A fully-automatic, temporal approach to single camera, glint-free 3D eye model fitting

Lech Świrski and Neil Dodgson
University of Cambridge

We present a 3D eye model fitting algorithm for use in gaze estimation, that operates on pupil ellipse geometry alone. It works with no user-calibration and does not require calibrated lighting features such as glints. Our algorithm is based on fitting a consistent pupil motion model to a set of eye images. We describe a non-iterative method of initialising this model from detected pupil ellipses, and two methods of iteratively optimising the parameters of the model to best fit the original eye images. We also present a novel eye image dataset, based on a rendered simulation, which gives a perfect ground truth for gaze and pupil shape. We evaluate our approach using this dataset, measuring both the angular gaze error (in degrees) and the pupil reprojection error (in pixels), and discuss the limitations of a user-calibration–free approach.

Keywords: gaze estimation, eye model, pupil detection, glint-free

Introduction

Camera-based eye tracking approaches are normally divided into two stages: eye/pupil detection and gaze-estimation based on the eye or pupil information. Eye/pupil detection algorithms normally work in 2D image space; a common approach is to detect the pupil as an ellipse in 2D. Gaze estimation algorithms then attempt to convert this eye/pupil information into a gaze vector or point of regard.

Most gaze estimation algorithms can be classified into two groups: regression-based and model-based. Regression-based algorithms assume that there is some unknown relationship between the detected eye parameters and the gaze. They then approximate this relationship using some form of regression — often this is polynomial regression, although there are also approaches using neural networks. Model-based approaches instead attempt to model the eye and thus the gaze. Given the detected eye/pupil parameters, these adjust the model (e.g. rotate the eye) to fit the data, and output the gaze information. We refer the reader to the in-depth survey by Hansen and Ji (2010).

Both approaches require some form of personal calibration, either to find the parameters of the regression function or to fit the eye model to the current user. Normally this consists of an interactive calibration, where the user is asked to look at several points at known locations; for example, many studies use a 9-point grid calibration. Model-based approaches often use anthropomorphic averages to decrease the number of variable parameters, however they normally still require some amount of calibration.

Additionally, many approaches use glints or Purkinje images as additional data in either the calibration (e.g. to obtain corneal curvature) or inference (e.g. using the glint–pupil vector rather than the pupil position alone). These are reflections of a light source from various parts of the cornea or lens. However, these require one or more calibrated light sources and can be fairly difficult to detect under uncontrolled lighting conditions. Some approaches also use multiple cameras to build a 3D model of the eye from stereo information.

However, there are many use cases where such high-quality, controlled and calibrated lighting is not available. There is an increasing amount of research into "home-made" cheap eye-trackers, where a webcam is mounted on glasses frames, and illuminated either by visible light or IR LEDs (Agustin, Skovsgaard, Hansen, & Hansen, 2009; Chau & Betke, 2005; Tsukada, Shino, Devyver, & Kanade, 2011). Figure 1 is an example of such an eye-tracker, built in our lab by the first author. In such systems, it can be difficult or even impossible to calibrate the positions of the lights, so glint-free approaches must be used.

We present a model-base gaze estimation approach that does not require any interactive calibration from the user, only requires a single camera, and does not require calibrated lights for glint information. Instead, we only require multiple images of the pupil from a head-mounted camera.

We evaluate our approach using a new dataset, created by rendering a highly-realistic 3D model of the eye and surrounding areas. Using a rendering rather than real video allows us to calculate the ground truth with perfect accuracy, rather than relying on another form

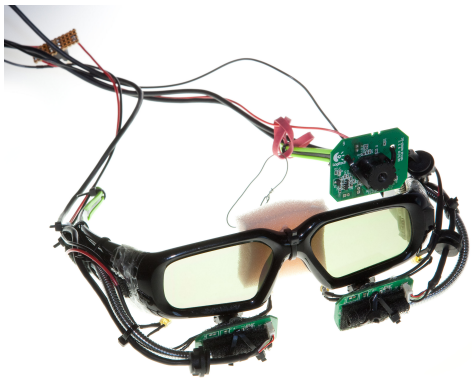

Figure 1. An example of a cheap, head-mounted eye tracker, build in our lab, which uses webcams and roughly positioned IR LED lighting.

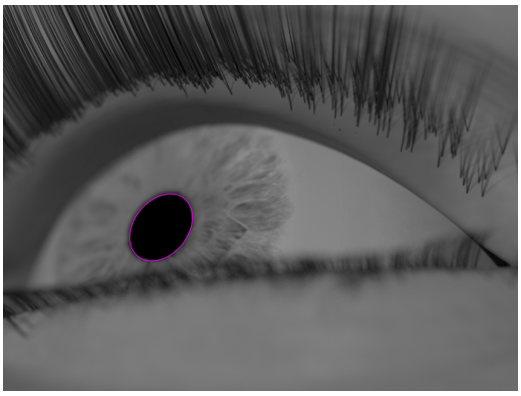

Figure 2. The pupil ellipse, in magenta, detected by our pupil tracking algorithm (Świrski et al., 2012)

of measurement. We also discuss the limitations of a fully-automatic system.

Our approach

Our approach is based on the projective geometry of the pupil. After detecting the pupil as an ellipse in the image, we approximate the orientation and position of the original circular pupil contour in 3D. By combining information from multiple frames, we build an eye model based on assumptions on how the motion of the pupil is constrained.

In our model, we do not consider the offset between the optical axis and the visual axis — that is, we calculate the gaze vector rather than the sight vector. While this means that the model cannot be directly used to calculate the point of regard, the offset between the gaze and sight vectors is constant per individual, and can either be approximated by anthropomorphic averages or trivially be found using a single point calibration.

We also do not consider head motion. Since we assume that the camera is head-mounted, we can operate entirely in camera space — this means that the gaze vector returned is relative to the camera. For a gaze-tracking system that requires gaze in world space, we expect the camera's position and orientation to be externally tracked. The transformation from camera space would simply amount to a single rotation and translation.

