	66	34	32
	15	0.997	0.962	0.364	4.2e-09	48	29	19

Table 8.2: Cascade structure of the range-and-intensity detector trained for a per-stage detection rate of $d = 0.997$.

ture responds to the range difference between the face and the background on either side; the intensity feature seems to respond to the difference between the eye region (which is typically darker) and the forehead and cheeks above and below (which are typically lighter).

The fact that the first stage achieves a false-positive rate of zero on the validation set is problematic for computing the false-positive rate of the entire cascade, which is used during training to decide when the detector has reached its performance goal. To be able to compute an overall false-positive rate, we conservatively assumed the false-positive rate for this stage to be 0.01; this assumption is also used in the cumulative rates shown in the table. The assumed rate of 0.01 is probably quite conservative and only affects the overall false-positive rate computed during training, but not the selection of weak classifiers or the false-positive rates computed on the test set.

Turning to the number of features per stage, we note that, in most stages, the range-and-intensity detector requires noticeably fewer features to reach its target performance than the other two detectors. Also, note that the range-and-intensity detector uses an approximately equal number of range and intensity features in each stage (with a tendency to use slightly more intensity features in the later stages). This indicates that the range and intensity data contribute approximately the same amount of information to the face detection task.

Table 8.2 shows the cascade structure of the range-and-intensity detector with $d = 0.997$. We see that the detector uses slightly more features than the $d = 0.995$ detector, but the most obvious difference is that the $d = 0.997$ detector needs to accept higher

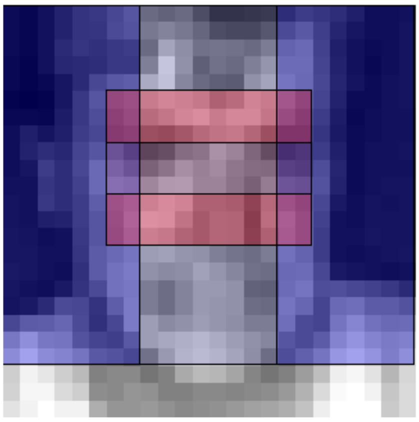


Figure 8.5: Features used by the first stage of the range-and-intensity detector. The blue (vertical) feature is a range feature; the red (horizontal) feature is an intensity feature.

false-positive rates in the early stages to meet its goal detection rate: $f_2 = 0.241$ and $f_3 = 0.299$ in stages 2 and 3 compared to $f_2 = 0.165$ and $f_3 = 0.138$ for the $d = 0.995$ detector. The effect of this is that the $d = 0.997$ detector requires two stages more than the $d = 0.995$ detector to achieve the target false-positive rate for the whole cascade.

Finally, we turn to the running times for the various detectors (Table 8.3); for comparison, the table also shows the detection rates achieved for a zero false-positive rate on the test set. The fastest detector is the range-and-intensity detector with $d = 0.995$. It is more than two times faster than the intensity-only detector and slightly faster than the range-only detector. This is despite the fact that the range-and-intensity detector needs to perform twice the amount of preprocessing since it computes the integral images for both range and intensity. The range-and-intensity detector with $d = 0.997$ has the best detection rate at the cost of a slightly higher running time than the $d = 0.995$ detector. Note that the running time for a detector is determined almost entirely by the number of features used in the first few stages; the latter stages are evaluated so rarely that they make almost no contribution to the running time.

8.3 Discussion

We have shown that, for the same parameter settings, a face detector trained on the combined range and intensity data from a TOF camera yields a higher detection rate (95.3%) than a detector trained on either type of data alone (intensity: 93.8%, range: 91.2%). Furthermore, the range-and-intensity detector requires fewer features than the other two detectors. This translates into faster running times: The range-and-intensity

	Detector	Detection rate	Running time per frame
$d = 0.997$	range & intensity	95.5%	5.54 ms
$d = 0.995$	range & intensity	95.3%	5.15 ms
	intensity	93.8%	10.69 ms
	range	91.2%	5.51 ms

Table 8.3: Performance summary of the detectors on the various types of data. Detection rates are given for a zero false-positive rate on the test set. Running times include preprocessing (computation of the integral images).

detector is over twice as fast as the intensity-only detector and slightly faster than the range-only detector (which misclassifies almost twice as many faces).

