UNIVERSITY OF CAMBRIDGE

Gaze Estimation with Graphics

Erroll Wood

Computer Laboratory
Gonville and Caius College

17th September 2017

This dissertation is submitted for
the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This dissertation does not exceed the regulation length of 60 000 words, including tables and footnotes.

Abstract

Gaze estimation systems determine where someone is looking. Gaze is used for a wide range of applications including market research, usability studies, and gaze-based interfaces. Traditional equipment uses special hardware. To bring gaze estimation mainstream, researchers are exploring approaches that use commodity hardware alone. My work addresses two outstanding problems in this field: 1) it is hard to collect good ground truth eye images for machine learning, and 2) gaze estimation systems do not generalize well – once they are trained with images from one scenario, they do not work in another scenario.

In this dissertation I address these problems in two different ways: learning-by-synthesis and analysis-by-synthesis. *Learning-by-synthesis* is the process of training a machine learning system with synthetic data, i.e. data that has been rendered with graphics rather than collected by hand. *Analysis-by-synthesis* is a computer vision strategy that couples a generative model of image formation (synthesis) with a perceptive model of scene comparison (analysis). The goal is to synthesize an image that best matches an observed image.

In this dissertation I present three main contributions. First, I present a new method for training gaze estimation systems that use machine learning: learning-by-synthesis using 3D head scans and photorealistic rendering. Second, I present a new morphable model of the eye region. I show how this model can be used to generate large amounts of varied data for learning-by-synthesis. Third, I present a new method for gaze estimation: analysis-by-synthesis. I demonstrate how analysis-by-synthesis can generalize to different scenarios, estimating gaze in a device- and person- independent manner.

Acknowledgements

Many people have supported me during my PhD. I would like to thank three in particular. I am most grateful to my supervisor Peter Robinson who has been a constant source of inspiration, guidance, and common-sense since I joined Gonville & Caius eight years ago. I am also indebted to Andreas Bulling whose enthusiasm and energy drove me towards goals I would never otherwise have thought possible. I will miss our frequent chats over Skype. I would also like to thank Tadas Baltrusaitis who has been more than just an excellent collaborator and role-model scientist, but also an invaluable friend.

I could not imagine a better place to do a PhD than the Computer Lab. For their guidance, I'd like to thank Neil, Alan, Rafal, Graham, and Lise. Special thanks go out to Richard who helped with the maths in Chapter 4, Brian who was always there to assist with hardware, and Christian who first supervised me in Graphics and Computer Vision. For their friendship, I'd like to thank Ian, Meredydd, Marwa, Jingjing, Flora, and Sam. Many of my fondest memories in the lab came from being a part of the Happy Hour crew. Thanks to Leszek, Vaiva, Ollie, Advait, and Tamas for all the good times, conversations, and beers we shared. An encyclopedic knowledge of Milton Brewery's selection will forever be burned into my mind.

This work would not have been possible without the generous financial support of the EPSRC, University of Cambridge Computer Lab, and Gonville & Caius College. To each I am grateful.

I was fortunate enough to enjoy an internship at Microsoft, so I'd like to extend thanks to those who supported me there, including Jamie Shotton, Andrew Fitzgibbon, Tom Cashman, and Jonathan Taylor. Indeed, it is their work on hand tracking that inspired chapters 6 and 7. Best of all, I now find myself back in the same team, and feel lucky to be able to call these wonderful people my colleagues.

I am deeply grateful to my family for their constant love and encouragement. I thank my father for passing on to me his love of maths and computing, and I thank my mother for teaching me the value of hard work. Lastly, I would like to thank Maeve. Thanks to your love, kindness, patience, and good humor, the time we shared together during our PhDs has been the happiest time of my life.

Publications

This dissertation includes work published in the following:

- Wood, E., Baltrušaitis, T., Zhang, X., Sugano, Y., Robinson, P., & Bulling, A. (2015, December). **Rendering of eyes for eye-shape registration and gaze estimation.** In 2015 IEEE International Conference on Computer Vision (ICCV) (pp. 3756-3764).

- Wood, E., Baltrušaitis, T., Morency, L. P., Robinson, P., & Bulling, A. (2016, March). **Learning an appearance-based gaze estimator from one million synthesised images.** In Proc. of the ACM Symposium on Eye Tracking Research & Applications (pp. 131-138).

- Wood, E., Baltrušaitis, T., Morency, L. P., Robinson, P., & Bulling, A. (2016, May). **A 3D Morphable Model of the Eye Region.** Eurographics Conference 2016 (poster).

- Wood, E., Baltrušaitis, T., Morency, L. P., Robinson, P., & Bulling, A. (2016, October). **A 3D morphable eye region model for gaze estimation.** In European Conference on Computer Vision (pp. 297-313). Springer.

Other publications from work not included in this dissertation:

- Wood, E., & Bulling, A. (2014, March). **Eyetab: Model-based gaze estimation on unmodified tablet computers.** In Proc. of the ACM Symposium on Eye Tracking Research and Applications (pp. 207-210).

- Wood, E., & Robinson, P. (2014, November). **NetBoards: Investigating a Collection of Personal Noticeboard Displays in the Workplace.** In Proc. of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (pp. 177-183).

- Taylor, J., Bordeaux, L., Cashman, T., Corish, B., Keskin, C., Sharp, T., ... & Shotton, J. (2016, July). **Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences.** ACM Transactions on Graphics (TOG), 35(4), 143.

- Wood, E., Taylor, J., Fogarty, J., Fitzgibbon, A., & Shotton, J. (2016, November). **ShadowHands: High-Fidelity Remote Hand Gesture Visualization using a Hand Tracker.** In Proc. of the 2016 ACM conference on Interactive Surfaces and Spaces (pp. 77-84).

Contents

1

Introduction

Gaze estimation is the process of determining where someone is looking, either as a gaze direction or as a point in space. As humans, we estimate gaze all the time. Though we are not always aware of it, we constantly monitor where other people are looking in social settings. This is because our eyes reveal so much about us. Indeed, the eyes are often said to be the window to the soul. Since we must turn our eyes towards something to see it clearly, our eyes indicate who or what we are interested in, or paying attention to. As a result, gaze and eye contact represent a major non-verbal channel for social signals; being used to regulate interaction, express intimacy, and exercise social control (Kleinke, 1986).

Researchers nowadays can use special eye tracking equipment to record where people are looking. The main users of eye tracking are researchers interested in human behaviour. Psychologists use eye tracking to investigate how we think. They want to know how and why eye movements are made, for example in relation to learning, memory, or problem solving. Marketing agencies use eye tracking to measure attention and response to marketing messages. They need to know what visual elements are noticed and which are ignored, so they can design products or adverts that catch the consumers's eye. User interface designers use eye tracking to evaluate different interfaces with the goal of improving user experience. Gaze data helps them address usability issues by revealing which features quickly draw a user's attention, and which are overlooked.

Though currently available eye tracking systems work well for these tasks, limitations have prevented widespread adoption. Commercial eye tracking equipment uses infrared lights and cameras to control

how subjects are illuminated and recorded – these cannot be found on typical everyday devices. Commercial eye trackers also do not work reliably in real world settings, so researchers must bring participants into a controlled environment to track their gaze. Consequently, eye tracking is limited to researchers with a suitable equipment budget who are willing to collect gaze data in person.

By *"commodity device"* I mean something easy to obtain off-the-shelf in a high street store.

If we could instead estimate eye gaze accurately, reliably, and conveniently using the sensors in a **single commodity device**, e.g. a typical webcam, eye tracking could be brought to the masses. But what amount of accuracy, reliability, and convenience is required?