Our approach proceeds as follows. We first detect the pupil ellipse in each image independently. We use our pupil detection algorithm (Świrski, Bulling, & Dodgson, 2012), which gives us a pupil ellipse (fig. 2) and a set of edge points from which the pupil was calculated.

Once we have the pupil ellipses in all the images, we independently unproject each one as a circle in 3D. We then combine the information from these 3D circles to create a rough model of the pupil motion. Finally, we optimise the model parameters to best fit the original image data.

Our approach of using a 3D eye model combined with an optimisation step is similar to Tsukada et al. (2011). However, this work manually sets the 3D eye model parameters and only refines the per-frame pupil parameters, whereas we automatically estimate and refine all of the parameters.

We describe the stages of our approach in detail in the following sections.

Two-circle unprojection

The first stage of our algorithm 'unprojects' each pupil ellipse into a 3D pupil circle — that is, we find a circle whose projection is the given ellipse. Many approaches simplify this unprojection by assuming a scaled orthogonal or weak perspective projection model, where the unprojection can be calculated using simple trigonometry (Schnieders, Fu, & Wong, 2010; Tsukada et al., 2011). However, weak perspective is only an approximation to full perspective, and it is valid only for distant objects that lie close to the optical axis of the camera. When the pupil is close to the camera, or far away from the optical axis of the camera, the weak perspective approximation begins to fail. Instead, we assume a full perspective projection with a pinhole camera model.

Under a a full perspective projection model, the space of possible projections of the pupil circle can be seen as a cone with the pupil circle as the base and camera focal point as the vertex. The pupil ellipse is then the intersection of this cone with the image plane.

This means that the circular unprojection of the ellipse can be found by reconstructing this cone, using the ellipse as the base. The circular intersection of this cone will then be the pupil circle (fig. 3). We find the circular intersection using the method of Safaee-Rad,

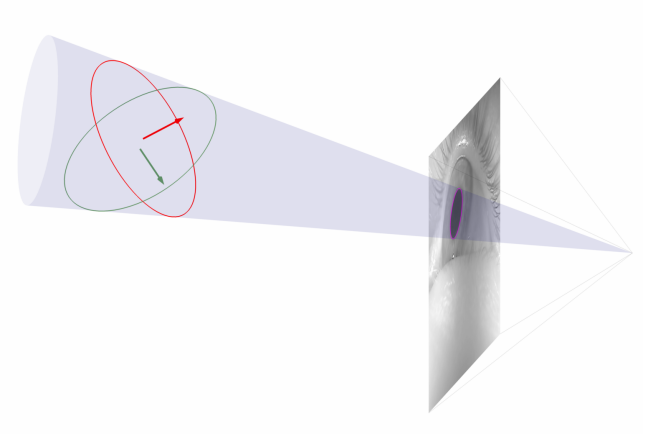

Figure 3. We unproject the pupil ellipse by constructing a cone through the camera focal point and pupil ellipse on the image plane. We then find circular intersections of this cone—at any given distance, there are two solutions for the intersection (green and red).

Tchoukanov, Smith, and Benhabib (1992). This unprojection operation gives us a pupil position, a gaze vector, and a pupil radius

$$\textit{pupil circle} = (\mathbf{p}, \mathbf{n}, r) \tag{1}$$

If necessary, the gaze is flipped so that it points 'towards' the camera.

There are two ambiguities when unprojecting an ellipse. The first is a distance–size ambiguity; any perspective unprojection from 2D into 3D is ill-posed, and there is no way to tell if the pupil is small and close, or large and far away. We resolve this by setting r to an arbitrary value in this stage, and finding the true radius in later stages of the algorithm.

The second is that there are two solutions when finding the fixed-size circular intersection of the constructed cone, symmetric about the major axis of the ellipse, arising from solving a quadratic (shown in green and red in figure 3). At this stage of the algorithm, we do not disambiguate between these two cases, and return both solutions, denoted as

$$(\mathbf{p}^+, \mathbf{n}^+, r), (\mathbf{p}^-, \mathbf{n}^-, r) \tag{2}$$

Model initialisation

To estimate the gaze, we consider only the pupil orientation and location, and so we do not require a full model of the eye. In fact, we wish to model only the pupil and its range of motion, and not the iris or the eyeball itself.

Therefore, instead of modelling the pupil as a hole in the iris of a spherical eyeball, we model it as a disk laying tangent to a rotating sphere. This sphere has the same centre of rotation as the eyeball. The gaze is then the normal of the disk, or equivalently a radial vector from the sphere centre to the pupil centre.

Sphere centre estimate

For each eye image, we consider the pupil circle $(\mathbf{p}_i, \mathbf{n}_i, r_i)$, where i is the index of the image. Given our pupil model, we wish to find a sphere which is tangent to every pupil circle. Since each pupil circle is tangent to the sphere, the normals of the circles, $\mathbf{n}_i$, will be radial vectors of the sphere, and thus their intersection will be the sphere's centre.

However, there are two problems with this approach, corresponding to the two ambiguities in the ellipse unprojection. Firstly, we do not know the true 3D position of each pupil circle, only the position under the assumption that the pupil is of a certain size. If the pupil radius r_i did not change between frames, as the relative unprojected positions would be correct up to scale. This is the case for approaches which use the iris contour rather than the pupil contour. However, due to pupil dilation, we cannot make this assumption. Secondly, we have two circles for each ellipse, rather than one, and we do not know which one of the two circles—$(\mathbf{p}_i^+, \mathbf{n}_i^+, r)$ or $(\mathbf{p}_i^-, \mathbf{n}_i^-, r_i)$—is correct.

We resolve both of these problems by considering the intersection of projected normal vectors $\tilde{\mathbf{n}}_i^{\pm}$ in 2D image-space rather than 3D world-space. In this case, the distance–size ambiguity disappears by construction, as we are using the same projection which originally introduced it. The two-circle ambiguity also disappears: both projected normal vectors are parallel:

$$\tilde{\mathbf{n}}_i^+ \propto \tilde{\mathbf{n}}_i^- . \tag{3}$$

Similarly, the line between the two projected circle centres, $\tilde{\mathbf{p}}_i^+$ and $\tilde{\mathbf{p}}_i^-$, is parallel to $\tilde{\mathbf{n}}_i^{\pm}$. This means that:

$$\exists s,t \in \mathbb{R}.\ \tilde{\mathbf{p}}_i^+ = \tilde{\mathbf{p}}_i^- + s\tilde{\mathbf{n}}_i^+ = \tilde{\mathbf{p}}_i^- + t\tilde{\mathbf{n}}_i^+ . \tag{4}$$

which means that we can arbitrarily choose either one of the two solutions for this stage.