The data obtained by the TOF camera is in effect a two-channel image, where one channel contains the range map and the other contains the intensity image. If the TOF camera is combined with a grayscale or RGB camera operating in the visible spectrum (as is the case in the 3DV Systems ZCam [1], for instance, see Section 5.1 and Figure 5.1d), it would be straightforward to extend the method to the additional channels obtained in this way.

The detector we used was a "stock" Viola-Jones face detector. Even better results might be possible using features that are specifically tuned to the type of structures typically found in range images. One could also investigate the idea of combining range and intensity information in a single feature. Additionally, the many refinements that have been made to the Viola-Jones algorithm since its inception could be incorporated.

Though there is thus much room for future work on improving detection performance, we are here not primarily interested in the maximum absolute performance that a TOF face detector can achieve but rather in the relative difference in performance between face detection on combined range and intensity data versus either type of data alone. We believe that the advantage of the combined range and intensity detector in terms of robustness and speed should be preserved when refinements are made to the underlying algorithms; whether this indeed holds true is a question for future research.

Outlook

I have already discussed the results presented in this thesis individually at the end of the corresponding chapters. Now, I will try to take a wider view and give an outlook on where I believe the area of computer-vision-based interaction is headed.

Do we even need new types of interaction? The way we interact with computers has remained largely unchanged since the mouse and graphical user interface (GUI) were invented at the Stanford Research Institute and Xerox PARC in the 1960s and brought into widespread use by the Apple Macintosh in 1984. This can mean one of two things. One possible explanation is that, like the user-interface of the car, human-computer interaction has reached its logical endpoint, its optimum. But while it is plausible that we have found the ideal interface for the relatively simple task of driving, I find it hard to believe that the same is true for the computer, which, thanks to its programmability, is an almost infinitely malleable general-purpose tool, for which new uses are being found all the time and whose capabilities continue to grow at an astounding rate. I am therefore inclined to believe in the alternative explanation: That, under the surface of the GUI, the tectonic plates have been shifting, building up stresses and strains that, sooner or later, will be released in a major upheaval – one in which I believe computer vision will play an important part.

The gaming industry may already be feeling some of the tremors. After the advent of video games in the 1970s, game interaction saw a "quiet period" with little innovation in the style of interaction, similar to the situation in general-purpose computing. For three decades, players interacted with games using the joystick or its cousin, the joypad, as well as those old acquaintances from general-purpose computing, the keyboard and mouse. In recent years, however, gaming has seen something of an interaction revolution. The Sony EyeToy can be considered to mark the beginning of this development. Introduced in 2003, the EyeToy itself is simply a camera that is placed on the game display, facing the user. The games introduced with it, however, defined a new interaction style by sensing the player's movements in the camera image to provide hands-off, gesture-based game control. The game console Nintendo Wii bases its entire concept on novel forms of interaction. The primary controller for the Wii, called the Wii Remote, contains accelerometers for sensing the player's hand movements as well as a camera which, together with a strip of infrared LEDs placed near the display, allows the console to deduce the position and orientation of the controller. The NaturalPoint TrackIR is a head tracker for gaming which consists of a camera mounted on the display and infrared markers worn on the player's head. The player can move the head to change the viewpoint and viewing direction in the virtual world. Finally, I have already mentioned the 3DV Systems ZCam, a low-cost time-of-flight (TOF) camera aimed at the gaming market (see Chapter 5). The prototype camera developed within the ARTTS project, which is the smallest TOF camera yet produced and is powered entirely via the USB bus, advances the technology even further in the directions required for gaming.

Interestingly, all of these interaction innovations use computer vision in one form or another. Within the context of this thesis, the obvious question to ask is whether the technologies discussed here – eye tracking and time-of-flight action recognition and tracking – could add to the list of computer-vision-based interaction devices in gaming and, eventually, in general-purpose computing.

In the case of time-of-flight technology, it seems that we will soon know. 3DV Systems, the manufacturer of the ZCam TOF camera, was recently bought by Microsoft, and Microsoft has already presented ideas for TOF-based game interaction as part of its concept study Project Natal. As we have seen in this thesis, face detection and facial feature tracking work more robustly on TOF data than on conventional intensity images (see Chapters 7 and 8). If this is true also for other detection and tracking tasks (and I see no reason why it should not be), this robustness should make the TOF camera particularly suitable for consumer applications.

