A Software Framework for Simulating Eye Trackers

Martin Böhme* Michael Dorr† Mathis Graw‡ Thomas Martinetz§ Erhardt Barth¶

Institute for Neuro- and Bioinformatics
University of Lübeck, Germany

Abstract

We describe an open-source software framework that simulates the measurements made using one or several cameras in a video-oculographic eye tracker. The framework can be used to compare objectively the performance of different eye tracking setups (number and placement of cameras and light sources) and gaze estimation algorithms. We demonstrate the utility of the framework by using it to compare two remote eye tracking methods, one using a single camera, the other using two cameras.

CR Categories: I.4.9 [Image Processing and Computer Vision]: Applications; J.4 [Social and Behavioral Sciences]: Psychology

Keywords: eye tracking, gaze estimation, optical simulation

1 Introduction

Research on eye tracking has brought forth a host of different techniques [Duchowski 2003]. Even if we restrict ourselves to video-oculography, there are still many different approaches, which differ in terms of the number and placement of cameras and light sources, the image processing algorithms employed to find various features in the image, and the gaze estimation algorithms used to determine the point of regard or line of sight from these features (see e.g. Morimoto and Mimica [2005] for a review).

Eye tracking methods described in the literature usually report the average accuracy of the system. However, it can be difficult to determine why system A performs better than system B: Is it due to a superior hardware setup (more light sources, more cameras, higher image resolution or quality)? Is it due to more accurate image processing? Is it due to a better gaze estimation algorithm or more extensive calibration? Or is it simply due to the test procedure – maybe the users of system B moved their heads more or covered a greater range of gaze angles than those of system A?

If two systems have similar accuracy, is this because all of their respective components perform equally well? Or could we obtain a superior system by combining, say, the image processing of system A with the gaze estimation of system B?

Say we have an existing system and wish to explore ways of improving its performance – should we maybe add more light sources? Or simply change the position of the existing light sources?

*e-mail: boehme@inb.uni-luebeck.de
†e-mail: dorr@inb.uni-luebeck.de
‡e-mail: grow@informatik.uni-luebeck.de
§e-mail: martinetz@inb.uni-luebeck.de
¶e-mail: barth@inb.uni-luebeck.de

http://doi.acm.org/10.1145/1344471.1344529

In this paper, we argue that simulation is an ideal tool for answering these and similar questions. While simulation has been used before in eye tracking (see e.g. Shih, Wu and Liu [2000]), it has usually had the role of a stopgap that is used to publish results before one has been able to build the actual hardware. We believe simulation is not just a second-class substitute for hardware but that it can be a powerful tool for answering questions, such as those raised above, that are difficult or impossible to investigate in hardware.

We will present a simulation framework based on an optical model of the eye (see Section 2), which is used 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. Standardized tests over a defined range of head positions and gaze targets can be run, and comparisons can be made with other gaze estimation algorithms or hardware setups.

The framework is written in MATLAB (`www.mathworks.com`), an interpreted language for numerical computing that facilitates experimentation and allows gaze estimation algorithms to be expressed concisely. The source code is freely available at our website, and we hope to encourage other researchers to use the framework in their work and to expand it to suit their needs. For details on obtaining and using the framework, see Section 3.

We demonstrate the simulation framework by using it to implement two approaches to remote eye tracking: a single-camera and a two-camera approach (see Section 4). We investigate the two approaches individually to see how their accuracy is influenced by factors such as camera noise or recalibration procedures as well as comparing the two approaches (Section 5). To our knowledge, such an analysis and comparison of different eye tracking methods by simulation has not so far been published in the literature.

2 What the Framework Models

Most videographic 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, we directly compute the positions where these features will lie in the camera image given the spatial positions of the eye, camera, and lights. We simulate the effect of finite camera resolution and of inaccuracies in the image analysis algorithms 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

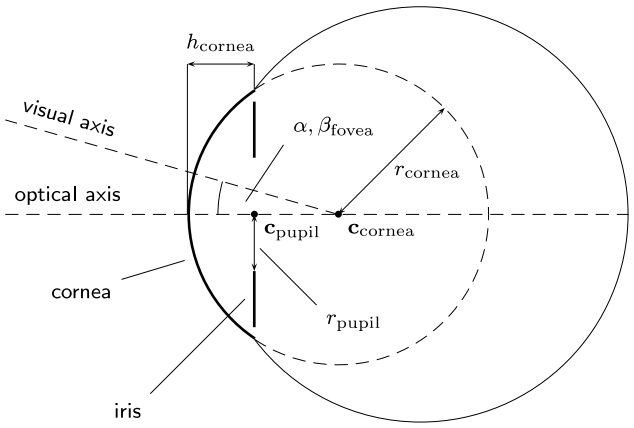


Figure 1: *Eye model used in the simulation framework. The meaning of the various parameters is explained in Section 2.1.*