Accuracy is commonly reported in angular degrees as the average difference between true and estimated gaze directions.

Accuracy With regards to accuracy, different applications of gaze estimation have different requirements. Some applications like gaze-based interaction with centimetre-sized icons require high accuracy ($<1°$), while others like emotion analysis or user engagement detection have lower accuracy requirements ($\sim 10°$). In this dissertation, I focus on methods that aim to cross this upper threshold of $10°$.

"In the wild" means outside a controlled environment.

Reliability Designing a computer vision system that works "in the wild" is challenging. If gaze estimation is to become mainstream, it must work under a range of different illumination conditions, both indoors and outdoors. Furthermore, the location and calibration of cameras vary a lot between different devices. A user viewed from a smartphone camera can appear very different from a user viewed from a laptop webcam. Gaze estimation algorithms should work across different devices. My work makes addressing environmental and device variation a top priority.

Convenience For gaze estimation to become adopted by the general public, convenience will be key. If we examine how people use smartphones, we notice they are willing to perform a calibration session every so often, e.g. for their fingerprint sensor or voice recognition model. But if they had to complete a gaze estimation calibration procedure every time they unlocked their phone, like how an eye tracker requires calibration when it is turned on, they would probably not use gaze estimation at all. Therefore, my work focusses on approaches that can be used without calibration, or calibration-free.

In the rest of this chapter, I first explain why the future of gaze estimation is so important. I then present two major outstanding problems in the field, and describe how I addressed them. I finally make the specific contributions of my work clear, and end with an outline of the dissertation as a whole.

1.1 Motivation

Research in visible light remote gaze estimation is driven by a range of important applications.

Mass-scale market research The ability to measure consumers' attention in the real world automatically would be very valuable indeed. Many of the internet services we take for granted are paid for by advertising. These services could make adverts more valuable by tracking the attention of millions of everyday visitors using their own webcams. This could translate into consumers seeing fewer, more effective adverts.

Mass-scale usability studies A popular way to compare different interface designs is *A/B testing* – randomly serve millions of users one of two different versions of an interface and measure the efficacy of each version. Gaze information could complement traditional metrics to help designers understand why a certain interface performed better than another.

Gaze-based interaction For some severely disabled people, eye movements are the only way they can communicate (Debeljak et al., 2012). Currently, assistive technology for these people is expensive. If individuals with special needs could control cheap off-the-shelf devices with their eyes, they would have greater opportunities for communication.

Facial performance capture Compelling virtual actors can make a big difference to a player's engagement with a video game. Small film and video game studios cannot afford professional facial capture setups, so webcam-based solutions have been proposed (Cao et al., 2014a). Since eye gaze is an important part of any facial performance it should be captured along with the rest of the face. Beyond games, facial performance capture is also important for remote communication with lifelike 3D avatars.

Performance capture is the process of transferring video footage of an actor onto a 3D model in a realistic fashion.

Affective computing Since emotions are so important in the way people interact with each other, it is believed that enabling computers to sense our emotional states would be beneficial. Applications could range from a video game that got easier if the user becomes frustrated, to a medical system for automatically diagnosing complex mental illnesses like depression. Much work has been done on tracking parts of the face to try and measure emotion – eye gaze is a key part of this.

Attentive user interfaces are interfaces that manage the user's attention (Vertegaal et al., 2003).

Attentive user interfaces We are surrounded by digital devices that constantly bombard us with notifications, vying for our attention. As displays become more ubiquitous, this will get worse, and our attention spans might shrink even further. To combat this, devices of the future should continuously measure and manage our attention to keep us focussed on the task at hand.

Gaze-contingent displays A gaze-contingent display performs a different function depending on where the user is looking. One example is foveated rendering. Computer graphics becomes more computationally expensive as displays become higher resolution. So when a display is particularly high-resolution, e.g. in VR, it is difficult to present high quality visuals to the user over the whole display. With foveated rendering, high-quality graphics are only rendered around the user's gaze point, enabling better use of computational resources.

Post-hoc gaze estimation With eye tracking equipment, it is possible to record someone's eye movements as long as you are with them in the same place at the same time. What if this is not the case? For example, one might want to analyse eye movements in videos that have previously been collected. In this case, calibration-free visible-light–based gaze estimation can be run offline on archive footage to recover gaze data.

It should be noted that different applications have different requirements on gaze estimation accuracy and speed. Some might require high accuracy, for example usability testing of a complex interface with lots of small buttons. Others might require high framerate, for example real time interaction with a gaze-based interface. In some cases, different goals are better served with different algorithms, and trade-offs must be made.

1.2 Outstanding problems

This work addresses two outstanding problems for gaze estimation.

The first problem is that **it is difficult to collect good ground truth eye gaze data**. Machine learning methods that require large amounts of training data perform best for many computer vision tasks (LeCun et al., 2015). If you want to train such a system for gaze estimation, you need a lot of labelled images of people looking in different directions. For traditional computer vision tasks like object recognition these

images can be collected off websites (e.g. Flickr) and labelled by Mechanical Turks – human workers hired over the internet. However, this post-hoc labelling is impossible for eye gaze. As a result, researchers spend a lot of time, effort, and money building eye gaze datasets by capturing the images themselves. Furthermore, the resulting datasets can be of limted use if they were recorded in an unnatural environment like a laboratory. We therefore ask ourselves: is there a better way to collect eye gaze training data?

The second problem is that **gaze estimation systems are generally tied to a specific usage scenario**. This means that they require a specific hardware configuration, expect the subject to be lit with a certain type of illumination, and only allow for a limited amount of eye and head movement. These limitations arise because images for training gaze estimators are collected with a single scenario in mind (e.g. laptop use), and taken with a single type of camera. In some cases it may be acceptable to have an eye tracker tailored to a specific scenario, but this is less valuable than a system that you can rely on in any setting. A generic gaze estimator that operated robustly under variations in facial appearance, head pose, lighting, and camera type would be incredibly useful. Is it possible to build a generic gaze estimation system?

1.3 My approaches

To address these two problems, I explored two different approaches: learning-by-synthesis and analysis-by-synthesis.

Learning-by-synthesis is the process of training a machine learning system using synthetic data. Acquiring good training data can be time-consuming and requires accurate ground truth labels. Furthermore, manually labelling the data can be expensive and tedious, and there is no guarantee that the human-provided labels will be correct. To address these problems, researchers have used computer graphics to synthesize training data. The first step is acquiring 3D models – these should be realistic for the training data to be useful. The next step is preparing a rendering framework – this should generate ground truth information as well as images. With these, it is possible to generate cleanly-labelled training data with a fraction of the time and effort required otherwise.

Analysis-by-synthesis is a computer vision strategy that couples a generative model of image formation (synthesis) with a perceptive

model of scene comparison (analysis). The goal is to search for the best explaination of an observed image in terms of an underlying scene model. The synthesis step first uses computer graphics to render an image of a scene it suspects might explain the observed image. The analysis step then compares the synthetic and observed images, determining how likely it was that the underlying scene produced the observed image, and suggesting new scenes that might explain it better. The description of the scene includes eye gaze direction. Learning-based methods hope to learn robustness to image variation through variety in training data. Instead, analysis-by-synthesis methods model these variations explicitly.

The common theme that links these two techniques is the use of computer graphics. With learning-by-synthesis, I used graphics in an offline manner – generating realistic eye images which were used to train machine learning systems. With analysis-by-synthesis, I used graphics in an online manner – rendering images of virtual eyes so they could be compared against an observed image from the real world.