We thus find the projected sphere centre $\tilde{\mathbf{c}}$ by calculating an intersection of lines. These lines correspond to the projected gaze: each line passes through the projected pupil centre and lies parallel to the projected pupil normal (figure 4). Formally, we find the intersection of the set of lines $\mathbf{L}_i$, where

$$\mathbf{L}_i = \{\ (x,y) = \tilde{\mathbf{p}}_i + s\tilde{\mathbf{n}}_i \mid s \in \mathbb{R}\ \} \tag{5}$$

Since there may be numerical, discretisation or measurement error in these vectors, the lines will almost certainly not intersect at a single point. We instead find the point closest to each line in a least-squares sense, by calculating

$$\tilde{\mathbf{c}} = \left(\sum_i I - \tilde{\mathbf{n}}_i \tilde{\mathbf{n}}_i^T \right)^{-1} \left(\sum_i (I - \tilde{\mathbf{n}}_i \tilde{\mathbf{n}}_i^T) \tilde{\mathbf{p}}_i \right) \tag{6}$$

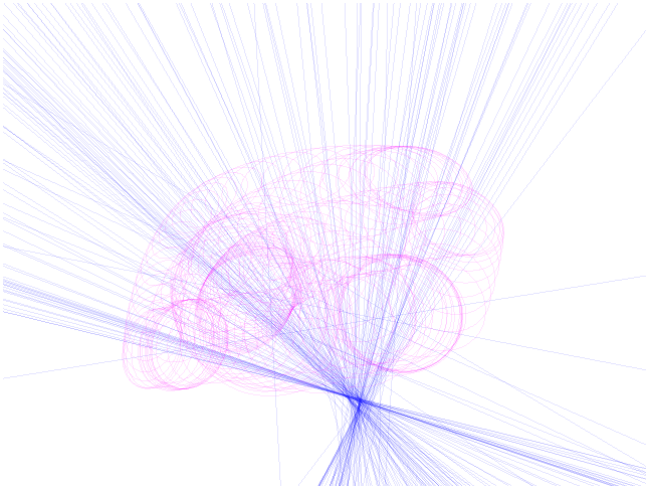

Figure 4. To find the centre of the sphere, we intersect the projected gaze vectors of each pupil. The projected sphere centre is estimated to be near the intersection of these lines.

We then unproject the projected sphere centre $\tilde{\mathbf{c}}$ to find the 3D sphere centre $\mathbf{c}$. Again, there is a size–distance ambiguity in the unprojection, which we resolve by fixing the z coordinate of $\mathbf{c}$.

Sphere radius estimate

Once we have the projected sphere centre $\tilde{\mathbf{c}}$, we note that each pupil's normal $\mathbf{n}_i$ has to point away from the sphere centre $\mathbf{c}$:

$$\mathbf{n}_i \cdot (\mathbf{c} - \mathbf{p}_i) > 0 \tag{7}$$

and therefore the projected normal $\tilde{\mathbf{n}}_i$ has to point away from the projected centre $\tilde{\mathbf{c}}$:

$$\tilde{\mathbf{n}}_i \cdot (\tilde{\mathbf{c}} - \tilde{\mathbf{p}}_i) > 0 \tag{8}$$

Furthermore, since they are symmetric about the major axis of the ellipse $\tilde{\mathbf{n}}_i^+$ and $\tilde{\mathbf{n}}_i^-$ point in opposite directions, which means that one will point towards $\tilde{\mathbf{c}}$, and one will point away from it. This allows us to disambiguate between the two-circle problem, by choosing the circle $(\mathbf{p}_i, \mathbf{n}_i, r_i)$ whose projected normal $\tilde{\mathbf{n}}_i$ points away from $\tilde{\mathbf{c}}$.

We can now use the unprojected pupils to estimate the sphere radius R. Since each pupil lies on the sphere, this should be the distance between $\mathbf{p}_i$ and the $\mathbf{c}$—however once again, due to the distance–size ambiguity in the pupil unprojection, and the potentially changing actual pupil size, we cannot use $\mathbf{p}_i$ directly.

Instead, we consider a different candidate pupil centre $\hat{\mathbf{p}}_i$, which is another possible unprojection of $\tilde{\mathbf{p}}_i$, but potentially at a different distance. This means that this point has to lie somewhere along the line of possible unprojections of $\tilde{\mathbf{p}}_i$, which is the line passing through $\mathbf{p}_i$ and the camera centre.

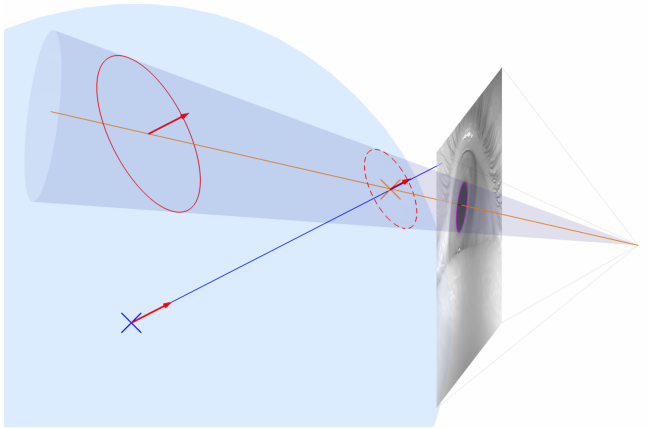

Figure 5. For each candidate pupil circle (red), we consider the possible unprojections of its centre $\mathbf{p}_i$ (orange line). We intersect this line with the gaze line from the sphere centre (blue line) to find $\hat{\mathbf{p}}_\mathbf{i}$ (orange cross), which is the centre of a circle that lies tangent to the sphere (dashed red). The distance from the sphere centre (blue cross) to $\hat{\mathbf{p}}_\mathbf{i}$ is the sphere radius.