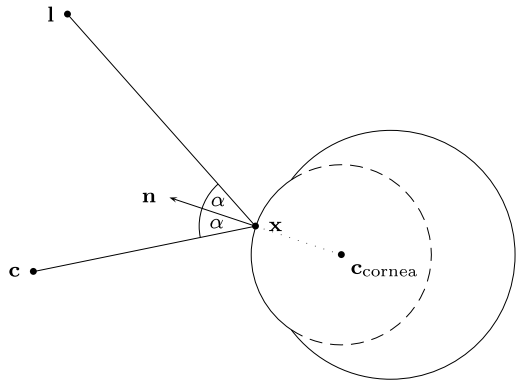


Figure 2: *Reflection of a ray from the light source* $\mathbf{l}$ *at a point* $\mathbf{x}$ *on the surface of the cornea towards the camera* $\mathbf{c}$*.* $\mathbf{n}$*: surface normal;* $\mathbf{c}_{\mathrm{cornea}}$*: centre of corneal curvature.*

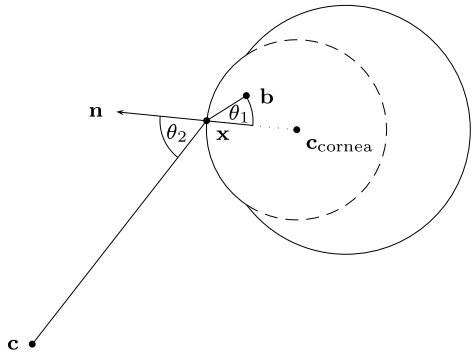


Figure 3: *Refraction of a ray from the pupil boundary point* $\mathbf{b}$ *at a point* $\mathbf{x}$ *on the surface of the cornea towards the camera* $\mathbf{c}$*.* $\mathbf{n}$*: surface normal;* $\mathbf{c}_{\mathrm{cornea}}$*: centre of corneal curvature;* θ_1, θ_2*: angles of incident and refracted ray with the surface normal.*

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 Eye

To specify the relative positions of the components of the eye, we use an eye coordinate system, whose origin lies at the centre of rotation of the eyeball. Within this coordinate system, we model the following components:

Cornea This is modelled as a spherical cap with a radius of r_{cornea}, a centre of curvature $\mathbf{c}_{\mathrm{cornea}}$ lying on the optical axis of the eye, and a cap height of h_{cornea} (see Figure 1). (The numerical values for these parameters and others that follow are taken from the standard eye in Boff and Lincoln [1988, Section 1.210].)

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 so-called *corneal reflexes* (CRs) or *glints* – are observed. Reflection at the surface of the cornea follows the law of reflection (angle of incidence equals angle of reflection [Forsyth and Ponce 2002], see Figure 2):

$$\frac{\mathbf{c}-\mathbf{x}}{\|\mathbf{c}-\mathbf{x}\|_2}\cdot\mathbf{n} = \frac{\mathbf{l}-\mathbf{x}}{\|\mathbf{l}-\mathbf{x}\|_2}\cdot\mathbf{n}$$

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

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

After the point of reflection $\mathbf{x}$ 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 3). This is governed by Snell's law [Forsyth and Ponce 2002]:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2,$$

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 $\mathbf{b}$ on the pupil border, we wish to find the location $\mathbf{x}$ on the corneal surface where an incident ray from $\mathbf{b}$ is refracted in such a way that it passes into the camera at $\mathbf{c}$. Similar to the case of reflection, we note that $\mathbf{c}$, $\mathbf{b}$, $\mathbf{c}_{\mathrm{cornea}}$ and $\mathbf{x}$ are coplanar and that $\mathbf{x}$ must lie on the half-sphere facing $\mathbf{c}$, giving a unique solution for $\mathbf{x}$. Again, we use a one-dimensional root finder to find $\mathbf{x}$; 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 $\mathbf{c}_{\mathrm{pupil}}$ on the optical axis.

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 $\mathbf{c}_{\text{cornea}}$; this is sufficiently accurate for our purposes.

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 [von Helmholtz 1910; Haustein 1989].

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:

$$\mathbf{v}_0 \cdot \mathbf{w} = \mathbf{v}_1 \cdot \mathbf{w} = 0,$$

where $\mathbf{v}_0$ is the visual axis in the primary position, $\mathbf{v}_1$ is the visual axis in the rotated position, and $\mathbf{w}$ is the direction of the axis around which the rotation takes place.

