Rendering synthetic ground truth images for eye tracker evaluation

Lech Świrski*   Neil Dodgson†

University of Cambridge

Abstract

When evaluating eye tracking algorithms, a recurring issue is what metric to use and what data to compare against. User studies are informative when considering the entire eye tracking system, however they are often unsatisfactory for evaluating the gaze estimation algorithm in isolation. This is particularly an issue when evaluating a system's component parts, such as pupil detection, pupil-to-gaze mapping or head pose estimation.

Instead of user studies, eye tracking algorithms can be evaluated using simulated input video. We describe a computer graphics approach to creating realistic synthetic eye images, using a 3D model of the eye and head and a physically correct rendering technique. By using rendering, we have full control over the parameters of the scene such as the gaze vector or camera position, which allows the calculation of ground truth data, while creating a realistic input for a video-based gaze estimator.

CR Categories: I.6.5 [Simulation and Modeling]: Model Development—Modeling methodologies; I.3.8 [Computer Graphics]: Applications;

Keywords: ground truth, pupil detection, eye tracking, rendering

1 Introduction

There is a wide and increasing range of eye tracking techniques [Hansen and Ji 2010]. With such a large body of literature, being able to evaluate the techniques well becomes critical. This is necessary for progress in both the field of eye tracking and in fields which use eye tracking as a tool.

However, it is not immediately obvious how to evaluate eye tracking algorithms, neither in terms of what metric to use nor what ground truth data to compare against. Furthermore, most video-based eye trackers consist of several stages, such as pupil detection, head pose estimation and gaze estimation, and evaluating these components independently is difficult. In this paper, we describe existing approaches to eye tracker evaluation, and present our own approach based on rendered images.

Our system uses modern computer graphics techniques to render highly realistic, physically correct eye images. These images are calculated using a parametric eye tracker model, which is compatible with exising geometric models of eye tracking systems. This combination of matching geometric model and eye images can be used for evaluating eye tracking algorithms.

*e-mail: lech.swirski@cl.cam.ac.uk
†e-mail: neil.dodgson@cl.cam.ac.uk

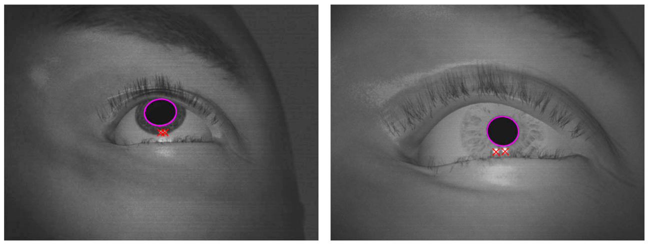
Figure 1: *Synthetic eye images, with ground truth pupil and glints.*

2 Background

Since it is not immediately obvious how to evaluate eye tracking algorithms, there have been several different approaches to evaluation. We summarise these below.

2.1 User studies

The most common approach to evaluating an eye tracking system is to run a user study to measure average gaze accuracy. This form of evaluation is informative, as ultimately what matters to the user of an eye tracking system is how accurate the gaze is when used by real users. However, it is also incomplete as an evaluation of a gaze estimation algorithm, as it evaluates the entire system as a black box. This means that there is no way of inferring the source of error, for example whether improving the pupil tracking would have a noticeable effect on the overall accuracy.

There are also many confounding variables in a user study which are almost impossible to control or account for. It is not feasible to control or measure every single such variable in a user study, and even if one did measure them, there may be error in this measurement. Thus, it is almost impossible for a user study to evaluate a gaze estimation algorithm on its own merit.

Furthermore, user studies are time consuming and require many participants to give statistically meaningful results. This time investment may be acceptable when performing a final evaluation of a system, but user studies are not feasible for testing during the development process. Lastly, user experiments require eye tracking hardware which is either expensive or built in-house, raising questions about the external repeatability of the experiment.

2.2 Human labelling

Another common approach to collecting ground truth is human labelling, where humans manually annotate images with ground truth data. Most commonly, this is used to label the pupil contour ellipse when evaluating pupil detection algorithms [Tsukada et al. 2011; Świrski et al. 2012]. The human labelled ellipses are compared to the output of the pupil detection algorithm, for example by measuring their overlap ratio or the Hausdorff distance between them. Human labelling has also been successfully used for optical flow datasets [Donath and Kondermann 2013].