For eye tracking in games, on the other hand, price is still a hurdle – but the potential demand is definitely there. In 2006, several colleagues and I exhibited results from the project ModKog at the computer trade show CEBIT. One of the exhibits was a gaze-controlled game. After trying it, many visitors asked enthusiastically how much it would cost them to buy an eye tracker so they could play at home. Their enthusiasm was dampened when they heard the price. Recent work on low-cost eye trackers built from off-the-shelf hardware may, however, help to overcome this hurdle – and the automatic calibration technique described in Chapter 4 could help make eye trackers easy enough for consumers to use.

If this discussion has focused on gaming, it is not because I think that it is the only field where computer-vision-based interaction is viable – quite the contrary. I believe the success of these new interaction styles will spill over into general-purpose computing and indeed beyond to other types of devices. It is not unprecedented for gaming to trigger innovation in computing: modern graphics processing units (GPUs) are an example of this. Originally developed for the sole purpose of rendering ever more detailed and realistic game environments, modern GPUs are now being used as high-performance vector processors for general-purpose computing tasks. Time-of-flight technology and eye tracking may well take a similar course.

Bibliography

[1] 3DV ZCam, 3DV Systems, Yokne'am, Israel. http://www.3dvsystems.com.

[2] Alea IG-30 System, Alea Technologies GmbH, Teltow, Germany. http://www.alea-technologies.de.

[3] EU project ARTTS (Action Recognition and Tracking based on Time-of-Flight Sensors). http://www.artts.eu.

[4] David A. Atchison and George Smith. *Optics of the Human Eye*. Butterworth Heinemann, Oxford, UK, 2000.

[5] Shumeet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 753–760, 1993.

[6] Andre L. C. Barczak, Martin J. Johnson, and Chris H. Messom. Real-time computation of Haar-like features at generic angles for detection algorithms. *Research Letters in the Information and Mathematical Sciences*, 9:98–111, 2006.

[7] Erhardt Barth, Terry Caelli, and Christoph Zetzsche. Image encoding, labeling, and reconstruction from differential geometry. *CVGIP: Graphical Models and Image Processing*, 55(6):428–446, November 1993.

[8] Erhardt Barth, Michael Dorr, Martin Böhme, Karl R. Gegenfurtner, and Thomas Martinetz. Guiding the mind's eye: improving communication and vision by external control of the scanpath. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, *Human Vision and Electronic Imaging*, volume 6057 of *Proc. SPIE*, 2006. Invited contribution for a special session on Eye Movements, Visual Search, and Attention: a Tribute to Larry Stark.

[9] Jan Beirlant, Edward J. Dudewicz, László Györfi, and Edward Cornelis van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39, 1997.

[10] Paul J. Besl and Ramesh C. Jain. Invariant surface characteristics for 3D object recognition in range images. *Computer Vision, Graphics, and Image Processing*, 33:33–80, 1986.

[11] David Beymer and Myron Flickner. Eye gaze tracking using an active stereo head. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 451–458, 2003.

[12] Chistopher Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[13] Kenneth R. Boff and Janet E. Lincoln. *Engineering Data Compendium: Human Perception and Performance*. AAMRL, Wright-Patterson AFB, OH, 1988.

[14] Martin Böhme and Erhardt Barth. Challenges in Single-Camera Remote Eye-Tracking. In *1st Conference on Communication by Gaze Interaction (COGAIN), Copenhagen, Denmark*, 2005.

[15] Martin Böhme, Michael Dorr, Mathis Graw, Thomas Martinetz, and Erhardt Barth. A software framework for simulating eye trackers. In *Proceedings of Eye Tracking Research & Applications (ETRA)*, pages 251–258, 2008.

[16] Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. A facial feature tracker for human-computer interaction based on 3D Time-of-Flight cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):264–273, 2008.

[17] Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. Shading constraint improves accuracy of time-of-flight measurements. In *CVPR 2008 Workshop on Time-of-Flight-based Computer Vision (TOF-CV)*, 2008.

[18] Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. Shading constraint improves accuracy of time-of-flight measurements. *Computer Vision and Image Understanding*, 2009. (in revision).