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 our 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 [Atchison and Smith 2000]; in spite of this, we have used a spherical model in our current implementation because it simplifies the optical calculations. However, users are cautioned 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. [Beymer and Flickner 2003]).

Refraction at the posterior cornea surface The cornea and the aqueous humour have slightly different refractive indices (Boff and Lincoln [1988] 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. We do 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* [Straube and Büttner 2007]). For this reason, the framework does not currently model eyelids.

Pupil shape The pupil is not precisely circular in shape; furthermore, when the pupil contracts or expands, its shape may change, and its centre may shift [Charlier et al. 1994]. None of these effects are modelled in the simulation framework.

2.2 Cameras

The framework models cameras using the pinhole camera model [Forsyth and Ponce 2002]. Each 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. Trucco and Verri [1998]), we measure the focal length in pixels.

Pan-and-tilt cameras are simulated; however, camera movement happens instantaneously, i.e. we do 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 [1998] 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. Forsyth and Ponce [2002]). 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.

2.3 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.

3 Using the Framework

The MATLAB source code for the simulation framework can be downloaded at `http://www.inb.uni-luebeck.de/tools-demos/et_simul.zip`. The framework comes with comprehensive documentation, so we will only briefly demonstrate its use here.

```
function et = interpolate_calib(et, calib_data)
    % Calculate pupil-CR-vector for each calib. 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 4: *Implementation of a simple eye tracker with bilinear interpolation.*

The framework uses a pseudo-object-oriented style, i.e. MATLAB structures are used to represent the various entities in the system (eyes, lights, cameras), and methods are implemented as functions that take the object they operate on as their first argument. The name of a method starts with the name of the class of objects it operates on (e.g. `eye_look_at()`); methods that create objects ("constructors") have the suffix `_make` (e.g. `eye_make()`).

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.

As an example, Figure 4 shows the calibration and evaluation function for a very simple eye tracker with one camera and one light source that uses the pupil-CR difference vector technique (see e.g. Morimoto and Mimica [2005]) with a bilinear interpolation function similar to the biquadratic function used for recalibration in Section 4.4. Note how the built-in matrix and vector features of MATLAB allow the algorithm to be expressed concisely.

4 Simulated Methods

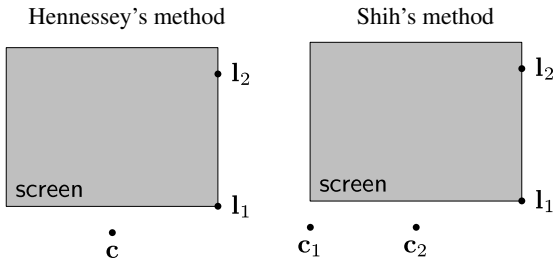
We will use the simulation framework to simulate and compare two different approaches to remote eye tracking. The first is a single-camera method by Hennessey, Noureddin, and Lawrence [2006], and the second is a multi-camera method by Shih, Wu, and Liu [2000]; for brevity, we will refer to these methods as "Hennessey's method" and "Shih's method".

Unfortunately, we do not have the space to describe both methods in detail, so we will only give the general principles; the reader is referred to the original publications for a full description.

4.1 Hardware setup

One advantage of using a simulation is that we can use the same hardware setup for both methods (except, of course, for the number of cameras). This eliminates any influence that the setup might have on the accuracy of the eye tracking.

The hardware setup is shown in Figure 5. We simulate a typical human-computer interaction setup, where we wish to measure gaze positions on a computer screen. The screen has a size of 40x30 cm, corresponding approximately to the usable area of a 19-inch LCD display. Two light sources are mounted on one of the vertical edges of the screen (this is the setup used by Hennessey et al.). Depending on the eye tracking method, one or two cameras are mounted below the screen; each camera has a resolution of 1280x1024 pixels and a focal length of 2000 pixels. This yields a wider field of view than the cameras used in the two original papers; we made this modification because it allows us to test the head movement tolerance of the systems over a greater range of head positions.