We want the circle $(\hat{\mathbf{p}}_i, \mathbf{n}_i, \hat{r}_i)$ to be consistent with our assumptions: that the circle lies tangent to the sphere whose centre is $\mathbf{c}$. This means that we want $\mathbf{n}_i$ to be parallel to the line from $\mathbf{c}$ to $\hat{\mathbf{p}}_i$.

Given these two constraints, $\hat{\mathbf{p}}_i$ can be found by intersecting the line from $\mathbf{c}$ to $\hat{\mathbf{p}}_i$—the gaze line from the centre of the sphere—with the line passing through $\mathbf{p}_i$ and the camera centre—the projection line of $\mathbf{p}_i$ (figure 5). Since lines in 3D almost certainly do not cross, we once again find the least-squares intersection point.

The sphere radius R is then calculated by finding the mean distance from the sphere centre to each pupil centre.

$$R = \text{mean}\left(\{\, R_i = \|\hat{\mathbf{p}}_i - \mathbf{c}\| \mid \forall i \,\}\right) \tag{9}$$

Consistent pupil estimate

Given the sphere, calculated above, we want the pupil circles to lay tangent to the surface. For each pupil, we wish for its centre to lie on the sphere, and for its projection to be $\tilde{\mathbf{p}}_i$. Due to the distance–size ambiguity, it is almost certain that the circle $(\mathbf{p}_i, \mathbf{n}_i, r_i)$ will not lie on the sphere. We therefore want to calculate a new circle, $(\mathbf{p}'_i, \mathbf{n}'_i, r'_i)$, where

$$\mathbf{p}'_i = s\mathbf{p}_i \tag{10}$$

$$\mathbf{p}'_i = \mathbf{c} + R\mathbf{n}'_i \tag{11}$$

$$\frac{r'_i}{z'_i} = \frac{r_i}{z_i} \tag{12}$$

The last of these defines r'_i as r_i scaled by perspective.

To find $\mathbf{p}'_i$, we wish to find a value of s such that $s\mathbf{p}_i$ lies on the surface of the sphere $(\mathbf{c}, R)$, which can be

calculated as a line–sphere intersection. This gives two solutions, of which we take the nearest. $\mathbf{n}_i'$ and r_i' are then trivially calculated. Note that as part of this process, we discard the original gaze vector $\mathbf{n}_i$.

Once these steps are taken, we have a rough model of the pupil motion, where every pupil circle lies tangent to the surface of a certain sphere.

Model optimisation

We parametrise our model using $3+3N$ parameters, where N is the number of pupil images. These parameters represent the 3D position of the sphere, $\mathbf{c}$, and for each pupil, its position on the sphere (as two angles, θ and ψ) and its radius r.

From the previous initialisation steps, we obtain a set of parameters approximating the original detected pupil ellipses. We then wish to optimise these parameters so that the projected pupil ellipses best fit the original eye image data. The result of the optimisation is a set of circles in 3D, all of which are constrained to lie tangent to a sphere. The normals of these circles correspond to the gaze vector of the user, and the projected ellipses are a good fit for the pupils in the images.

We investigate two metrics for defining the "best fit". The first is an region comparison, where we attempt to maximise the contrast between the inside and outside of each pupil. The second is a point distance, where we attempt to minimise the distance of each pupil ellipse from the pupil edge pixels found by the pupil tracker.

Region contrast maximisation

The first metric we use is a region contrast metric. Simply put, it requires that the pupil ellipse be dark on the inside and light on the outside, and for the difference in brightness of the inside and outside to be maximal.

For each pupil image, we consider a thin band around the interior and exterior of the pupil ellipse, R^+ and R^- (figure 6). These are defined as

$$R^+ = \left\{ \mathbf{x} \;\middle|\; \mathbf{x} \in \mathbb{R}^2, 0 < d(\mathbf{x}) \leq w \right\} \tag{13}$$

$$R^- = \left\{ \mathbf{x} \;\middle|\; \mathbf{x} \in \mathbb{R}^2, -w < d(\mathbf{x}) \leq 0 \right\} \tag{14}$$

where w is the width of the band (we use $w = 5\,\text{px}$), and $d(\mathbf{x})$ is a signed distance function of a point to the ellipse edge, positive inside the ellipse and negative outside. As calculating the distance of a point to the ellipse edge is non-trivial, we use an approximate distance function, described in the later "Ellipse distance" section.

We then find the mean pixel value of each region – that is, we find

$$\mu^\pm = \text{mean}\left(\left\{ I(\mathbf{x}) \;\middle|\; \mathbf{x} \in R^\pm \right\} \right) \tag{15}$$

where $I(\mathbf{x})$ is the image value at $\mathbf{x}$.

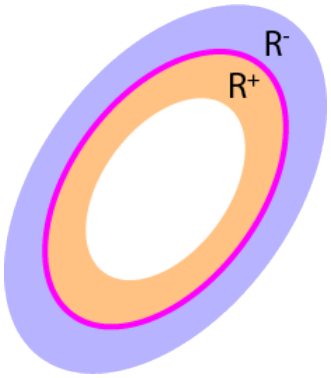

Figure 6. To optimise the model, we consider a thin band around the interior (R^+) and exterior (R^-) of each pupil ellipse. We maximise contrast by maximising the difference in the average pixel value inside these two regions.

This can be rewritten as a weighted average

$$\mu^\pm = \frac{\int B^\pm(d(\mathbf{x})) \cdot I(\mathbf{x})\, d\mathbf{x}}{\int B^\pm(d(\mathbf{x}))\, d\mathbf{x}} \tag{16}$$

(17)

where

$$B^+(t) = \begin{cases} 1 & \text{if } 0 < t \leq w \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$B^-(t) = \begin{cases} 1 & \text{if } -w < t \leq 0 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

so that $B^\pm(d(\mathbf{x}))$ is 1 when $\mathbf{x} \in R^\pm$, and 0 otherwise.