This approach assumes that humans can identify the pupil contour perfectly. This assumption appears to be reasonable — Donath and Kondermann report an average error on the order of 0.5 px — although it is not clear if this continues to be the case for blurred or partially occluded ellipses. It is also better if multiple humans label the same contours and the results are averaged. This avoids individual bias and removes outliers, and crowdsourcing services such as Amazon Mechanical Turk allow simple tasks to be done cheaply and in parallel, making this cost negligible.

Human labelling can be considered a high quality source of ground truth for tasks such as pupil detection or eyelid detection — tasks which require labelling of visible 2D features. It is not, however, suitable as a ground truth of other types of data, such as 3D gaze vectors, eyeball centre, or heavily occluded pupils where the pupil tracking must rely on temporal information alone.

2.3 Artificial eyes

Some approaches are evaluated using artificial eyes [Moore et al. 1996; Zhu et al. 1999; Clarke et al. 2002; Imai et al. 2005], where a physical artificial eye attempts to mimic the appearance of a real eye. This artificial eye has known size, and its position is controlled by motors. Given some form of extrinsic calibration, its position relative to a camera can be calculated.

This gives the experimenter ground truth data on eye rotation and position, while maintaining realistic camera and light behaviour. However, the artificial eye is not necessarily a good model of a real eye. The simple construction of existing artificial eyes does not model glints, corneal refractions or varying pupil size, nor does it model the eyelids or eyelashes which occlude the eye. Also, using a physical object incurs some amount of measurement error, which will be indistinguishable from error in the gaze estimation itself.

2.4 Geometrical models

To avoid issues with physical measurements, one can perform a purely geometrical analysis of the eye tracking model to calculate a theoretical lower bound on the gaze estimation error, and to analyse the effect of adding a known amount of noise to the system [Wang et al. 2005; Villanueva et al. 2006]. Of particular interest is the simulation framework introduced by Böhme et al. [2008], which simulates many aspects of a ‘standard’ eye tracking system, including corneal refraction, finite camera resolution and location of glints.

Böhme et al. argue that geometric modelling can objectively analyse the performance of eye tracker set-ups, in particular when comparing two systems against each other, as it allows one factor (such as light positions) to be changed, while keeping everything else constant. Also, unlike a user study, one has ground truth for internal values of the system, such as projected pupil contour points or gaze vector in camera-space.

However, geometrical modelling does not perform image processing, injecting Gaussian noise to simulate the error. Böhme et al. refer to this random perturbation as the “feature position error”. The problem with this approach is that it is not a realistic model of the real error, and is biased towards approaches which assume that all noise is Gaussian. Current simulations are also limited in what they model; these limitations could be overcome by a sufficiently complex model, but still geometrical modelling does not allow one to evaluate image processing algorithms, as they do not generate images. This is a severe limitation.

2.5 Synthetic images

To allow one to evaluate image processing while maintaining the control that one has in a geometrical model, one can generate synthetic images of an eye based on the geometrical model using computer graphics [Morimoto and Mimica 2005]. As with human labelling, synthetic images have successfully been used as ground truth in optical flow [Baker et al. 2010].

Synthetic images are, in many ways, complementary to geometrical models. They use the same data as a geometrical model, such as eye position, light position or camera focal length, but instead of directly using a mathematical projection model to calculate image points, they generate images using some form of computer graphics. This allows a far more accurate model than Gaussian noise of things like camera distortion, focal blur or finite image resolution, as these sources of noise are introduced implicitly by the rendering algorithm. This provides input to the eye tracker evaluation, while the geometrical models can be used as before — without adding noise — to generate the ground truth data.

Modern computer graphics, thanks to advances in algorithms and in hardware, has achieved a level of fidelity which makes rendered images nearly indistinguishable from real life. Furthermore, physically correct rendering techniques allow the rendering of images which are visually plausible and accurately model real-world light and surface behaviour.

We describe a system which, given input model parameters similar to those used by Böhme et al., uses path-tracing to renders realistic synthetic images of the eyeball and surrounding facial structure. Our system and example scripts and images are available online[1].

3 Our simulator

Current simulation approaches do not model the parts of the head surrounding the eye, such as the eyelids, eyelashes, and surrounding skin, which limits their use when evaluating image processing algorithms. Eyelids and eyelashes occlude parts of the eye, and can cause simple techniques to fail if they are not robust to occlusions; similarly, the skin and shadows upon it may have similar intensity profiles to the pupil or iris, again causing simple methods to fail. Furthermore, some techniques rely on there being eyelids, for example using the eye corners to compensate for head movement [Hansen and Ji 2010].