System component	Coordinates
Screen	$x = -0.2 \ldots 0.2$, $y = 0.05 \ldots 0.35$, $z = 0$
Lights	$\mathbf{l}_1 = (0.2, 0.05, 0)$ $\mathbf{l}_2 = (0.2, 0.3, 0)$
Camera (Hennessey's method)	$\mathbf{c} = (0, 0, 0)$
Cameras (Shih's method)	$\mathbf{c}_1 = (-0.2, 0, 0)$ $\mathbf{c}_2 = (0, 0, 0)$
Eye position volume	$(0, 0.35, 0.55) \pm (\frac{w}{2}, \frac{h}{2}, \frac{d}{2})$ $w = 0.25, h = 0.2, d = 0.2$

Figure 5: *Hardware setup and positions of system components (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.*

Note the asymmetric camera placement for Shih's method. This is to ensure that the CRs remain visible across the whole range of head positions and gaze directions. We will discuss this issue in more detail in Section 5.

We define an *eye position volume*, i.e. an area within which we wish to be able to track a user's eye. This volume is box-shaped and is centred around a point at a perpendicular distance of 55 cm from the top centre of the monitor. The dimensions (width, height, and depth) of the volume are 25x20x20 cm. Assuming an interocular distance of 7 cm, this eye position volume allows head movements of 18x20x20 cm if we wish to track both eyes. (However, all results presented in the following are for a single eye only.) All simulated cameras are oriented with their optical axes pointing towards the centre of the eye position volume.

The table in Figure 5 gives the precise positions of the individual system components. All positions are given in a right-handed coordinate system that has its origin below the centre of the screen, with the screen lying in the x-y-plane.

4.2 Hennessey's method

Hennessey's method uses a single camera and two or more light sources. From the position of the CRs in the image, the method computes the position of the corneal sphere in space; to obtain a unique solution, the radius of corneal curvature needs to be known, for which a population average is used. Then, for a number of pupil contour points in the image, the method traces the corresponding light rays back into space and refracts them at the corneal surface; these refracted rays are then used to find the pupil centre by exploiting the additional criterion that all points on the pupil contour are a distance of $r_{\mathrm{ps}} = \sqrt{r_{\mathrm{d}}^2 + r_{\mathrm{p}}^2}$ from the corneal centre, where r_{d} is the distance between the corneal centre and the pupil centre (set to a population average), and r_{p} is the radius of the pupil (obtained using the major axis of an ellipse fit to the pupil image, the distance of the eye from the camera, and the camera model). The corneal centre and pupil centre determine the optical axis of the eye, which is intersected with the screen to find the point of regard. A recalibration procedure (see Section 4.4) is then used to correct for any remaining errors (due for example to the offset between the optical and visual axes or the use of population averages for the parameters of the eye).

4.3 Shih's method

Shih's method uses two or more cameras and two or more light sources. It exploits two geometrical properties of the multi-camera setting to determine the direction of the optical axis without needing any subject-dependent eye parameters. These two properties are: (i) For each camera, the position of the CRs in the image constrains the possible positions of the cornea centre to a line in space. By intersecting the lines obtained for two or more cameras, the exact position of the cornea centre in space can be determined. (ii) For each camera, the position of the cornea centre in space (now known) together with the position of the pupil centre in the image constrains the possible positions of the pupil in space to a plane. Intersecting the planes obtained for two or more cameras yields a line; because each of the planes also contained the corneal centre, this line is the optical axis of the eye. As for Hennessey's method, we intersect with the screen to determine the point of regard, then use a recalibration procedure to correct for any remaining errors.

4.4 Recalibration procedures

As noted above, the raw gaze estimates for both methods contain residual errors on the order of several degrees, due for example to the offset between the optical and visual axes, which neither method takes into account explicitly.

These errors may be corrected by asking the subject to fixate a number of calibration targets; the difference between the known gaze direction and the gaze direction estimated by the system is stored and is used to correct subsequent measurements.