1.4 Contributions

There are three main contributions of my work.

Chapter 3

First, I investigate learning-by-synthesis for gaze estimation as an alternative to manual data collection. I demonstrate that readily-available 3D models and modern computer graphics techniques can be used to generate valuable training data for gaze estimation systems that use machine learning.

Chapters 4 and 5

Second, I present a new multi-part morphable model of the eye region. This model captures the anatomical details and movement of the eye better than previous models. I show how this model can be used to generate more varied training data for learning-by-synthesis, particularly for gaze estimation in the wild. I have made this morphable model freely available to the research community.

Chapters 6 and 7

Third, I propose a new method for eye gaze estimation: analysis-by-synthesis. By coupling an articulated 3D model of the facial eye region with a photometric model of image formation, we can recover eyeball orientation by fitting our generative scene model to an observed image. Consequently, we can estimate gaze in a person- and device-independent way.

1.5 Structure

This dissertation is structured as follows:

Chapter 1 serves as an introduction by describing the research problems faced by the field, outlining the approaches I took to address them, and making my specific contributions clear.

Chapter 2 provides an overview of gaze estimation. It describes historical and mainstream gaze tracking techniques used today. It then reviews the state of the art in visible light remote gaze estimation – the scenario I focussed on in my work.

Chapter 3 introduces a new training method for appearance-based gaze estimation: learning-by-synthesis with posable 3D models. I present the *SynthesEyes* synthetic dataset which was used to train systems for eye alignment and gaze estimation.

Chapter 4 describes a new multi-part morphable model of the eye region. I explain the shortcomings of previous morphable models of the face, and outline the construction and operation of my new more detailed model.

Chapter 5 presents *UnityEyes*, a tool for rapidly generating eye gaze datasets. Using this tool, I demonstrate the importance of good training data by showing how a lightweight learning system trained with a large number of synthetic images can outperform previous complex systems trained with real images.

Chapter 6 switches focus from learning-by-synthesis to analysis-by-synthesis. I present a new method for gaze estimation: fitting a morphable eye region model to an image using a render-and-compare energy minimization approach.

Chapter 7 extends this analysis-by-synthesis method to both eyes, and accelerates it with a GPU-based second order energy minimization strategy, allowing it to operate in real time.

Chapter 8 concludes this dissertation by summarizing the contributions of my work, discussing its limitations, and outlining possible directions for future research.

2

Eye gaze estimation

Eye movements have been studied by scientists for centuries. At first, these studies were made using direct observation. In 1879, Louis Émile Javal famously discovered that eyes do not move smoothly across text when reading, but rather make a series of short rapid movements (saccades) between short stops (fixations) (Javal, 1879). The question of what drives the eye to fixate on one location over another has occupied researchers ever since.

To obtain more accurate and objective eye movement measurements, the first instances of eye tracking technology were developed shortly thereafter (Delabarre, 1898; Huey, 1898, 1900). Figure 2.1 shows an example by Huey (1898). These early mechanical devices were invasive; they used an eye-cup that was physically attached to a metal lever that moved when the eye did. This lever moved a pen over a revolving drum, which was then photographed and engraved. However, these lever-based eye trackers applied tension on the eye, and their inertia caused eye movements to overshoot. Consequently, less invasive devices were sought. The first photographic eye tracker was developed in the early 20th Century by Dodge and Cline (1901). It didn't attach anything to the eye so was more comfortable for participants. Dodge's key insight was to photograph corneal reflections that moved when the eye did – a common technique still used by eye trackers today. Following the success of Dodge's photographic eye tracker, other scientists started to build similar devices, and the field began to evolve. Technological improvements allowed researchers to ask new questions, and study eye gaze during tasks other than reading. For example, Stratton (1902) examined how people looked at geometric patterns, exploring the links between visual phenomena, cognition, and eye movement mechanisms;

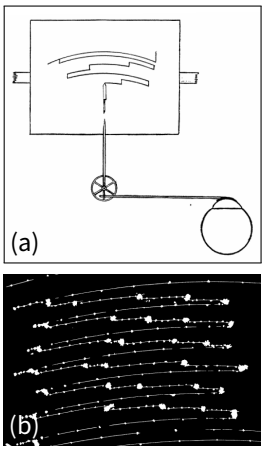


Figure 2.1: (a) The mechanical apparatus used by Huey (1898); and (b) an example engraved recording.

Eye-cups were a type of contact lens made of plaster, and required the eye to be anaesthetised with cocaine.

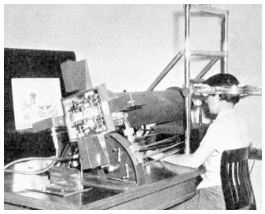


Figure 2.2: The photographic apparatus used by Buswell (1935) took up a whole desk and required a head clamp.

and Buswell (1935) (see Figure 2.2) and Yarbus (1967) investigated how people looked at pictures of complex scenes, noting that some elements were looked at more often than others.

In the 1970s and 1980s, eye tracking flourished. Eye trackers became more convenient, being able to separate head movement from eye movement (Cornsweet and Crane, 1973); and computers became fast enough to track gaze in real time, enabling gaze-based human-computer interaction (Levine, 1981). Eye tracking also moved beyond academia and became an important tool for industry, being used for evaluating interface usability and testing marketing strategy.

Of course, eye tracking technology continues to evolve today. In the past, gaze was tracked over two dimensional surfaces, be they pages, pictures, or digital screens. One field of eye tracking research is therefore in three dimensional gaze estimation (Duchowski et al., 2014) – this will allow us to better understand how we look and see in the real world or in virtual environments. A current barrier to wide-spread eye movement research is the fact that eye trackers are not yet everyday devices that you can find in your average home. Consequently, researchers are also currently investigating if eye tracking can be done with unmodified commodity devices – this would allow researchers to collect eye movement data from a larger population than ever before, improving the validity of their data, and could allow them to study a specific group of people more easily .

In this chapter, I first briefly describe the anatomy and physiology of the human eye to help understand how eye trackers operate. Following that, I put my own work in context by describing the mainstream and alternative methods for gaze estimation that are currently available, making their limitations clear. I then introduce the specific area of research my work addresses: visible light remote gaze estimation, i.e. estimating gaze using commodity RGB cameras alone. I outline the state of the art and discuss the research problems that my work addresses. Finally, I describe the gaze datasets used in my work, and explain why they were chosen.

2.1 The human eye

Before considering how eye trackers work, it is important to know a little about the human eye itself. In this section I will introduce the anatomy of the eye, and describe how it works.

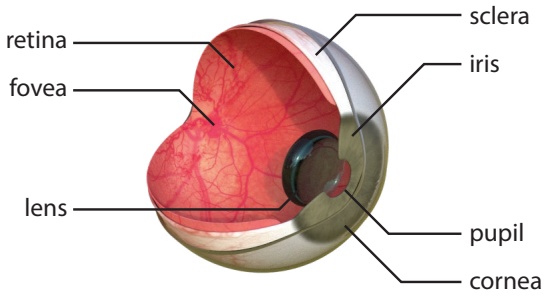


Figure 2.3: The internal anatomy of the human eye.

2.1.1 Internal anatomy

The eye is an image-forming organ. It works like a camera, taking light in from the outside world and focussing it on a plane to form an image. Light enters the eye through the *cornea* – the transparent dome at the front of the eye. It then passes through the anterior chamber and enters the circular apperture in the middle of the eye known as the *pupil*. The size of the pupil is controlled by the *iris* – a thin ring of circular and radial muscles which enlarge or contract the pupil, letting in more or less light to compensate for differences in environmental illumination. The iris gives the eye its colour through pigmentation with melanin, and can range between brown, green, grey, and blue. Light that has passed through the pupil is then focussed by the eye's lens. Unlike artificial lenses, the eye's lens is flexible so its curvature can be changed to focus at different distances.