For these reasons, we decided to build a 3D model which includes both the eye and the surrounding facial structure — in fact, we model the entire head, as this allows us to model both mobile and remote eye tracking systems. We also wanted to use a physically correct rendering method, which would be able to correctly handle reflections, refraction, shadows and depth-of-field blur.

To generate our synthetic images, we use Blender [2013], an open-source 3D computer graphics software product which allows both modelling and rendering of 3D scenes. Using Blender had several benefits. It includes a GPU implementation of path-tracing [Kajiya 1986], a physically correct rendering method that can render photographic-quality images in less then a minute, and preview-quality images at interactive rates. It is also a very powerful modelling and animation tool, which allows us to build a realistic head rather than a simple sphere-based eye model. Additionally, it has an embedded Python interpreter, allowing us to script the positions of objects. This is critical when generating a large amount of test data, without requiring a user to manually edit the model.

3.1 The model

We start with an existing public domain head model [Holmberg 2012]. This model allows control of the position of the eyes, the eyelids (Fig. 2), the radius of the pupil (Fig. 3), and the positions of a variety of facial features that are mostly irrelevant to eye tracking.

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

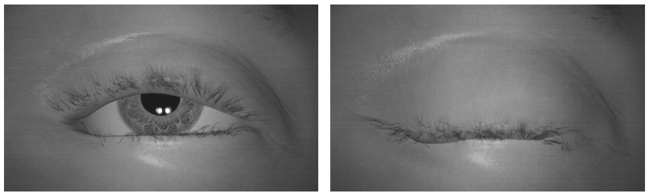

Figure 2: *The user has control over the positions of the eyelids, allowing simulation of blinks.*

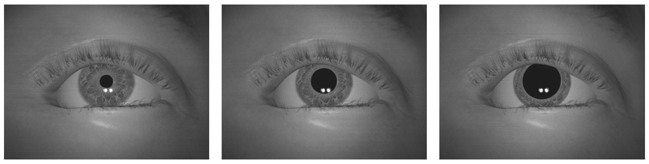

Figure 3: *Users can vary the dilation of the pupil. Note that the iris texture stretches to match the pupil dilation.*

We modified this model to make it better suited for generating synthetic eye images: we retextured the head and iris to have the appearance of being under near-infrared illumination, as this is what a large proportion of video oculography techniques use; we remodelled the eye and cornea to make them consistent with Listing's reduced eye model, including making the iris flat and the cornea spherical; we improved the realism of the cornea, complete with refraction and glints, and allowing the user to change the corneal index of refraction (Fig. 5); and we added eye lashes, using Blender's directable hair particle modelling. We allow the user to alter the iris colour by allowing them to change the iris texture (Fig. 4).

Most systems use near-infrared (NIR) LED light sources for illumination — additionally, many systems use the corneal reflections of the LEDs (i.e. the glints) as part of the eye tracker algorithim. It is therefore important to model these light sources realistically. We model the LEDs as spheres which emit light directionally, with some viewing angle (we use $45°$ by default). Using directed lights is a more accurate model of real LEDs than omni-directional point lights. Additionally, our lights have an adjustable non-zero size — again, this is a better approximation of real LEDs than infintessimaly small point lights, as the corresponding corneal glint is not an idealised single point, but an area with some shape.

We also perform a post-render compositing step which adds a small amount of random shot noise and line noise to the image. This provides a simulation of standard camera noise.

3.2 Generating images

One of the major advantages of using a simulation framework, like that of Böhme et al., is that simulation parameters can scripted. This allows one to, for example, test an algorithm against a variety of eye tracking setups. Our model also allows this. We have written a Python library which sets the model parameters and renders the model using Blender. This library allows the user to set: eye radius, position and orientation (3 DoF); how closed the eye is; iris texture; cornea refractive index; pupil dilation; camera position and orientation; camera focal length; camera F-number (for depth of field); image size; and lights (position, orientation, viewing angle, size and intensity).

Once these are set, the user can call the `render` function, and the library passes these values to Blender, which appropriately modifies the model to use these parameters and renders a single frame. The majority of these have default values based on anthropomorphic averages and standard webcam parameters, so a user can easily generate images by just setting camera position, orientation and adding a couple of lights. All of the images in this section were generated using this library.