We refer to this type of procedure as a *recalibration procedure* because it takes an existing gaze estimate and applies a correction to it. In this paper, we will compare three different types of recalibration procedure:

Weighted offsets The first recalibration procedure is the one used by Hennessey et al.; for clarity, we will refer to it as the "weighted offsets" procedure because we will also use other recalibration procedures with Hennessey's eye tracking method.

The weighted offsets procedure is based on the actual and estimated gaze positions in screen coordinates. In the original paper, the corners of the screen are used as calibration targets, but the technique can be used for calibration points at arbitrary positions.

We will refer to the calibration targets (or actual gaze positions) as $\mathbf{g}_i, i = 1, \ldots, n$, and the corresponding estimated gaze positions as $\hat{\mathbf{g}}_i$. We denote the error at each calibration point by $\mathbf{e}_i = \mathbf{g}_i - \hat{\mathbf{g}}_i$.

In the subsequent measurement phase, we correct a raw gaze estimate $\mathbf{g}_{\mathrm{est}}$ as follows: We first compute the distance d_i of $\mathbf{g}_{\mathrm{est}}$ from the estimated gaze position for each calibration point:

$$d_i = \|\mathbf{g}_{\mathrm{est}} - \hat{\mathbf{g}}_i\|_2, \quad i = 1, \ldots, n.$$

We then obtain the corrected gaze estimate $\mathbf{g}_{\mathrm{corr}}$ by adding a weighted combination of the errors $\mathbf{e}_i$ to $\mathbf{g}_{\mathrm{est}}$:

$$\mathbf{g}_{\mathrm{corr}} = \mathbf{g}_{\mathrm{est}} + \sum_{i=1}^{n} w_i \mathbf{e}_i,$$

where the weights w_i are inversely proportional to the distances d_i:

$$w_i = \frac{1}{d_i \sum_{j=1}^{n} \frac{1}{d_j}}.$$

If any d_i should be zero, we set the corresponding w_i to one and the remaining w_j to zero.

Screen coordinate interpolation This method constructs a biquadratic interpolation function that maps $\mathbf{g}_{\mathrm{est}} = (x_{\mathrm{est}}, y_{\mathrm{est}})$ to $\mathbf{g}_{\mathrm{corr}} = (x_{\mathrm{corr}}, y_{\mathrm{corr}})$ as follows:

$$\begin{pmatrix} x_{\mathrm{corr}} \\ y_{\mathrm{corr}} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \end{pmatrix} \begin{pmatrix} 1 \\ x_{\mathrm{est}} \\ y_{\mathrm{est}} \\ x_{\mathrm{est}}\, y_{\mathrm{est}} \\ x_{\mathrm{est}}^2 \\ y_{\mathrm{est}}^2 \end{pmatrix}$$

where the parameters $a_{11}, \ldots, a_{26}$ are chosen using least squares estimation on the calibration data.

Eye angle interpolation This method works in a similar way to screen coordinate interpolation but acts on eye rotation angles instead of screen coordinates. Theoretically, this type of recalibration should compensate better for the angular offset between the visual and optical axes than a recalibration in screen coordinates.

5 Experiments

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 6, which visualizes the visibility of the CRs in the cameras of the Shih setup for two different head positions. 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;

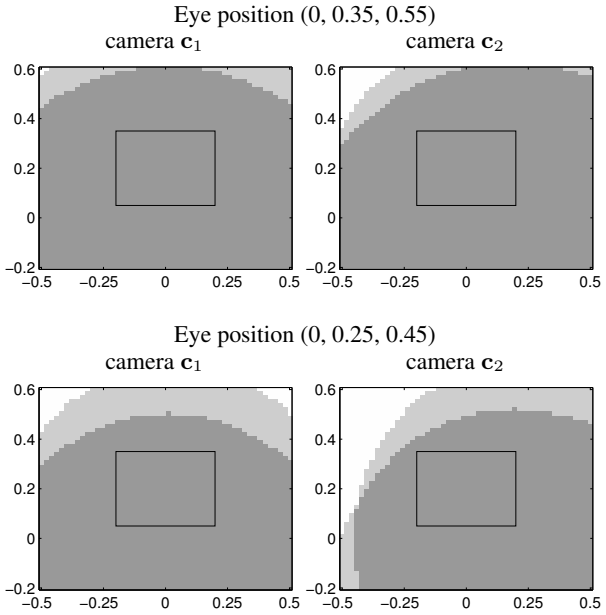


Figure 6: *Visibility of CRs as a function of gaze position in the Shih setup (white: no CR visible; light grey: one CR visible; dark grey: two CRs visible). The rectangular inset marks the boundaries of the screen. (Note: Camera $\mathbf{c}_2$ is identical to the single camera of the Hennessey setup.)*

to ensure that gaze tracking is possible across the whole screen, two glints must be visible for all gaze positions within the screen.

Several factors affect CR visibility. First, compare the top row (where the eye is at the centre of the eye position volume) to the bottom row (where the eye is lower and closer to the screen). As the eye moves closer to the screen, the region where two CRs are visible shrinks. This is because the CRs move further apart on the cornea, and hence the eye can rotate less before one of the CRs moves off the cornea. As the eye moves down, the region where two CRs are visible likewise moves down. This is because, as the eye moves, the same gaze target requires different eye rotation angles, thus moving the CRs closer to or further away from the border of the cornea.

Now compare the two columns of Figure 6, corresponding to the two cameras: The CR visibility region for camera $\mathbf{c}_2$ is shifted to the right compared to camera $\mathbf{c}_1$. This illustrates the reason for the asymmetric placement of the cameras: For a camera positioned a little further right than $\mathbf{c}_2$, one of the CRs would no longer be visible when the user gazes at the top left corner of the screen.