[19] Martin Böhme, Martin Haker, Kolja Riemer, Thomas Martinetz, and Erhardt Barth. Face detection using a time-of-flight camera. In *Dynamic 3D Imaging – Workshop in Conjunction with DAGM*, volume 5742 of *Lecture Notes in Computer Science*, pages 167–176, 2009.

[20] Martin Böhme, André Meyer, Thomas Martinetz, and Erhardt Barth. Remote eye tracking: State of the art and directions for future development. In *The 2nd Conference on Communication by Gaze Interaction – COGAIN 2006, Turin, Italy*, pages 10–15, 2006.

[21] Xavier L. C. Brolly and Jeffrey B. Mulligan. Implicit calibration of a remote gaze tracker. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '04)*, volume 8, page 134, 2004.

[22] S. Charles Brubaker, Jianxin Wu, Jie Sun, Matthew D. Mullin, and James M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision*, 77(1–3):65–86, 2008.

[23] Andreas Bulling, Daniel Roggen, and Gerhard Tröster. Wearable EOG goggles: Seamless sensing and context-awareness in everyday environments. *Journal of Ambient Intelligence and Smart Environments*, 1(2):157–171, 2009.

[24] Bernhard Büttgen, Thierry Oggier, Michael Lehmann, Rolf Kaufmann, and Felix Lustenberger. CCD/CMOS lock-in pixel for range imaging: Challenges, limitations, and state-of-the-art. In *1^{st} Range Imaging Research Day*, pages 21–32, ETH Zürich, Switzerland, 2005.

[25] Canesta DP200 Electronic Perception Technology Development Platform, Canesta, Inc., Sunnyvale, CA, USA. `http://www.canesta.com`.

[26] Jacques R. Charlier, Maurice Behague, and Cathy Buquet. Shift of the pupil center with pupil constriction. *Investigative Ophthalmology and Visual Science*, 35(4):1278, 1994.

[27] COGAIN. Network of Excellence on Communication by Gaze Interaction. `http://www.cogain.org`.

[28] COGAIN. D3.1 User requirements report with observations of difficulties users are experiencing. Technical report, European Union Network of Excellence COGAIN (contract no. IST-2003-511598) of the 6th Framework Programme, 2005. `http://www.cogain.org/results/reports/COGAIN-D3.1.pdf`.

[29] COGAIN. D3.2 Report on features of the different systems and development needs. Technical report, European Union Network of Excellence COGAIN (contract no. IST-2003-511598) of the 6th Framework Programme, 2006. `http://www.cogain.org/results/reports/COGAIN-D3.2.pdf`.

[30] Tom N. Cornsweet and Hewitt D. Crane. Accurate two-dimensional eye tracker using first and fourth Purkinje images. *Journal of the Optical Society of America*, 63(8):921–928, 1973.

[31] James Edwin Cryer, Ping-Sing Tsai, and Mubarak Shah. Integration of shape from shading and stereo. *Pattern Recognition*, 28(7):1033–1043, 1995.

[32] Edmund Burke Delabarre. A method of recording eye-movements. *American Journal of Psychology*, 9(4):572–574, 1898.

[33] James R. Diebel and Sebastian Thrun. An application of Markov random fields to range sensing. In *Advances in Neural Information Processing Systems 18*, pages 291–298, 2006.

[34] James R. Diebel, Sebastian Thrun, and Michael Brünig. A Bayesian method for probable surface reconstruction and decimation. *ACM Transactions on Graphics*, 25(1):39–59, 2006.

[35] Raymond Dodge and Thomas Sparks Cline. The angle velocity of eye movements. *Psychological Review*, 8(2):145–157, 1901.

[36] Michael Dorr, Laura Pomarjanschi, and Erhardt Barth. Gaze beats mouse: A case study on a gaze-controlled Breakout. *PsychNology*, 7(2):197–211, 2009.

[37] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Springer, New York, 2003.

[38] Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109(1):22–43, 2008.

[39] Jean-Denis Durou and Didier Piau. Ambiguous shape from shading with critical points. *Journal of Mathematical Imaging and Vision*, 12(2):99–108, 2000.

[40] Epitex Incorporation, Kyoto, Japan. `http://www.epitex.com`.