Finally, light hits the *retina*, the photoreceptive layer of tissue on the inside of the eye. In a traditional camera, this would be the film. However, unlike film, the granularity of the retina is not uniform. Instead, the density of light sensing cells is highest at the fovea, and rapidly decreases away from it. As a result, to see things clearly we must rotate our eyeballs so that light from the object of interest falls upon the fovea. These eye movements are what we attempt to recover with eye tracking. When light strikes a photoreceptive cell in the retina, it induces nerve impulses that are sent to the brain via the optic nerve. These signals are interpreted by our brain to produce vision.

Eyes are not perfect spheres. They are roughly described as a small sphere, the anterior part, smoothly fused to a larger sphere, the posterior part. The anterior part contains the cornea, iris, and pupil; and the posterior part contains the outer white shell (the sclera), vitreous humour, and retina. There are two axes that describe the eye. The *optical axis* is the line that intersects the centre of both spherical parts of the eye; this describes its geometry. The *visual axis* is the line that

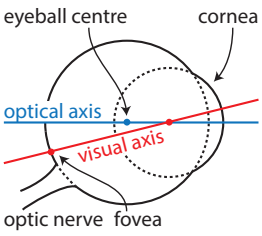


Figure 2.4: The difference between the visual and optical axes in a reduced eyeball model.

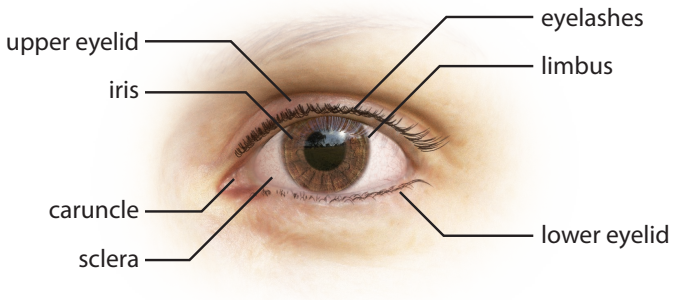


Figure 2.5: The external anatomy of the human eye.

passes through the nodal point of the lens and the fovea; this describes the path of sharpest vision, and is what we wish to recover with eye tracking. The difference between them can be seen in Figure 2.4. Since the fovea's position in the retina is slightly off-centre, the visual axis is offset from the optical axis by about $5°$. This angular offset varies per individual so must be determined using a user calibration procedure.

2.1.2 External anatomy

We are all familiar with the external appearance of the eye. Figure 2.5 shows a typical example. The externally visible parts of the eyeball are the sclera, cornea, iris, and pupil. The boundary between the iris and the sclera is called the *limbus*. From oblique angles, the refraction of the cornea is quite apparent, and the iris and pupil appear distorted.

Much of the eyeball is covered by the eyelids. These are thin folds of skin that protect the eyeball by squinting: closing the eye partially to shield the retina from excessive amounts of light, and blinking: coating the eyeball with tear fluid and removing irritants. The eyelids are lined with hundreds of eyelashes. These are short, thin hairs around 10mm long that block dust and other irritants from entering the eye, and also provide a visual cue for eyelid motion and gaze direction. Eyelids are also used to communicate emotions and social signals, for example squinting with suspicion, or fluttering eyelashes with affection.

The upper and lower eyelids meet at the eye corners, also known as the *medial* (inner) and *lateral* (outer) *canthi*. The medial canthus and sclera is joined by the *lacrimal caruncle* – a pink, triangular fleshy nodule. So, unlike the outer eye corner where the sclera and both eyelids meet in one place, the "inner eye corner" is less precisely defined. Is it the where the sclera meets the caruncle, or the middle of the caruncle, or where the eyelid skin meets the caruncle? To further complicate matters, some eyes feature an epicanthic skin fold that can completely obscure the medial canthus. Unfortunately, there is no consensus on

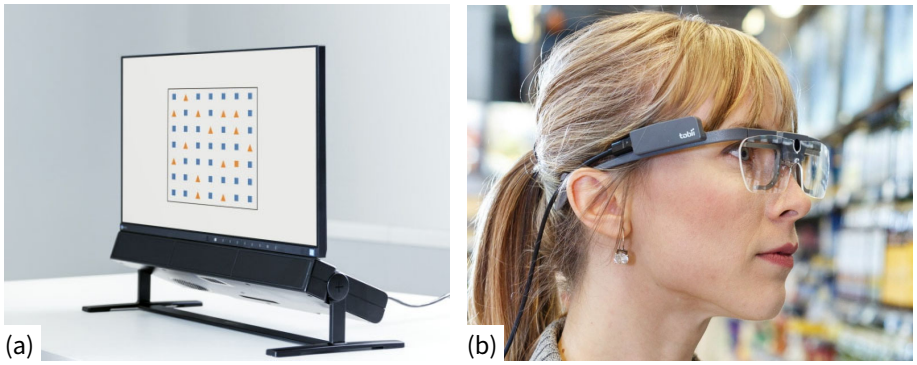


Figure 2.6: Two commercial gaze estimation platforms for researchers: (a) the Tobii Pro Specturm remote screen-based eye tracker, and (b) the Tobii Pro Glasses wearable eye tracker.

what exactly is the "inner eye corner" in the research community.

Moving further away from the eyeball we find the eyebrows, nose ridge, and upper cheek. Depending on how hooded or inset the eye is, the eyebrows can be very close to the eyeball, or far away. For deep inset eyes, shadows cast by the brow ridge and nose can obscure certain eye features.

2.2 Mainstream eye tracking

Eye trackers have improved a great deal over the historical systems I described earlier, becoming less invasive, more accurate, and more convenient. Nowadays, most commercial eye trackers like those in Figure 2.6 perform video-oculography (VOG): tracking gaze using computer vision. Given input images from one or more cameras, VOG systems perform two main tasks: 1) They first detect and localize features of the eyes in the images. 2) They then interpret these features to determine gaze direction. In this section I describe how currently available eye trackers work, and explain their limitations.

2.2.1 PC-CR eye tracking

The most widely used VOG techniques track the *pupil center* and *corneal reflections*, so are known as **PC-CR** techniques. The idea is to illuminate the eye in such a way to cause small, bright reflections on the surface of the cornea. Computer vision techniques are then used to accurately localize these corneal reflections and the pupil centre in an image. Having found these feature points, there are two ways to estimate gaze: regression-based methods and geometry-based methods.

Corneal surface reflections are the first Purkinje images, and are also known as *glints*.

2D regression-based PC-CR learns a mapping from image-space offset vectors between pupil centre and corneal reflections onto a 2D screen coordinate. This mapping is learned via a subject-specific calibration procedure, and many different regression tools have been proposed in the literature. However, regression-based methods only implicitly model the geometry of the scene, so they often fail under head movement. They remain popular as they are simple to implement, and do not require camera or geometric calibration.

3D geometry-based PC-CR uses a anatomy-based eyeball model to estimate a gaze vector in 3D space. Using a global geometric model of light sources, cameras, and corneal curvature, the corneal reflections can be analysed to determine the cornea centre in 3D. The optical axis is then given as the vector between cornea centre and pupil centre, and the visual axis can be estimated using anatomical averages, or determined through calibration. These methods are more robust to head movement but require precisely calibrated hardware setups.