Though we have only shown the results for two eye positions, we have validated that both CRs remain visible for a finely spaced grid of positions spanning the eye position volume. Because camera $\mathbf{c}_2$ of the Shih setup is identical to the single camera of the Hennessey setup, this result holds for both setups.

Comparison of recalibration procedures We will now compare the different recalibration procedures from Section 4.4 in terms of accuracy and head movement tolerance. Figure 7 shows the gaze estimation error of Hennessey's method with the different recalibration procedures. To test the head movement tolerance, we varied the distance of the observer from the screen; at each position, we swept the gaze across a grid of screen positions and recorded the mean gaze estimation error. The eye tracker was calibrated using a

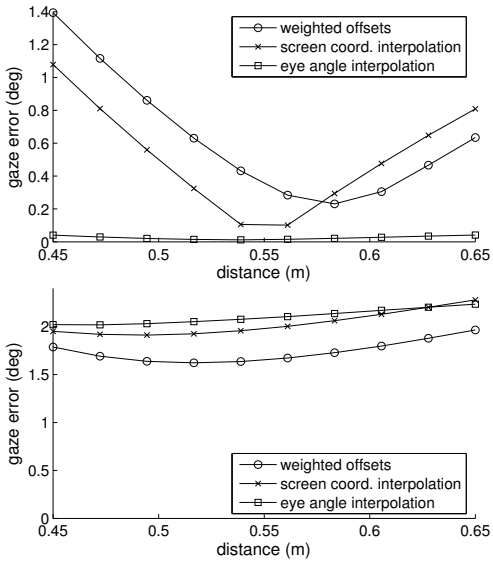


Figure 7: *Comparison of recalibration procedures for Hennessey's method. Top: without feature position error. Bottom: With feature position error (noise level 0.5 pixels).*

nine-point calibration pattern with the eye at the centre of the eye position volume (distance $z = 0.55$ m).

We tested two different levels of feature position error: No error (upper plot) and uniformly distributed error with a maximum magnitude of 0.5 pixels on both coordinate axes (lower plot).

In the zero-error case, it is evident that all recalibration procedures produce the smallest error at or near the calibration position. Moving away from this calibration position, the error for "weighted offsets" and "screen coordinate interpolation" increases quickly to reach more than one degree at $z = 0.45$ m. In contrast, the "eye angle interpolation" procedure is affected much less by head movement; this is to be expected, because most of the residual error before recalibration is due to the offset between the visual axis and the optical axis, which is captured well by this type of recalibration.

Compare these results to the case with feature position error in the lower plot. Not surprisingly, the gaze estimation error for all methods has increased substantially; however, the relative order of the recalibration procedures has also changed. In the presence of feature position error, weighted offsets recalibration performs better than the two interpolation techniques. It appears that weighted offsets is more robust towards inaccuracies in the calibration data, which are now also affected by noise.

Note that there are two effects at work that influence how the error changes with distance. The first is the same that we observed in the zero-error case: The recalibration error tends to be smallest at the calibration position. The second effect is that as the distance of the eye from the camera increases, the image of the eye becomes smaller, so that a given feature position error in pixels leads to a higher gaze error in degrees. Depending on the amount of feature position error, one or the other of these effects will dominate.

As these results show, a method that is superior in theory (i.e. under a zero-error assumption) may not always perform as well in practice. When error is present, weighted offsets recalibration seems to be a good choice, at least compared to the alternatives tested here. For this reason, we will use weighted offsets recalibration in all further experiments, both for Hennessey's and for Shih's method.

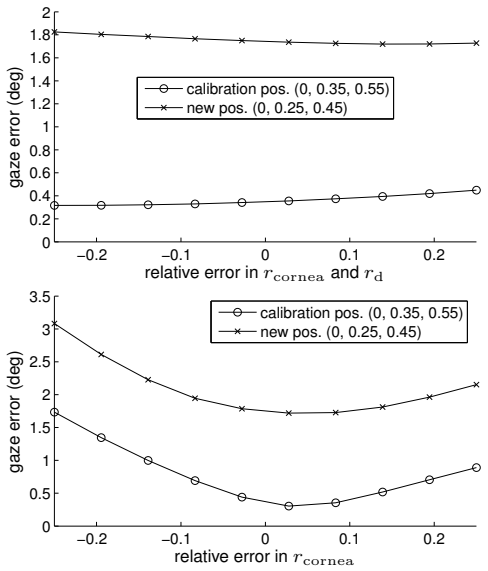


Figure 8: *Effect of inaccuracies in the eye model parameters of Hennessey's method. Top: The parameters r_{cornea} and r_{d} of the simulated eye are varied by the same ratio relative to the values assumed by Hennessey's method. Bottom: Only r_{cornea} is varied while r_{d} is fixed at the assumed value.*

Effect of inaccuracies in eye model parameters Hennessey's method uses population averages for two parameters of the eye: The corneal curvature radius r_{cornea} and the distance r_{d} between corneal centre and pupil centre. An obvious question is how deviations of the actual values for a specific user from the assumed values will affect gaze estimation accuracy, and how well the recalibration procedure can compensate for this.

Eysteinsson et al. [2002] found the standard deviation of r_{cornea} to be about 0.6 mm, which corresponds to around 8%. 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_{d} has a similar distribution, we varied both parameters of the simulated eye from 25% below their assumed values to 25% above. Figure 8 shows the effect of this on the gaze estimation error, both at the calibration position and at the new position $(0, 0.25, 0.45)$. (The feature position error was set to zero.)