[41] Thor Eysteinsson, Fridbert Jonasson, Hiroshi Sasaki, Arsaell Arnarsson, Thordur Sverrisson, Kazuyuki Sasaki, Einar Stefánsson, and the Reykjavik Eye Study Group. Central corneal thickness, radius of the corneal curvature and intraocular pressure in normal subjects using non-contact techniques: Reykjavik eye study. *Acta Ophthalmologica Scandinavica*, 80:11–15, 2002.

[42] Ian Fasel, Bret Fortenberry, and Javier Movellan. A generative framework for real time object detection and classification. *Computer Vision and Image Understanding*, 98(1):182–210, 2005.

[43] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[44] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[45] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[46] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[47] Pascal V. Fua and Yvan G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56, 1995.

[48] Wilson S. Geisler and Jeffrey S. Perry. A real-time foveated multiresolution system for low-bandwidth video communication. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *Human Vision and Electronic Imaging: SPIE Proceedings*, volume 3299 of *Proc. SPIE*, pages 294–305. 1998.

[49] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, second edition, 2001.

[50] Elias Daniel Guestrin and Moshe Eizenman. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on Biomedical Engineering*, 53(6):1124–1133, 2006.

[51] Tom S. F. Haines and Richard C. Wilson. Integrating stereo with shape-from-shading derived orientation information. In *British Machine Vision Conference*, volume 2, pages 910–919, 2007.

[52] Tom S. F. Haines and Richard C. Wilson. Combining shape-from-shading and stereo using Gaussian-Markov random fields. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[53] Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth. Geometric invariants for facial feature tracking with 3D TOF cameras. In *Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 109–112, Iasi, Romania, 2007.

[54] Martin Haker, Martin Böhme, Thomas Martinetz, and Erhardt Barth. Scale-invariant range features for time-of-flight camera applications. In *CVPR 2008 Workshop on Time-of-Flight-based Computer Vision (TOF-CV)*, 2008.

[55] Radim Halíř and Jan Flusser. Numerically stable direct least squares fitting of ellipses. In *Proceedings of the 6th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG'98)*, volume 1, pages 125–132, 1998.

[56] Dan Witzner Hansen. *Committing Eye Tracking*. PhD thesis, The IT University of Copenhagen, Denmark, 2004.

[57] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. (to appear).

[58] Dan Witzner Hansen, Rasmus Larsen, and François Lauze. Improving face detection with TOF cameras. In *Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 225–228, 2007.

[59] Dan Witzner Hansen and Arthur E. C. Pece. Eye tracking in the wild. *Computer Vision and Image Understanding*, 98(1):155–181, 2005.

[60] John Paulin Hansen, Allan W. Andersen, and Peter Roed. Eye-gaze control of multimedia systems. In *Proceedings of the 6th International Conference on Human Computer Interaction*, pages 37–42, 1995.

[61] Keith Hartt and Mark Carlotto. A method for shape-from-shading using multiple images acquired under different viewing and lighting conditions. In *Proceedings of Computer Vision and Pattern Recognition*, pages 53–60, 1989.

[62] Werner Haustein. Considerations on Listing's Law and the primary position by means of a matrix description of eye position control. *Biological Cybernetics*, 60(6):411–420, 1989.

[63] Horst G. Heinol. *Untersuchung und Entwicklung von modulationslaufzeitbasierten 3D-Sichtsystemen*. PhD thesis, University of Siegen, Germany, 2001.

[64] Heliopan Lichtfilter-Technik Summer GmbH & Co KG, Munich, Germany. `http://www.heliopan.de`.

[65] Craig Hennessey, Borna Noureddin, and Peter Lawrence. A single camera eye-gaze tracking system with free head motion. In *Proceedings of Eye Tracking Research & Applications (ETRA)*, pages 87–94, 2006.

[66] Dennis Howitt and Duncan Cramer. *Introduction to Research Methods in Psychology*. Pearson / Prentice Hall, Harlow, New York, 2005.

[67] Edmund Burke Huey. Preliminary experiments in the physiology and psychology of reading. *American Journal of Psychology*, 9(4):575–586, 1898.

[68] Thomas E. Hutchinson, K. Preston White, Jr., Worthy N. Martin, Kelly C. Reichert, and Lisa A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1527–1533, 1989.