3D gaze vectors can be intersected with a 3D screen plane to determine a point of gaze.

To make locating feature points easier, the majority of commercial systems use infrared (IR) light to illuminate the subject, and IR cameras to capture noise-free, high quality images. Such methods are called *active IR* approaches. Active IR systems generally use IR light sources with wavelength around 780–880nm (Hansen and Ji, 2010). Light at these wavelengths is invisible to the human eye, so does not distract the subject, or cause pupillary contractions.

To localize the pupil centre in an image, active IR systems can exploit the *bright pupil* effect (Hansen and Ji, 2010). This phenomenon occurs when an illuminator is positioned close to the optical axis of a camera – the light from the illuminator bounces back off the eye's retina and into the camera, causing the pupil to appear bright. If the eye is illuminated with a light source positioned away from the camera's optical axis, this phenomenon does not occur and the pupil appears dark, as normal. Bright pupil tracking systems use a co-axial light source synchronized with a camera to alternate between capturing bright and dark pupil images. When we look at the difference between the dark and bright pupil images, the pupil's shape is clearly apparent.

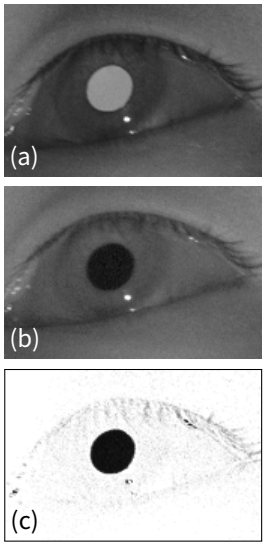


Figure 2.7: (a) a bright pupil image; (b) a dark pupil image; (c) the difference between them.

However, the bright pupil effect's intensity varies between subjects, ethnicities, and age groups (Nguyen et al., 2002), so commercial systems use a combination of bright and dark pupil tracking. Dark pupil tracking exploits the well-defined shape of the pupil – it is roughly circular, so it appears as an ellipse in images. This pupil ellipse can be found using voting-based methods like the elliptical Hough transform (Young et al., 1995), or model-fitting methods like Starburst (Li

and Parkhurst, 2005) or image-aware RANSAC (Swirski et al., 2012). Though dark pupil tracking does not require synchronized cameras and illuminators, it is more computationally expensive than bright pupil tracking, and less robust.

2.2.2 Remote eye tracking

The most convenient eye trackers operate remotely, at a distance. These devices generally take the form of long black rectangles that house IR cameras and illuminators, and track gaze over computer screens. They must be mounted directly below the screen to work effectively and have a limited operational distance of between 50–90cm. Futhermore, they must be placed directly in front of the subject, and only work on screens up to 24 inches across[1]. This is because PC-CR features can disappear at extreme eyeball rotations. Despite these restrictions, remote eye trackers are popular with researchers as they are quick to set up, and do not interfere with the subject being tracked.

In the past, remote eye trackers were very expensive and only available to the research community. Nowadays, recent models have finally become more affordable as developers look into integrating eye tracking into mainstream products such as video games. For example, the Tobii EyeX[2] can be bought for ~£100.

2.2.3 Wearable eye tracking

Capturing a good image of an eye is key for estimating gaze successfully. It is no wonder then that some gaze estimation systems choose to position a camera as close as possible to the eye itself. These systems consist of IR cameras and light sources mounted on lightweight eyeglass frames worn by a user. In contrast to remote systems these wearable systems allow the user to walk around freely, hence they are also known as mobile eye trackers. Being able to move freely is critical for observing eye movements during tasks in a naturalistic environment or outside in the real-world.

Wearable eye tracking is also used with virtual reality, a field that has grown rapidly in recent years. Virtual reality headsets lend themselves well to eye tracking, as the addition of small cameras and illuminators is a minor issue considering the large amount of hardware already worn on the face. Some vendors offer hardware kits for modifying mainstream virtual reality devices, implanting eye tracking systems

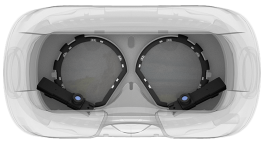


Figure 2.8: The black components are an add-on kit for eye tracking with VR headsets (Kassner et al., 2014).

[1] http://www.tobiipro.com/product-listing/tobii-pro-spectrum/
[2] https://tobiigaming.com/product/tobii-eyex/

inside them (see Figure 2.8), while others offer custom-built devices with fully integrated eye tracking systems, e.g. FOVE[3].

2.3 Alternative approaches

Though PC-CR remains the most popular VOG technique, there are other "glint-free" VOG methods that do not use corneal reflections (Swirski and Dodgson, 2013; Kassner et al., 2014). A key advantage of these methods is that they can track gaze in environments with uncontrolled illumination, e.g. outdoors in the daytime, where other unwanted reflections may obscure small glints (see Figure 2.9). Furthermore, glint-free systems have less complex hardware requirements than PC-CR systems – they do not require a precise geometric calibration of their IR illuminators. Glint-free techniques are therefore popular with researchers investigating low-cost, low-complexity eye tracking. For example, Swirski and Dodgson (2013) use pupil ellipse geometry alone, tracking the pupil over time and fitting its moving boundary onto a consistent sphere-based pupil motion model.

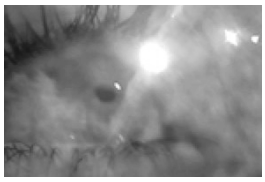


Figure 2.9: An in the wild eye image exhibiting strong reflections that would obscure corneal glints (Tonsen et al., 2016).

Figure 2.10: An example direct infrared oculography system: the Saccadometer by Ober Consulting.

Aside from VOG, there are other approaches for gaze estimation that do not use video cameras (see Figure 2.10). Direct infrared oculography determines eyeball rotation by illuminating the sides of the eyeball with IR light at a close range, and measuring the amount of scattered light that returns using a photodetector, e.g. a photoresistor. Since the iris is darker than the sclera, the amount of light measured will be lower when the iris is pointing towards the photodetector, and higher when the iris is pointing away. This change in returning light level can be mapped onto eyeball movement. Because these systems use photodetectors rather than cameras, they can sample eye movement at a higher rate (~1kHz) compared to VOG (~60Hz), and can be considered more ergonomic since they do not place cameras or illuminators in the subject's field of view. However, these photodetector-based systems are highly sensitive to environmental illumination which can vary over a measurement session, so are therefore better suited to high-frequency saccade detection rather than accurate gaze estimation.

Electro-oculography (EOG) is another video-free approach. EOG uses the fact that the eyeball is an electric dipole, with a positive pole at the cornea and negative pole at the retina (Brown et al., 2006). As the eyeball rotates, the dipole orientation shifts, causing a change in the surrounding electric potential field. These changes can be measured

[3] https://www.getfove.com/

using skin-contact electrodes placed around the eyes. EOG systems have been used to record eye movements in everyday life as they are computationally lightweight (no video processing), and relatively unobtrusive (no protruding cameras) (Bulling et al., 2009). Additionally, since EOG uses sensors placed directly on the skin, the face remains mostly unobscured; this makes EOG useful in facial performance capture (Krupinski and Mazurek, 2010). EOG's main limitation arises from the fact that skin-contact electrodes are not very reliable. Their response signal varies with temperature, skin conductance and illumination – factors that can change over a recording session. As a result, the accuracy of EOG degrades over time.