$B^\pm(t)$ can then be defined using the Heaviside function

$$B^+(t) = H(t) - H(t-w) \tag{20}$$

$$B^-(t) = H(t+w) - H(t) \tag{21}$$

This gives a closed definition of "mean pixel value in the region". Our contrast metric per ellipse is then simply

$$E = \mu^- - \mu^+ \tag{22}$$

which increases when the interior region becomes darker, or the exterior region becomes lighter. We then maximise the contrast over all ellipses, that is, for the parameter vector $\mathbf{p}$ we find

$$\underset{\mathbf{p}}{\text{argmax}} \sum_i E_i \tag{23}$$

We maximise this using gradient ascent — specifically, using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Nocedal & Wright, 1999).

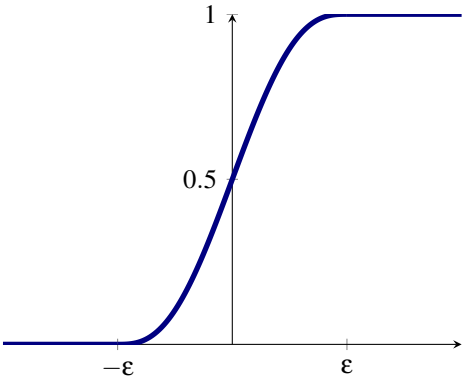

Figure 7. The soft step function, H_ε.

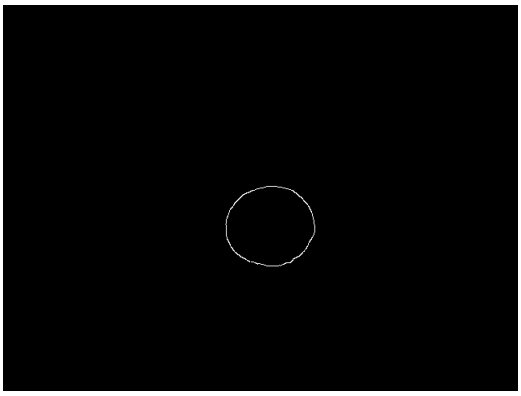

Figure 8. The edge pixels for a detected pupil ellipse. We wish to minimise the distance of all of these edge points to their respective ellipses.

Since we are dealing with discrete pixels rather than a continuous image space, the above weighted integral becomes a weighted sum over pixels. This introduces discretisation issues in the differentiation, as the differential of $H(t)$ is 0 everywhere except at $t = 0$.

To ameliorate these issues, we take the approach of Zhao, Chan, Merriman, and Osher (1996) of using a regularised version of the Heaviside function, $H_\varepsilon(t)$, which provides a smooth, continuous unit step rather than an immediate step. We use `smootherstep` (Ebert, Musgrave, Peachey, Perlin, & Worley, 2002):

$$H_\varepsilon(t) = \begin{cases} 1 & \text{if } t \geq \varepsilon \\ 0 & \text{if } t \leq -\varepsilon \\ 6\left(\frac{t+\varepsilon}{2\varepsilon}\right)^5 - 15\left(\frac{t+\varepsilon}{2\varepsilon}\right)^4 + 10\left(\frac{t+\varepsilon}{2\varepsilon}\right)^3 & \text{otherwise} \end{cases} \tag{24}$$

which defines a sigmoid for t between $-\varepsilon$ and ε, and behaves like the Heaviside function otherwise (figure 7). We use $\varepsilon = 0.5\,\text{px}$. Note that $\lim_{\varepsilon\to 0} = H_\varepsilon(t)$.

It is interesting to note what happens as w tends to 0. The integrals over $B^+(t)$ and $B^-(t)$ tend towards path integrals over the ellipse contour, and so:

$$\lim_{w\to 0} \int B^{\pm}(d(\mathbf{x}))\, d\mathbf{x} = \oint 1\, dt \tag{25}$$

$$\lim_{w\to 0} \int B^{\pm}(d(\mathbf{x})) \cdot I(\mathbf{x})\, d\mathbf{x} = \oint I(\mathbf{x}(t))\, dt \tag{26}$$

This means that μ^+ and μ^- tend towards being the average value of the pixels along the circumference of the ellipse C:

$$\lim_{w\to 0} \mu^{\pm} = \frac{1}{C} \oint I(\mathbf{x}(t))\, dt \tag{27}$$

As μ^+ and μ^- approach the ellipse boundary from opposite "sides", their difference (scaled by w) can be interpreted as a differential of this average value across the ellipse boundary:

$$\begin{aligned} \lim_{w\to 0} \frac{E}{w} &= \lim_{w\to 0} \frac{\mu^- - \mu^+}{w} \\ &= \frac{\partial}{\partial r} \frac{1}{C} \oint I(\mathbf{x}(t))\, dt \end{aligned} \tag{28}$$

This bears a strong resemblance to the integrodifferential operator used by Daugman for pupil and iris detection (Daugman, 2004).

Edge distance minimisation

The second metric we use is an edge pixel distance minimisation. From the pupil detection, we obtain a list of edge pixel locations which were used to define the given pupil ellipse (fig. 8). Then, for each reprojected pupil ellipse, we wish to minimise the distances to the corresponding edge pixels.

We minimise these distances using a least-squares approach. For each ellipse, we consider the set of edge pixels $E = \{\mathbf{e}\}$. We wish to minimise the squared distance of each edge pixel to the ellipse edge; that is, we wish to minimise

$$\sum_{\mathbf{e}\in E} d(\mathbf{e})^2 \tag{29}$$

where $d(\mathbf{x})$ is a signed distance function of a point $\mathbf{x}$ to the ellipse edge, defined in the next section.

We then wish to do optimise the parameters to minimise this distance over all ellipses

$$\underset{\mathbf{p}}{\text{argmax}} \sum_i \sum_{\mathbf{e}\in E_i} d(\mathbf{e})^2 \tag{30}$$

Note that this is a minimisation of a sum of squares. Thus, we can use a least squares minimisation algorithm — we use the Levenberg-Marquadt implementation in Ceres Solver, a C++ least squares minimisation library (Agarwal & Mierle, n.d.).