[69] Gabi J. Iddan and Giora Yahav. 3D imaging in the studio. In *Proceedings of SPIE*, volume 4298, pages 48–56, 2001.

[70] ifm efector pmd, ifm electronic GmbH, Essen, Germany. `http://www.ifm-electronic.com`.

[71] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[72] Robert J. K. Jacob. The use of eye movements in human-computer interaction techniques: What you look at is what you get. *ACM Transactions on Information Systems*, 9(3):152–169, 1991.

[73] Bernd Jähne, Horst Haußecker, and Peter Geißler, editors. *Handbook of Computer Vision and Applications*. Academic Press, San Diego, USA, 1999.

[74] Jeremy Yermiyahou Kaminski, Mina Teicher, Dotan Knaan, and Adi Shavit. Three-dimensional face orientation and gaze detection from a single image. 2004.

[75] Jens Keiner, Stefan Kunis, and Daniel Potts. Using NFFT 3 – a software library for various nonequispaced fast Fourier transforms. *ACM Transactions on Mathematical Software*, 36(4), 2009. (to appear).

[76] Wolf Kienzle, Bernhard Schölkopf, Felix A. Wichmann, and Matthias O. Franz. How to find interesting locations in video: a spatiotemporal interest point detector learned from human eye movements. In *Proceedings of the 29th Annual Symposium of the German Association for Pattern Recognition (DAGM 2007)*, pages 405–414, Berlin, Germany, 2007. Springer Verlag.

[77] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-flight cameras in computer graphics. *Computer Graphics Forum*. (to appear).

[78] Stefan Kunis and Daniel Potts. Stability results for scattered data interpolation by trigonometric polynomials. *SIAM Journal on Scientific Computing*, 29:1403–1419, 2007.

[79] Robert Lange. *3D Time-of-Flight Distance Measurement with Custom Solid-State Sensors in CMOS/CCD-Technology*. PhD thesis, University of Siegen, Germany, 2000.

[80] Yvan G. Leclerc and Aaron F. Bobick. The direct computation of height from shading. In *Computer Vision and Pattern Recognition (CVPR '91)*, pages 552–558, 1991.

[81] R. John Leigh and David S. Zee. *The Neurology of Eye Movements*. Oxford University Press, fourth edition, 2006.

[82] Lexus LS driver monitoring system. http://www.lexus.co.uk/range/ls/key-features/safety/safety-driver-monitoring-system.aspx. (Accessed on 20 May 2008).

[83] Dongheng Li, David Winfield, and Derrick J. Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Proceedings of the IEEE Vision for Human-Computer Interaction Workshop at CVPR*, pages 1–8, 2005.

[84] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Proceedings of the 25th Annual Symposium of the German Association for Pattern Recognition (DAGM)*, pages 297–304, 2003.

[85] Jin Liu and Siegmund Pastoor. Computer-aided video-based method for contactlessly determining the direction of view pertaining to an eye of a user for the eye-guided interaction between human beings and computers and a device for carrying out said method, 2001. World Patent WO0133323.

[86] Lumenera Corp., Ottawa, Canada. http://www.lumenera.com.

[87] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, 1979.

[88] George W. McConkie and Keith Rayner. The span of the effective stimulus during a fixation in reading. *Perception & Psychophysics*, 17:578–586, 1975.

[89] John Merchant, Richard Morrissette, and James L. Porterfield. Remote measurement of eye direction allowing subject motion over one cubic foot of space. *IEEE Transactions on Biomedical Engineering*, 21(4):309–317, 1974.

[90] MESA SR4000, MESA Imaging AG, Zürich, Switzerland. http://www.mesa-imaging.ch.

[91] André Meyer. Single-camera remote eye tracking. Diploma thesis, Universität zu Lübeck, 2006.

[92] André Meyer, Martin Böhme, Thomas Martinetz, and Erhardt Barth. A single-camera remote eye tracker. In *Perception and Interactive Technologies*, volume 4021 of *Lecture Notes in Artificial Intelligence*, pages 208–211. Springer, 2006.

[93] I. L. Meyers. Electronystagmography: A graphic study of the action currents in nystagmus. *Archives of Neurology and Psychiatry*, 21:901–908, 1929.