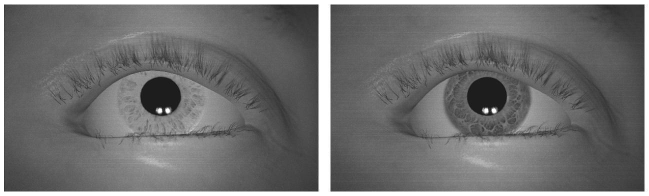

Figure 4: *We include two iris textures with the model, light and dark. Users can also specify their own iris texture.*

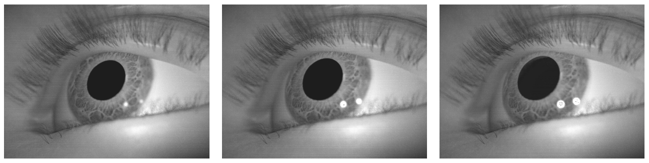

Figure 5: *Users can vary the index of refraction of the cornea. Note that this changes the size of the glints — this is realistic physical behaviour, governed by the Fresnel equations.*

We chose our parameters and eye model to be compatible with the model used by Böhme et al. (Fig 6). This allows us to use their framework to calculate ground truth image coordinates for the pupil contour and glints (Fig 1). We do this by generating a MATLAB script compatible with the Böhme et al. simulator, which initialises an eye, camera and lights so that their parameters match those used in Blender. This is how Figures 1 and 6 were generated.

3.3 Limitations

Although our model is highly realistic, it has several limitations. Firstly, our eye model has, by design, similar limitations to the Böhme et al. simulator: we model neither the aqueous humour nor the lens, which means that we cannot render Purkinje images [Crane and Steele 1978]; we use a perfectly spherical cornea, while a real cornea is slightly ellipsoidal; and we assume an immobile pupil. Böhme et al. [2008] discuss these limitations — we maintain the same limitations to be compatible with their simulator, although we could improve the Blender model to remove them.

Furthermore, we currently cannot dynamically change the pupil shape, iris radius, cornea radius or interocular distance — however, these can be edited manually. We also do not model the inside of the eye, and therefore cannot synthesise bright-pupil images. Nor do we implement Listing's law [Haustein 1989], as it does not necessarily always apply.

We render each frame individually, rather than as an animation. This means that we do not have any motion blur, which is a source of error when tracking the eye during saccades. Also, the small amount of post-render camera noise that we add is plausible, but has no physical basis. It would be better to model or get samples of real camera noise, and apply that instead. Finally, our model is limited to one head shape, with the facial structure of a white male, and therefore has limited use if trying to analyse differences between races and sexes.

4 Discussion

We have presented a model that uses computer graphics and physically correct rendering techniques to synthesise parametrised eye-

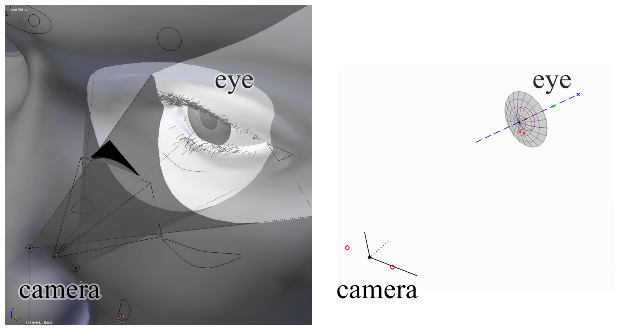

Figure 6: *Our model (left) uses similar parameters to Böhme et al., which means that we can pass our parameters into their* MATLAB *simulation framework (right).*

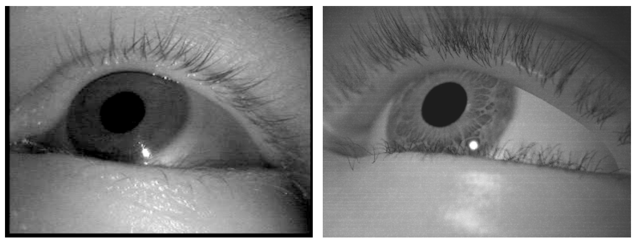

Figure 7: *An image of a real eye (left), and a rendered image from our model (right).*

and-head images, which can be used alongside ground truth data calculated using an existing geometric simulation framework.

We believe that our images are sufficiently realistic that they can plausibly be used for the evaluation of the image processing side of eye tracking algorithms. Although the use of realistic synthetic images was not feasible in the past, due to the rendering time being measured in hours, advances in computer graphics and in hardware have solved this issue, and no image in this paper took longer to render than one minute.