Ellipse distance

Both metrics above require a function $d(\mathbf{x})$ which is a signed distance of the point $\mathbf{x}$ to the the ellipse edge, positive inside the ellipse an negative outside.

Calculating the true Euclidean distance of a point to an ellipse is computationally expensive, requiring the solution of a quartic. Instead, we transform the image space so that the ellipse becomes the unit circle, find the signed distance to this circle, and scale by the major radius of the ellipse to get an approximate pixel distance.

While this does not give a true Euclidean distance, it is a good approximation for ellipses with low eccentricity, and we have found it to be sufficient for our needs. It is also very efficient to calculate this distance over a grid of pixels, as the transformation can be represented by a matrix multiplication, and this matrix can be precomputed for each ellipse rather than being computed for every pixel.

Automatic differentiation

For both of the above optimisations, we need to calculate the gradient of the metric with respect to the original parameters. To avoid both having to calculate the gradient function by hand, and the inaccuracy of numeric differentiation, we employ automatic differentiation (Rall, 1981).

Notice that any implementation of a function will, ultimately, be some composition of basic operations, such as addition, multiplication or exponentiation, or elementary functions such as *sin*, *cos* or *log*. For any of these, we can already calculate the derivative of any value trivially; by repeatedly applying the chain rule, we can then calculate the value of the derivative of any arbitrary composition of these functions.

Automatic differentiation works based on this observation. Every basic operation or elementary function is redefined to take a pair of inputs: a value and its differentials. These functions are then rewritten to simultaneously compute both the output value and its differentials, applying the chain rule to the input differentials where appropriate. For example, the sin function

$$f(x) = \sin(x) \tag{31}$$

is rewritten as the simultaneous calculation of value and gradient:

$$f\left(\left\langle x, \frac{dx}{dp}\right\rangle\right) = \left\langle \sin(x), \cos(x)\frac{dx}{dp}\right\rangle \tag{32}$$

Thus, for every function, the values of the differentials of that function can be calculated at the same time as the value of the function itself, by repeating the above reasoning. In particular, this can be done for our metric functions. Each parameter is initialised with a differential of 1 with respect to itself, and 0 with respect to all other differentials; passing these annotated parameters through the metric function gives a vector of differentials of the function with respect to each parameter. This can be implemented simply in C++ using operator overloading and function overloading.

Note that, while the individual basic function differentiations are performed manually (this is, it is the programmer who specifies that $\frac{d}{dp}\sin(x) = \cos(x)\frac{dx}{dp}$), this is not symbolic differentiation. This is because symbolic differentiation operates on a symbolic expression of the entire function, and returns a symbolic expression describing the differential, whereas automatic differentiation only operate on the values of the differential at any given point.

Performing the differentiation as described above would increase the runtime by $O(3+3N)$, as we would have to calculate the differential with respect to all $O(3+3N)$ parameters. However, the bulk of each metric calculation is performed on each pupil independently. Since pupils are independent of each other — that is, each pupil only depends on the 3 sphere parameters and its own 3 parameters — we can pass only the 6 relevant parameters to the part of the calculation dealing with only that pupil, hence only imparting a constant runtime increase for the bulk of the calculation.

Evaluation

We evaluate our algorithm by calculating the gaze error and pupil reprojection error on a ground truth dataset. We evaluate both optimisation metrics, and compare them to naïve ellipse unprojection.

Ground truth dataset

To evaluate our algorithm, we chose to work entirely under simulation, so that we would have perfect ground truth data. We took a model of a head from the internet (Holmberg, 2012), and modified it slightly in Blender to add eyelashes, control of pupil dilation, and changed the texture to be more consistent with infrared eye images. We then rendered an animation of the eye looking in various directions, with the pupil dilating and constricting several times (fig. 9). This dataset, as well as the 3D model used to generate it, are publically available[1].

It is often argued that simulation can never accurately emulate all the factors of a system working in real life, however we argue that simulated data is perfectly appropriate for evaluating this sort of system.

Firstly, using simulated images gives a fully controlled environment; in particular measurements of gaze direction, pupil size or camera position can be specified precisely. In real images, on the other hand, this information is obtained by additional measurement, by using another system, assumptions on where

[1] http://www.cl.cam.ac.uk/research/rainbow/projects/eyemodelfit/

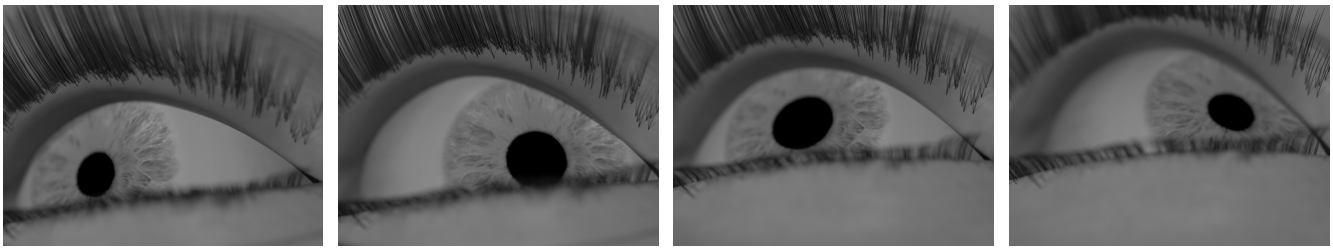

Figure 9. Examples of rendered images from our ground truth dataset.

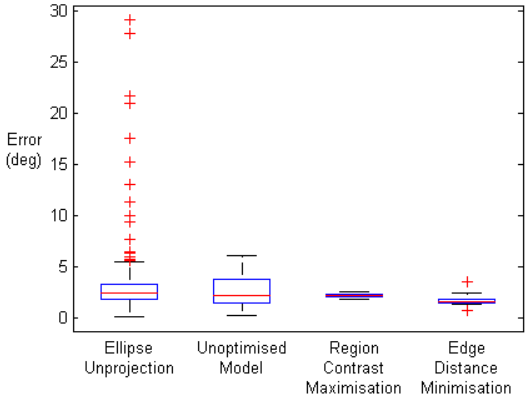

Figure 10. A comparison of the angular gaze error, in degrees, of naïve unprojection and our approach.