[94] Carlos H. Morimoto, Arnon Amir, and Myron Flickner. Detecting eye position and gaze from a single camera and 2 light sources. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 4, pages 314–317, 2002.

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

[96] Mostafa G.-H. Mostafa, Sameh M. Yamany, and Aly A. Farag. Integrating shape from shading and range data using neural networks. In *Computer Vision and Pattern Recognition (CVPR '99)*, volume 2, page 2015, 1999.

[97] Cicero Mota and Erhardt Barth. On the uniqueness of curvature features. In *Dynamische Perzeption*, volume 9 of *Proceedings in Artificial Intelligence*, pages 175–178, 2000.

[98] Sateesha G. Nadabar and Anil K. Jain. Fusion of range and intensity images on a Connection Machine (CM-2). *Pattern Recognition*, 28(1):11–26, 1995.

[99] Donald A. Norman. *The Design of Everyday Things*. Doubleday, New York, 1990.

[100] Thierry Oggier, Bernhard Büttgen, Felix Lustenberger, Guido Becker, Björn Rüegg, and Agathe Hodac. SwissRanger™ SR3000 and first experiences based on miniaturized 3D-TOF cameras. In *Proceedings of the 1st Range Imaging Research Day*, pages 97–108, Zürich, Switzerland, 2005.

[101] Takehiko Ohno and Naoki Mukawa. A free-head, simple calibration, gaze tracking system that enables gaze-based interaction. In *Eye Tracking Research and Applications (ETRA)*, pages 115–122, 2004.

[102] Takehiko Ohno, Naoki Mukawa, and Atsushi Yoshikawa. Freegaze: A gaze tracking system for everyday gaze interaction. In *Eye Tracking Research and Applications (ETRA)*, pages 125–132, 2002.

[103] Open Computer Vision Library (OpenCV). `http://www.sourceforge.net/projects/opencvlibrary`.

[104] Martin T. Orne. On the social psychology of the psychological experiment: With particular reference to demand characteristics and their implications. *American Psychologist*, 17:776–783, 1962.

[105] Pentax Europe GmbH, Hamburg, Germany. `http://www.pentax.de`.

[106] Alex Pentland, Baback Moghaddam, and Thad Starner. View-based and modular eigenspaces for face recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 84–91, 1994.

[107] Antonio Pérez, Maria Luisa Córdoba, Antonio García, Rafael Méndez, Luisa Mu noz, José Luis Pedraza, and Francisco Sánchez. A precise eye-gaze detection and tracking system. In *Proceedings of the 11th International Conference in Central Europe of Computer Graphics, Visualization and Computer Vision*, Plzen, Czech Republic, 2003.

[108] Maurizio Pilu. Self-calibration for an eye tracker, 2005. United States Patent Application US 2005/0225723 A1.

[109] PMD[vision] CamCube, PMD Technologies, Siegen, Germany. `http://www.pmdtec.com`.

[110] Daniel Potts, Gabriele Steidl, and Manfred Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In John J. Benedetto and Paulo J. S. G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 247–270. Birkhäuser, Boston, 2001.

[111] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, second edition, 1992.

[112] Daniel C. Richardson and Michael J. Spivey. Eye-tracking: Characteristics and methods. In Gary E. Wnek and Gary L. Bowlin, editors, *Encyclopedia of Biomaterials and Biomedical Engineering*, pages 568–572. Marcel Dekker, Inc., 2004.

[113] Kolja Riemer. Gesichtsdetektion mit Time-of-Flight-Kameras. Diploma thesis, Universität zu Lübeck, 2008.

[114] David A. Robinson. A method of measuring eye movement using a scleral search coil in a magnetic field. *IEEE Transactions on Biomedical Engineering*, 10:137–145, 1963.

[115] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural-network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[116] Dimitrios Samaras, Dimitris Metaxas, Pascal V. Fua, and Yvan G. Leclerc. Variable albedo surface reconstruction from stereo and shape from shading. In *Proceedings of Computer Vision and Pattern Recognition*, volume 1, pages 480–487, 2000.

[117] Erich Schneider and Thomas Eggert. Methoden der Augenbewegungsmessung. Invited talk at the convention of the Deutsche Gesellschaft für klinische Neurophysiologie und funktionelle Bildgebung, 2006. `http://www.forbias.de/papers/dgkn06.pdf`.