2.4 Visible light remote gaze estimation

Though the systems described in the previous two sections provide solutions for many scenarios, eye tracking has not yet become an every-day technology for the average consumer. This is in contrast to other face-related computer vision tasks like facial landmark tracking which is now used worldwide in popular smartphone apps. The obvious problem is that eye tracking still requires special hardware not available on commodity devices.

Gaze estimation researchers are therefore currently attempting to remove the need for special equipment – the IR cameras and illuminators. If gaze estimation could work accurately and reliably using a **single commodity camera**, like those found in webcams or smartphones, eye tracking could be brought to the masses. Since these systems only use light in the visible spectrum, and work at a distance, they are classed as *visible light remote eye trackers*. The systems I developed and discuss in the rest of this dissertation fall into this category. This section discusses the challenges of working with visible light, reviews the state of the art in this field, and explores their limitations.

2.4.1 Challenges

Visible light gaze estimation is more challenging than traditional active IR gaze estimation. A key step in many eye tracking algorithms is finding the pupil. Visible light gaze estimators use passive light only, so cannot exploit the bright pupil effect. Unfortunately, dark pupil tracking is out of the question too. Under controlled IR illumination, the boundaries between sclera, iris, and pupil can be made quite clear. However, as shown in Figure 2.11, this is not the case with visible light images. For dark coloured irises especially, the pupil may not

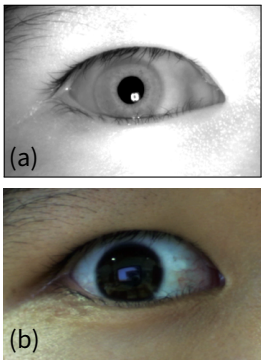


Figure 2.11: The same eye under active IR (a) and visible light (b). Note the clear pupil/iris boundary in (a).

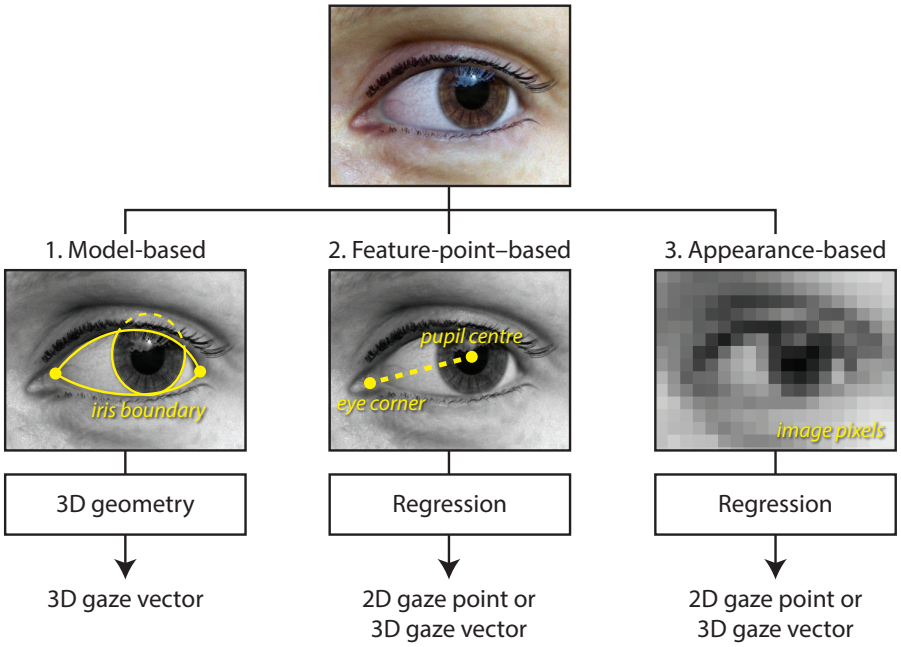


Figure 2.12: There are three main ways to do gaze estimation. Model-based methods rely on 3D geometry, and feature-point– and appearance-based methods rely on machine learnt regression.

be discernible at all. New algorithms that do not depend on accurate pupil tracking are needed.

Traditional eye trackers can adjust their level of illumination to obtain sharp images with good contrast around the eye region. However, the cameras used in commodity devices must rely on passive light to illuminate their subjects. In everyday life people often use their devices in low light or back-lit conditions, and so these cameras can have trouble picking a suitable exposure level. As a result, we can expect under- or over-exposed images where the details of the eye region are no longer clearly visible. Furthermore, camera sensitivity may have to be turned up in low light conditions, so sensor noise becomes more apparent. A good gaze estimator should therefore work for noisy, poorly exposed eye images.

2.4.2 Categorization

There is a large body of previous work on visible light remote gaze estimation, and many attempts have been made to categorize the different approaches (Ferhat and Vilarino, 2015; Hansen and Ji, 2010). This is challenging considering the boundaries between different methods are not always clear – many systems use a combination of techniques. I roughly follow Ferhat and Vilarino's (2015) taxonomy and categorize previous work into three different types (Figure 2.12):

1. **Model-based**. Similar to their active IR counterparts, visible light model-based systems use geometric models to recover the position and orientation of eyeballs in an image.

2. **Feature-point–based**. Techniques of this type localize feature points in an image, like the glints in PC-CR, and map from these points onto a gaze point or direction.

3. **Appearance-based**. These methods are the most different from mainstream eye tracking techniques, and try to map directly from eye image pixels onto eye gaze.

Model-based methods are generative, and require prior knowledge of eyeball anatomy. Feature-point–based and appearance-based methods are discriminative; they do not care how the eye image was formed, and simply try to assign the correct gaze label.

For the sake of brevity, when I mention "gaze estimation" from now on, I mean visible light remote gaze estimation using a single camera.

2.4.3 Model-based gaze estimation

There are two common types of model used to estimate gaze direction: 1) elliptical iris boundary models, and 2) spherical eyeball models. Iris boundary models use the fact that the iris is circular in real life, so its boundary (the limbus) appears as an ellipse in an image. The idea is to extract the 2D limbal ellipse from an image as accurately as possible, then back-project it onto its generating 3D circle, and take the normal vector of the plane the 3D circle lies in as the gaze direction. The limbal ellipse can be found using Canny (1986) edge detection (Wang et al., 2003), active contours (Hansen and Pece, 2005), a randomized Hough transform (Huang et al., 2010), or RANSAC model fitting (Zhang et al., 2010; Wood and Bulling, 2014). The back-projection from 2D ellipse to 3D circle requires knowledge of the iris radius, for which anatomical averages are often used (Wang et al., 2003).

However, the limbal ellipse can be hard to extract accurately, especially if the subject's eyelids are half-closed. Spherical eyeball techniques avoid this challenge by estimating both the pupil position and eyeball centre in 3D, and taking the line between them as the gaze direction. Since the pupil itself may not be discernible, these methods generally use the iris centre as a proxy. The iris centre can be estimated using shape-based techniques (Valenti and Gevers, 2008; Timm and Barth, 2011) or machine learning methods (Baltrušaitis et al., 2016). The 3D eyeball centre can then be inferred using facial landmarks (Ishikawa, 2004; Baltrusaitis et al., 2013), head tracking (Mora and Odobez, 2014), or eyelid contours (Wu et al., 2007). Since iris centre localisation and facial landmark tracking are less prone to extreme outliers than

limbus extraction, spherical eyeball model methods can be more robust. However, they are also less precise considering the uncertainty in 3D eyeball centre estimation.