We can only repeat the sentiments of Böhme et al. on the advantages of using simulation for the evaluation and development of eye tracking systems. Simulation allows us to calculate ground truth data which is entirely correct, and does not rely on external measurement which will always entail some amount of error; it allows us to control parameters which are implausible or even impossible to control in a real user study; we can run exactly the same experiment with exactly the same parameters as many times as we want, making the evaluation externally reproducible; and we can do all of this without having to leave our desks, organise participants or even buy hardware. We are currently using this system for developing and evaluating our work, and we have already found it invaluable during the development process for testing as well as debugging.

We have not presented an evaluation of the realism of our images, as it is not obvious how to evaluate them — again we encounter the problem of ground truth. Ideally, there would be an existing ground truth dataset with known parametrisation, but we are not aware of such a dataset for eye images. Our rendered images appear to agree with the data from the Böhme et al. simulator, and qualitatively appear realistic. In Figure 7, we compare an real eye image, captured using a *Dikablis* system, with an image rendered by our model. Upon close inspection, it is clear which image is real and which is synthetic, however we claim that it is not obvious at first glance, and that the difference is much smaller than with previous simulation approaches.

References

BAKER, S., SCHARSTEIN, D., LEWIS, J. P., ROTH, S., BLACK, M. J., AND SZELISKI, R. 2010. A Database and Evaluation Methodology for Optical Flow. *International Journal of Computer Vision 92*, 1 (Nov.), 1–31.

BLENDER FOUNDATION, 2013. Blender 2.69. `http://www.blender.org/`.

BÖHME, M., DORR, M., GRAW, M., MARTINETZ, T., AND BARTH, E. 2008. A Software Framework for Simulating Eye Trackers. In *Proc. ETRA*, no. 212.

CLARKE, A. H., DITTERICH, J., DRÜEN, K., SCHÖNFELD, U., AND STEINEKE, C. 2002. Using high frame rate CMOS sensors for three-dimensional eye tracking. *Behavior research methods instruments & computers 34*, 4, 549–560.

CRANE, H. D., AND STEELE, C. M. 1978. Accurate three-dimensional eyetracker. *Applied optics 17*, 5 (Mar.), 691–705.

DONATH, A., AND KONDERMANN, D. 2013. Is Crowdsourcing for Optical Flow Ground Truth Generation Feasible? In *Proc. ICVS*, Springer, M. Chen, B. Leibe, and B. Neumann, Eds., vol. 7963 of *Lecture Notes in Computer Science*, 193–202.

HANSEN, D. W., AND JI, Q. 2010. In the eye of the beholder: a survey of models for eyes and gaze. *IEEE Trans. PAMI 32*, 3 (Mar.), 478–500.

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.

HOLMBERG, N., 2012. Advance head rig. `http://www.blendswap.com/blends/view/48717`.

IMAI, T., SEKINE, K., HATTORI, K., TAKEDA, N., KOIZUKA, I., NAKAMAE, K., MIURA, K., FUJIOKA, H., AND KUBO, T. 2005. Comparing the accuracy of video-oculography and the scleral search coil system in human eye movement analysis. *Auris Nasus Larynx 32*, 1, 3–9.

KAJIYA, J. T. 1986. The rendering equation. *Computer Graphics 20*, 4, 143–150.

MOORE, S. T., HASLWANTER, T., CURTHOYS, I. S., AND SMITH, S. T. 1996. A geometric basis for measurement of three-dimensional eye position using image processing. *Vision research 36*, 3 (Feb.), 445–459.

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

ŚWIRSKI, L., BULLING, A., AND DODGSON, N. 2012. Robust real-time pupil tracking in highly off-axis images. In *Proc. ETRA*.

TSUKADA, A., SHINO, M., DEVYVER, M. S., AND KANADE, T. 2011. Illumination-free gaze estimation method for first-person vision wearable device. *Computer Vision in Vehicle Technology*.

VILLANUEVA, A., CABEZA, R., AND PORTA, S. 2006. Eye tracking: Pupil orientation geometrical modeling. *Image and Vision Computing 24*, 7 (July), 663–679.

WANG, J.-G., SUNG, E., AND VENKATESWARLU, R. 2005. Estimating the eye gaze from one eye. *Computer Vision and Image Understanding 98*, 1, 83–103.

ZHU, D., MOORE, S. T., AND RAPHAN, T. 1999. Robust pupil center detection using a curvature algorithm. *Computer methods and programs in biomedicine 59*, 3 (June), 145–57.