The upper 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. While parameter changes do have an effect, this is quite small compared to the effect of head movement.

Because the relative deviation of both parameters will not be exactly the same, we have also investigated the effect of varying just one parameter. The results of varying only r_{cornea} are shown in the lower plot of Figure 8. (Varying only r_{d} has a similar effect, so we omit the corresponding plot.) Now, the effect on gaze estimation error is stronger and reaches almost two degrees for $\pm 25\%$ of relative parameter change. However, it is probably not realistic for r_{cornea} to change this much without r_{d} changing at all; for smaller changes of $\pm 10\%$, gaze error increases by only about half a degree.

In summary, the use of population averages for the eye parameters in Hennessey's method seems justified because it is not a major source of error compared to other factors such as head movement or feature position error.

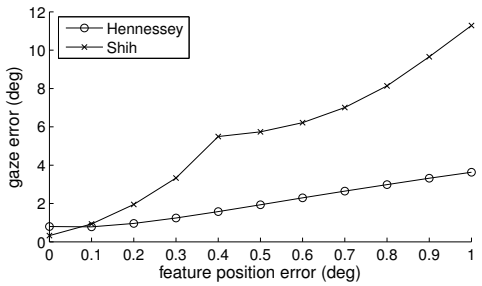


Figure 9: *Gaze estimation error of Hennessey's and Shih's method as a function of feature position error.*

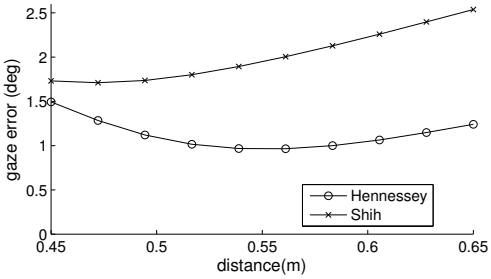


Figure 10: *Gaze estimation error of Hennessey's and Shih's method as a function of eye position (feature position error 0.2 pixels).*

Comparison of Shih's and Hennessey's methods We now compare the gaze estimation accuracy of Shih's and Hennessey's methods. In the tests that follow, we set the parameters r_{cornea} and r_{d} of the simulated eye to 15% and 5% below the values assumed by Hennessey's method. This is intended to simulate the deviations that might be observed in a typical user; setting the parameters to the precise values assumed by Hennessey's method would put it at an unfair advantage compared to Shih's method.

Figure 9 shows the effect of various amounts of feature position error on both methods. (Gaze estimation was performed at the calibration position.) We note that the gaze estimation error for both methods increases in an approximately linear fashion with feature position error. Interestingly, Hennessey's method, which uses only one camera, performs substantially better than Shih's method, which uses two. We speculate that the eye model parameters used by Hennessey's method may be the reason for this; even though the values assumed for these parameters were not identical to the values in the simulated eye, they still constitute additional prior knowledge that provides a strong constraint on the position of the cornea and pupil centre.

What is also apparent is that the high gaze estimation accuracies achieved by current systems (half a degree to one degree) require precise image analysis; according to the plot, to achieve an accuracy of one degree, Hennessey's method requires the position of image features to be determined to within about 0.2 pixels. (Note that these results cannot be compared directly to the accuracies reported by Hennessey et al. because we used a simulated camera with a larger field of view.)

Figure 10 shows the effect of head movement on the two methods at a fixed feature position error of 0.2 pixels. Both methods compensate for head movements fairly well, though slightly stronger deviations are apparent towards the front of the eye position volume for Hennessey's method and towards the back for Shih's method.

6 Discussion and Outlook

We hope this paper has shown how simulation of eye trackers allows us to answer questions that are difficult or impossible to analyze on a physical implementation. Simulation allows us to get at ground truths that we do not know in the real world; to manipulate parameters that are hard to control in the real world; and to make changes to the simulated hardware and algorithms and evaluate the effects of these changes with ease. This ease of experimentation may in turn prompt us to ask questions that we might never consider if we had only the physical implementation to work with.

Another thing that simulation allows us to do is to compare different eye tracking methods under otherwise identical conditions. We have performed such a comparison in this paper and obtained a somewhat surprising result: The single-camera method of Hennessey et al. [2006] is substantially more robust to feature position error than the multi-camera method of Shih et al. [2000].