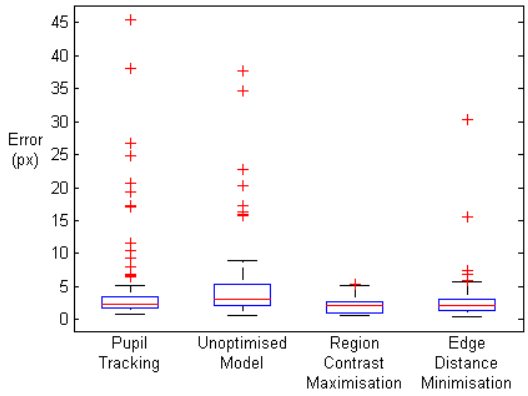

Figure 11. A comparison of the pupil ellipse reprojection error, in pixels, of the pupil tracker input and our approach.

Table 1
Mean and standard deviation of the angular gaze error.

	Mean	Std. dev.
Ellipse Unprojection	3.5558	4.2385
Unoptimised Model	2.6890	1.5338
Region Contrast Maximisation	2.2056	0.1629
Edge Distance Minimisation	1.6831	0.3372

Table 2
Mean and standard deviation of the ellipse error.

	Mean	Std. dev.
Pupil Tracker	3.9301	5.8056
Unoptimised Model	4.4420	4.9157
Region Contrast Maximisation	2.1194	1.1354
Edge Distance Minimisation	2.7474	2.7615

the user is looking or manual labelling. All of these are prone to error, while the ground truth from simulation data is perfect by definition.

Secondly, we argue that modern image rendering techniques are closing the gap between simulation and real life. Our simulation includes eyelashes, skin deformation, iris deformation, reflections, shadows and depth-of-field blur. Furthermore, future work on rendered datasets could include further simulation of real-world variables, such as varying eye shapes, eye colours, different lighting conditions, environmental reflections, or motion blur. We believe that this is an area with vast potential for future work.

In short, we argue that, while simulation is not perfect, the quality of simulated images is approaching that of real images, and the benefits of perfect ground truth data far outweigh the issues of simulation.

Results

We measured the angular gaze error of our approach, using both optimisation metrics, and compared it against the error from simple ellipse unprojection, and the error of the model before optimisation. Figure 10 shows a graph of these results, and Table 1 shows a summary.

The simple ellipse unprojection had, as expected, the highest gaze error. We found that this approach particularly suffers when the pupil fit is poor, as there is no regularisation of the resulting gaze vector (fig. 12). This results in significant gaze errors, as high as 29°. This gaze data would not be suitable for eye tracking without some form of smoothing or outlier rejection.

We were surprised to find that an unoptimised version of the model gave good gaze data, both in terms of mean and variance. This is the model after only the initialisation, before applying any form of optimisation.

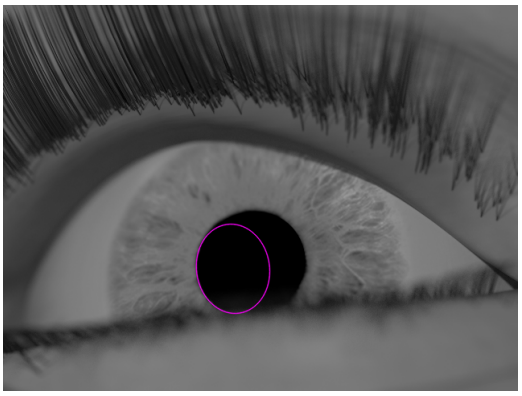

Figure 12. Example of a poor initial pupil ellipse fit, with a gaze error of 29.1°.

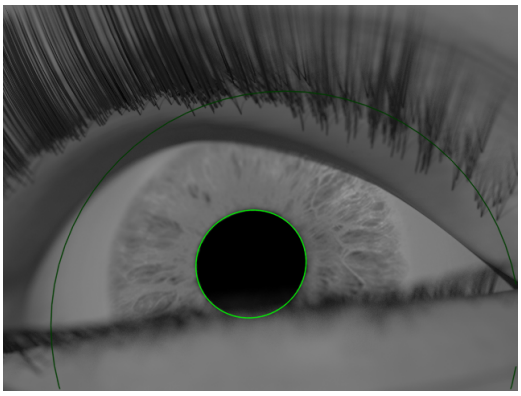

Figure 13. The ellipse fit from Figure 12 after region contrast maximisation. The ellipse reprojection error is 1.77 px, and the gaze error is 2.25°. The dark green circle is the outline of the optimised model's sphere.

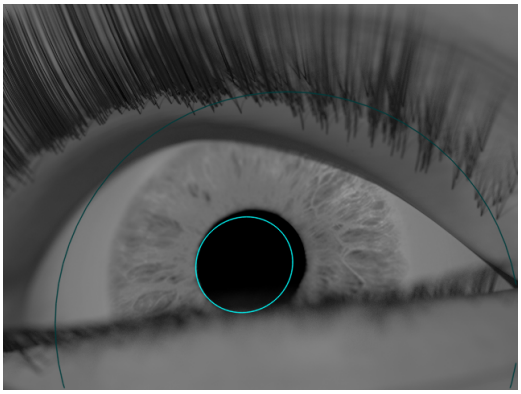

Figure 14. The ellipse fit from Figure 12 after edge distance minimisation. Note that in the original pupil fit, the left side of the ellipse was a good fit, while the right side was a poor fit. This is also the case in the optimised ellipse, as the quality of the edge distance minimisation is limited by the output of the pupil tracker. However, the optimised ellipse is still a better fit due to to the constraints imposed by the model and the other images in the dataset.

We believe that this is in large part due to a good initialisation of the sphere centre and radius, and the last stage of the initialisation which discards gaze information and only uses pupil centre information. While the ellipse shape may be wrong, and thus give bad gaze data, the ellipse does tend to be in vaguely the right area. By discarding the gaze in the latter stage of the initialisation, we obtain a gaze estimate based on the sphere radial vector rather than the ellipse shape, and this tends to be more reliable.