2.4.4 Feature-point–based gaze estimation

Feature-point–based techniques are similar to PC-CR regression methods in that they localize key points in the eye image and map from these onto a gaze direction or gaze point. This mapping is learnt during a calibration procedure. By focussing on extracting a handful of facial key points, these methods can made robust to typical computer vision challenges like changes in illumination or subject pose. The offset vector between the pupil centre and inner eye corner (PC-EC) is often used. For example, Hansen et al. (2002) use an active appearance model to find the PC-EC vector which they map onto gaze using Gaussian processes, and Valenti et al. (2009) tracked PC-EC points using isophote curvature and determined gaze using linear regression. However, researchers have struggled with the fact that the inner eye corner moves as the user looks in different directions (Sesma et al., 2012). This means the eye corners are not stable "anchor" points for reliable regression. Furthermore, PC-EC regression suffers the same problem as PC-CR regression: simple regression models that were trained for frontal head poses become significantly less accurate under head movement.

Consequently, researchers have explored using a larger set of feature points to become more reliable under different gaze directions and head poses. For example, Bäck (2006) combined the PC-EC features with nostril positions and head pose to create a richer feature vector for regression, and Skodras et al. (2015) tracked additional points on the eyelids for improved vertical gaze tracking. There is unfortunately no consensus on what set of facial landmark points should be tracked for reliable feature-point–based gaze estimation. In theory, an ideal set of landmarks might exist, but how should researchers choose them? To address this, new techniques were developed that use the whole eye as a feature for regression: appearance-based gaze estimation.

2.4.5 Appearance-based gaze estimation

Appearance-based techniques represent the state of the art in gaze estimation. Both model- and feature-point–based techniques track specific parts of the eye region. For example, many model-based approaches must be able to clearly see the limbus, but this is not possible if someone's eyes are nearly closed. Feature-point–based

approaches must track the eye corners accurately, but the inner eye corner is a poorly defined landmark.

Appearance-based approaches side step these problems by analysing the entire eye image holistically. If it's possible to regress from eye image pixels onto a gaze direction, certain ambiguities concerning eye image features can be ignored. They work by transforming the eye image into some feature space, and mapping from this feature vector onto gaze using classification or regression. There are two main ways of mapping an eye image into a feature space: 1) using traditional hand-crafted computer vision features, or 2) using deep learning.

Hand-crafted features The simplest way to represent an image is to use its raw pixel values. Lu et al. (2014) downsample eye image to 3×5px and use adaptive linear regression to map this 15-long vector onto gaze. However, they require sub-pixel eye image alignment to achieve good results. Many traditional computer vision problems have been made tractable through the use of more sophisticated feature spaces. These have also been explored for gaze estimation. For example Schneider et al. (2014) investigated a range of different feature types (DCT, LBP, and HOG), and showed that dimensionality reduction using manifold-alignment could improve accuracy. Huang et al. (2015) recently performed a similar investigation but in a more challenging scenario – natural tablet use. They found that multilevel HOG features used with a random forest regressor yielded the best results.

Deep learning As shown by the cross-feature experiments performed by Schneider et al. (2014) and Huang et al. (2015), it can be tricky to know which type of feature will perform best. However, we now know that better results can often be obtained by *learning* the feature representation itself, rather than hand-crafting it. This is done by training a deep neural network end-to-end (LeCun et al., 2015). Zhang et al. (2015) was the first to demonstrate a successful appearance-based gaze estimation system that used deep-learning. They trained a deep convolutional neural network (CNN) using 200,000 images of 15 people collected during everyday laptop use, and showed improvements over the previous state-of-the-art in appearance-based gaze estimation. Recently, Krafka et al. (2016) trained a multi-input CNN using 2.5M iPhone images of 1,450 people collected with crowdsourcing, also demonstrating impressive results. Furthermore, these deep learning systems have been made real time through careful engineering and exploiting dark-knowledge (Hinton et al., 2015) in the neural network.

Many appearance-based methods assumed a fixed head pose to make

the problem simpler (Tan et al., 2002; Sugano et al., 2013). If the gaze estimator is combined with some sort of face tracker, head pose information can be used to allow for free head motion. One choice is to train an additional sub-system to compensate for changes in head pose (Lu et al., 2011). An alternative option is to cluster eye images based on head pose into separate bins, and pass them onto different regressors (Sugano et al., 2014). If the 3D surface of the face is tracked in detail, face images can be normalized into a canonical frontal view before regression (Egger et al., 2014; Jeni and Cohn, 2016).

This face normalization process is often called *frontalization*.

For machine learning methods to work well, they must be trained with good ground truth data. As discussed in Chapter 1, manually collecting such data for gaze estimation can be time-consuming, unreliable, and expensive. Furthermore, these datasets are generally collected with a specific scenario in mind, e.g. laptop use (Zhang et al., 2015) or smartphone use (Krafka et al., 2016). The resulting appearance-based systems are tied to those scenarios.

2.5 Taxonomy of tracking types

In Table 2.1 I outline some key properties of commercial eye tracking systems and gaze estimation methods found in research today, the latter including my work in this dissertation. I have chosen four broad categories that represent the majority of eye tracking technology to try and make the main distinctions clear: 1) Commercial remote eye trackers, 2) commercial wearable eye trackers, 3) appearance-based approaches for gaze estimation, and 4) model-based approaches for gaze estimation. In the rest of the section, I discuss a number of important differences between the approaches.

Hardware and price

Figure 2.13: Commodity devices with eye tracking do exist, like the Alienware 17R4 laptop. However, they are targeted at gaming enthusiasts, not the mainstream.

Considering how cheap the actual hardware involved in a traditional active IR eye tracker is (several LEDs and a camera), we should ask ourselves: "If we want gaze estimation to be available to the general public, why not put a typical eye tracker into everyday devices?". Indeed, this has been done (Figure 2.13) for devices aimed at enthusiasts. While it would be possible to integrate additional IR cameras and illuminators into a standard laptop, tablet, or phone; device manufacturers would prefer to avoid this and use the front-facing cameras that come as standard in any smart phone. As well as increasing the total bill-of-parts for each device, more cameras and lights means higher power consumption – something we want to avoid with portable devices to save battery life. Furthermore, an array of illuminators would complicate the industrial

	Commercial		Research	
	Remote	Wearable	Appearance-based	Model-based
Illumination	Controlled; Active IR	Controlled; Active / Passive IR	Uncontrolled; Visible light	Uncontrolled; Visible light
Hardware	A desk-mounted box housing IR cameras and lights	A pair of glasses with IR cameras and illuminators	A typical webcam	A typical webcam
Price	~£100–10,000	~£1,000	~£20	~£20
Accuracy	Very high ($<1°$)	Very high ($<1°$)	High (~3°) if trained on a specific user. Else low (~10°).	Low (~10°)
Sampling rate	Very fast (>500Hz)	Fast (>60Hz)	Depends on machine learning approach (~30Hz)	Depends on model and fitting approach (~15Hz)
Data	Eye gaze; Pupil diameter	Eye gaze; Pupil diameter	Eye gaze	Eye gaze
Calibration	Required	Required	Optional	Optional
Environment	Indoor use only	Glint-free algorithms can work outdoors	Can cope with different illumination with diverse training data	Illumination can be factored into scene model
Head pose	User must face device	Not an issue since attached to head	Depends on training data. Generally requires frontal pose	Free head movement handled through 3D model
Eyewear	Handled through per-session calibration	Handled through per-session calibration	Handled if trained on a specific user	Refraction through glasses not handled
My work	–	–	Chapters 3 and 5	Chapters 6 and 7

Table 2.1: A comparison of gaze estimation approaches used in commercial eye trackers and research. It can be seen that each type of approach has its own benefits and drawbacks. Though the methods in research still lag behind industry with regards to accuracy and sampling rate, it is hoped that further research can address these gaps in performance while still maintaining the key benefits of simplified hardware requirements and robustness to viewing conditions.

design, and might be distracting to the user.