Of course, mathematical analysis is another important tool for understanding the properties of individual eye tracking methods as well as the eye tracking problem in general; as examples of the kind of insights that can be obtained through rigorous mathematical analysis, we cite the work of Guestrin and Eizenman [2006] and Shih et al. [2000]. However, to remain tractable, a mathematical analysis must make certain simplifications; of course, the same applies to simulation, but simulation can account for a range of factors with relative ease that would be much more difficult to grasp analytically – an obvious example for this is camera noise.

That said, there are quite a number of factors of interest that are not currently simulated in our software, such as the non-spherical shape of the cornea, occlusion of the eye by the eyelids, or the effects of glasses and contact lenses.

So how realistic is the simulation in its present form? We have no definite answer to this, but if anything, the simulated results are probably slightly optimistic, i.e. they underestimate the gaze error that would be made by a real implementation. Given that the simulation already requires fairly low feature position errors to obtain the kind of gaze accuracies achieved by existing eye tracker implementations, we are fairly confident that the simulation models the major factors that affect gaze error in real systems.

An issue that we have sidestepped entirely in this paper is image analysis. We have assumed throughout that features in the image can be localized with a certain degree of accuracy; it would certainly be worthwhile to examine how different factors (camera resolution, signal-to-noise ratio, algorithms employed etc.) influence this accuracy. Such an investigation would use synthetic eye images generated using 3D rendering software and post-processed to introduce noise, model the effect of lens distortion, and so on. The coordinates of the image features extracted by the image analysis routines could then be compared against the known ground truth.

We encourage researchers to download our software, experiment with it, apply it to their own eye tracker systems, make additions and improvements to the underlying simulation framework, and to share those improvements with the community in the spirit of open-source software.

Acknowledgements

Our research has received funding from the European Commission within the project GazeCom (contract no. IST-C-033816) and (for travel support) the Network of Excellence COGAIN (contract no. IST-2003-511598) within the Information Society Technologies (IST) priority of the 6th Framework Programme. This publication reflects the views only of the authors; the the European Commission cannot be held responsible for any use which may be made of the information contained therein.

References

ATCHISON, D. A., AND SMITH, G. 2000. *Optics of the Human Eye*. Butterworth Heinemann, Oxford, UK.

BEYMER, D., AND FLICKNER, M. 2003. Eye gaze tracking using an active stereo head. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 451–458.

BOFF, K. R., AND LINCOLN, J. E. 1988. *Engineering Data Compendium: Human Perception and Performance*. AAMRL, Wright-Patterson AFB, OH.

CHARLIER, J. R., BEHAGUE, M., AND BUQUET, C. 1994. Shift of the pupil center with pupil constriction. *Investigative Ophthalmology and Visual Science 35*, 4, 1278.

DUCHOWSKI, A. T. 2003. *Eye Tracking Methodology: Theory and Practice*. Springer, New York.

EYSTEINSSON, T., JONASSON, F., SASAKI, H., ARNARSSON, A., SVERRISSON, T., SASAKI, K., STEFÁNSSON, E., AND THE REYKJAVIK EYE STUDY GROUP. 2002. 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.

FORSYTH, D. A., AND PONCE, J. 2002. *Computer Vision: A Modern Approach*. Prentice Hall.

GUESTRIN, E. D., AND EIZENMAN, M. 2006. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on Biomedical Engineering 53*, 6, 1124–1133.

HALÍŘ, R., AND FLUSSER, J. 1998. 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)*, vol. 1, 125–132.

HAUSTEIN, W. 1989. 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.

HENNESSEY, C., NOUREDDIN, B., AND LAWRENCE, P. 2006. A single camera eye-gaze tracking system with free head motion. In *Proceedings of Eye Tracking Research & Applications (ETRA)*, 87–94.

MORIMOTO, C. H., AND MIMICA, M. R. M. 2005. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding 98*, 1, 4–24.

SHIH, S.-W., WU, Y.-T., AND LIU, J. 2000. A calibration-free gaze tracking technique. In *Proceedings of the 15th International Conference on Pattern Recognition*, 201–204.

STRAUBE, A., AND BÜTTNER, U. 2007. *Neuro-Ophthalmology: Neuronal Control of Eye Movements*. Karger, Basel, Switzerland.

TRUCCO, E., AND VERRI, A. 1998. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall.

VON HELMHOLTZ, H. 1910. *Handbuch der physiologischen Optik*, 3rd ed. Voss, Hamburg, Leipzig.