However, the optimisation stage reduces the error further. We found that on this dataset, the edge distance minimisation gave a lower mean gaze error than the region contrast maximisation (Table 1). We are not certain why this is the case, however we suspect that it is a case of over-optimisation. The lower variance of the region contrast maximisation gaze error supports this hypothesis.

We also investigated the reprojection error of the pupils, in pixels. We used the Hausdorff distance, as described by Świrski et al. (2012), to compare projections of the pupils against the ground truth. We also calculated the ellipse fit error of the output of the initial pupil tracking stage. Figure 11 shows a graph of these results, and Table 2 shows a summary.

We were not surprised that the unoptimised model had higher reprojection error than the pupil tracker. Interestingly, however, the optimised model had lower reprojection error than the pupil tracker, for both optimisation metrics. It is not immediately obvious that this should be the case; the pupil tracker is designed only to find and fit the pupil ellipse, whatever size or shape it may be, while our model imposes additional constraints on the possible shapes and locations of the pupils. Thus, we would expect the pupil tracker to over-fit the pupil ellipse. However, we have found that, in the case of bad pupil information such as occlusions or weak edges, the constraints on the optimised model provide additional information where the pupil tracker has none, and thus limit the magnitude of the error (fig. 13 and 14).

Of the two optimisation metrics, we found that the region contrast maximisation had a smaller mean reprojection error than the edge distance minimisation, as well as a far smaller variance. This is because the edge distance minimisation metric uses the output of the pupil tracker; namely, the list of edge points used to fit the original ellipse. When there is a bad pupil fit, this information is poor, and so the edge distance minimisation will also give poor results (fig. 14). The region contrast minimisation, on the other hand, works directly on the image pixels, and so is independent of the pupil tracker (fig. 13).

These pupil reprojection results present an interesting insight when combined with the gaze error results. Although the region contrast maximisation has a lower reprojection error than the edge distance minimisation, it has a higher gaze angle error. Similarly, the pupil tracking has a lower ellipse error than the reprojected unoptimised model, yet the unprojected ellipses have a higher gaze error. It appears therefore that an improvement in pupil ellipse fitting does not necessarily correlate with an improvement in gaze accuracy, and that the dominant factor in establishing a good gaze vector is finding the correct sphere centre rather than a good fit of the pupil contour.

Conclusion

We have presented a novel algorithm for fitting a 3D pupil motion model to a sequence of eye images, with a choice of two different metrics for fitting the model. Our approach does not require user calibration, calibrated lighting, or any prior on relative eye position or size, and can estimate gaze to an accuracy of approximately 2° (see Table 1).

We have also introduced a novel eye image data set, rendered from simulation, which provides a perfect ground truth for pupil detection and gaze estimation. We have evaluated our algorithm on this dataset, and have found it to be superior in terms of gaze angle error to previous approaches which did not constrain the pupil motion. We have also found that the 2D pupil ellipses calculated as a side-effect of the model fitting are superior to the ellipses found by current pupil tracking algorithms, but that a better ellipse fit does not necessarily correlate with better gaze estimation, and that even poor ellipse fits result in a surprisingly good gaze vector.

We believe that our algorithm, in particular the region contrast minimisation optimisation, is close to the limit of how good an ellipse fitting approach can be, especially given issues such as depth-of-field blur or the fact that real pupils are not perfectly circular. Despite this, the gaze error is relatively high compared to calibration-based approaches. Therefore, we believe that our algorithm approaches the limit of accuracy one can obtain purely from a geometric analysis of the pupil, and any significant improvements in gaze accuracy can only be obtained through collecting additional data (such as user calibration) rather than by improving the pupil ellipse fit.

Although our approach does not achieve the gaze accuracy of systems that include user calibration, our approach is accurate enough for approximate gaze estimation. Furthermore, our pupil motion model could be used as part of a user-calibrated gaze estimation; as an initialisation, a first pass approximation, or for regularisation.

References

Agarwal, S., & Mierle, K. (n.d.). Ceres Solver: Tutorial & Reference [Computer software manual].

Agustin, J. S., Skovsgaard, H., Hansen, J. P., & Hansen, D. W. (2009). Low-cost gaze interaction: ready to deliver the promises. In *Proc. chi ea* (pp. 4453–4458). ACM.

Chau, M., & Betke, M. (2005). *Real Time Eye Tracking and Blink Detection with USB Cameras.*

Daugman, J. (2004, January). How Iris Recognition Works. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(1), 21–30. doi: 10.1109/TCSVT.2003.818350

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (2002). *Texturing and Modeling: A Procedural Approach* (3rd ed.). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hansen, D. W., & Ji, Q. (2010, March). In the eye of the beholder: a survey of models for eyes and gaze. *IEEE Trans. PAMI*, *32*(3), 478–500. doi: 10.1109/TPAMI.2009.30

Holmberg, N. (2012). *Advance head rig.* Retrieved from `http://www.blendswap.com/blends/view/48717`

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Springer.

Rall, L. B. (1981). *Automatic Differentiation: Techniques and Applications* (Vol. 120). Berlin: Springer.

Safaee-Rad, R., Tchoukanov, I., Smith, K., & Benhabib, B. (1992). Three-dimensional location estimation of circular features for machine vision. *IEEE Transactions on Robotics and Automation*, *8*(5), 624–640. doi: 10.1109/70.163786

Schnieders, D., Fu, X., & Wong, K.-y. K. (2010). Reconstruction of Display and Eyes from a Single Image. In *Proc. cvpr* (pp. 1442–1449).

Świrski, L., Bulling, A., & Dodgson, N. A. (2012, March). Robust real-time pupil tracking in highly off-axis images. In *Proceedings of etra.*

Tsukada, A., Shino, M., Devyver, M. S., & Kanade, T. (2011). Illumination-free gaze estimation method for first-person vision wearable device. *Computer Vision in Vehicle Technology*.

Zhao, H., Chan, T., Merriman, B., & Osher, S. (1996). A variational level set approach to multiphase motion. *Journal of computational physics*, *195*, 179–195.