[118] E. Schott. Über die Registrierung des Nystagmus und anderer Augenbewegungen vermittels des Seitengalvanometers. *Deutsches Archiv für Klinische Medizin*, 140:79–90, 1922.

[119] Rudolf Schwarte, Horst G. Heinol, Zhanping Xu, and Klaus Hartmann. New active 3D vision system basd on rf-modulation interferometry of incoherent light. In *Intelligent Robots and Computer Vision XIV*, volume 2588 of *Proceedings of SPIE*, pages 126–134, 1995.

[120] Sheng-Wen Shih, Yu-Te Wu, and Jin Liu. A calibration-free gaze tracking technique. In *Proceedings of the 15th International Conference on Pattern Recognition*, pages 201–204, 2000.

[121] SMI iView X Hi-Speed 1250, SensoMotoric Instruments GmbH, Teltow, Germany. `http://www.smivision.com`.

[122] SMI iView X RED, SensoMotoric Instruments GmbH, Teltow, Germany. `http://www.smivision.com`.

[123] Susan Standring. *Gray's Anatomy: The Anatomical Basis of Clinical Practice*. Elsevier Churchill Livingstone, Edinburgh, New York, 39th edition, 2004.

[124] Andreas Straube and Ulrich Büttner. *Neuro-Ophthalmology: Neuronal Control of Eye Movements*. Karger, Basel, Switzerland, 2007.

[125] Lawrence A. Symons, Kang Lee, Caroline C. Cedrone, and Mayu Nishimura. What are you looking at? Acuity for triadic eye gaze. *The Journal of General Psychology*, 131(4):451–469, 2004.

[126] Clay Matthew Thompson. Robust photo-topography by fusing shape-from-shading and stereo. AI Technical Report 1411, Massachusetts Institute of Technology, 1993.

[127] Tobii 1750 eye tracker, Tobii Technology AB, Stockholm, Sweden.

[128] Tobii Technology AB, Stockholm, Sweden. *Product Description ClearView 2.7 Eye gaze analysis software*, 2006.

[129] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[130] Eleonora Vig, Michael Dorr, and Erhardt Barth. Efficient visual coding and the predictability of eye movements on natural movies. *Spatial Vision*, 22(5):397–408, 2009.

[131] Arantxa Villanueva, Gintautas Daunys, Dan Witzner Hansen, Martin Böhme, Rafael Cabeza, André Meyer, and Erhardt Barth. A geometric approach to remote eye tracking. *Universal Access in the Information Society*, 2009. (to appear).

[132] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[133] Hermann von Helmholtz. *Handbuch der physiologischen Optik*. Voss, Hamburg, Leipzig, 3rd edition, 1910.

[134] Nicholas Wade and Benjamin W. Tatler. *The Moving Tablet of the Eye: The Origins of Modern Eye Movement Reserach*. Oxford University Press, 2005.

[135] Peng Wang, Matthew B. Green, Qiang Ji, and James Wayman. Automatic eye detection and its validation. In *IEEE Conference on Computer Vision & Pattern Recognition*, page 164, 2005.

[136] David J. Ward and David J. C. MacKay. Fast hands-free writing by gaze direction. *Nature*, 418(6900):838, 2002.

[137] Bryn Wolfe and David Eichmann. A neural network approach to tracking eye position. *International Journal of Human-Computer Interaction*, 9(1):59–79, 1997.

[138] Giora Yahav, Gabi J. Iddan, and David Mandelboum. 3D imaging camera for gaming application. In *International Conference on Consumer Electronics*, pages 1–2, 2007.

[139] Alfred L. Yarbus. *Eye Movements and Vision*. Plenum Press, New York, 1967.

[140] Dong Hyun Yoo and Myung Jin Chung. A novel non-instrusive eye gaze estimation using cross-ratio under large head motion. *Computer Vision and Image Understanding*, 98:25–51, 2005.

[141] Laurence R. Young. Recording eye position. In Manfred Clynes and John H. Milsum, editors, *Biomedical Engineering Systems*. McGraw-Hill, New York, 1970.

[142] Laurence R. Young and David Sheena. Survey of eye movement recording methods. *Behavior Research Methods & Instrumentation*, 7(5):397–439, 1975.

[143] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[144] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

Index