Illumination

Commercial eye trackers make excellent use of IR illuminators to make computer vision easier. By flooding the user with IR, they ensure there is enough light to capture a sharp image with little motion blur. Through the use of an array of IR illuminators, they ensure the appearance of small and easily-located glints that serve as an efficient encoding of the eyeball's position and orientation in 3D space. However, by relying on these features, they limit themselves to environments

without IR pollution, or must take measures to alleviate IR pollution as much as possible. If gaze estimation could work with visible light alone, changes in environmental illumination would be much less of an issue. However, for gaze estimation, this means reinventing the wheel – throwing away the nice easy-to-find features of IR glints, and trying to estimate gaze in some other way. As a result the algorithms used by eye tracking in industry and gaze estimation in research end up very different.

Sampling rate, and type of data
High sample rate data (>500Hz) collected by remote eye trackers allows for special types of analysis not available at lower sampling rates, e.g. the analysis of microsaccades or smooth pursuit. Such a high sampling rate requires active illumination to flood the user with enough light, and significant power requirements for processing. Therefore, this level of gaze analysis is likely to remain in the laboratory for some time yet. But gaze information at lower sample rates could still be useful for mainstream devices, e.g. if you just want to know if a user has noticed a pop-up dialog on their phone. Furthermore, the high sampling rates afforded by commercial devices represent the result of decades of careful engineering. Many of the methods found in research today could be accelerated with the right implementation, or specific hardware. For example, dedicated neural network co-processors promise to drastically improve the speed and efficiency of neural network inference. Although a state-of-the-art neural network approach might require a high-performance GPU today, it might easily fit on dedicated compute hardware in the near future.

2.6 Datasets

Eye gaze datasets are collections of images (or videos) of people looking in different directions. Each image is labelled with either a gaze direction, or a gaze point. These datasets play a key role in the development of gaze estimation systems as they are used for both machine learning and evaluation. There are a number of datasets available in the research community, and their quality has increased over the years. While older datasets include a handful of subjects looking in a few discrete directions, newer datasets include a wider variation of subject appearance, head pose, environmental lighting, and gaze direction. This section describes the gaze estimation datasets that I used throughout this dissertation.

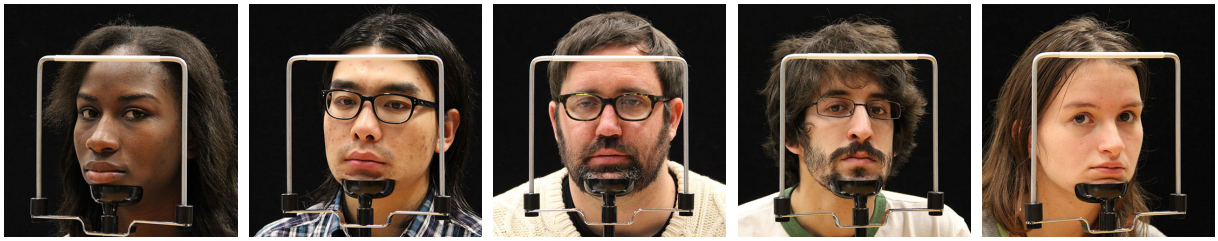


Figure 2.14: The Columbia gaze dataset by Smith et al. (2013) contains high quality images of 56 different subjects taken under five discrete head poses ranging from $-30°$ to $30°$.

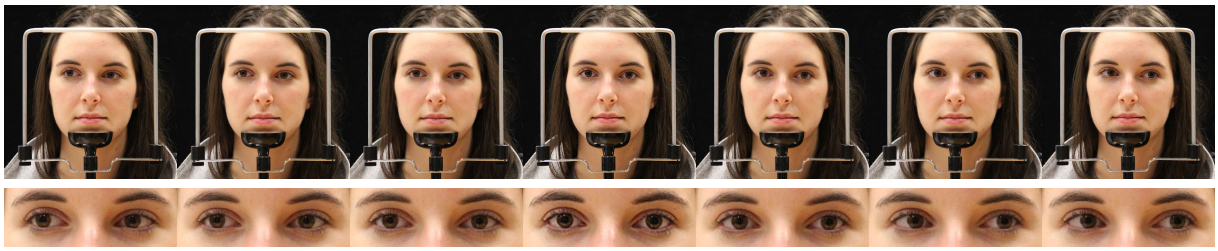


Figure 2.15: For each head pose, the subjects were instructed to look at a set of 21 dots marked on a wall, resulting in three vertical and seven horizontal gaze conditions. Horizontal gaze is shown here.

2.6.1 Columbia gaze dataset

Smith et al. (2013) created the Columbia gaze dataset to train an appearance-based eye-contact detector. They demonstrated its use in eye-contact–aware human-computer interaction, though the dataset has since been used by the gaze estimation community as well.

The dataset contains 5,880 high resolution images (5184×3456px) of 56 subjects (32 male, 24 female). The subjects were aged between 18 and 36, and included a range of different ethnicities. For each subject, 105 images were taken representing a combination of five different head poses ($0°, \pm 15°, \pm 30°$, see Figure 2.14), seven different horizontal gaze directions ($0°, \pm 5°, \pm 10°, \pm 15°$, see Figure 2.15), and three different vertical gaze directions ($0°, \pm 10°$). The subjects were illuminated evenly across the face with soft overhead lighting, so no hard cast shadows were present.

The Columbia dataset was captured with a DLSR camera so its images are of higher quality than other webcam-based datasets. The photos were taken in a controlled laboratory environment – subjects were seated in front of a black screen, and their head position was stabilised using an adjustable chin rest. A grid of dots was attached to the wall in front of the subjects to act as gaze targets. To simulate different head poses, the photographer moved around the subject, capturing images

Figure 2.16: Images from the Eyediap dataset (Funes Mora et al., 2014) taken with the HD webcam. The LEDs in the bottom left of the images were used to synchronize the multiple video feeds.

from five fixed positions with the camera positioned at eye level. Each photo was checked manually to ensure the subject was looking in the right direction.

I chose to use the Columbia dataset as it represents the best case scenario for visible light remote gaze estimation: high resolution images with sharp focus and simple illumination. Though this dataset is not as challenging as others in terms of head pose or illumination, it is valuable since it includes a large number of subjects and covers a wide range of gaze directions and head poses.

2.6.2 Eyediap

Funes Mora et al. (2014) collected the Eyediap video dataset to develop and evaluate RGB-D gaze estimation algorithms. Though it was primarily used to train Mora and Odobez's (2014) Kinect-based Geometric Generative Gaze Estimation (G^3E) system, it has also been adopted as a benchmark by the gaze estimation community.

The dataset includes videos of 16 subjects (12 male, 4 female) over 94 sessions, taken with two different cameras simultaneously: 1) a Microsoft Kinect, producing a VGA resolution (640×480px) RGB video and a separate depth-only video. 2) a typical webcam, producing a full-HD resolution (1920×1080px) RGB video (see Figure 2.16). The visible light systems described in this dissertation do not use the depth video from the Kinect, and use the two RGB videos only.

Videos were recorded over six sessions for each participant, representing a combination of three target types and two head pose conditions. The three different types of gaze target were:

1. **Discrete screen targets**: Subjects looked at stationary targets displayed at random locations on a computer screen directly in front of them.

2. **Continuous screen targets**: To obtain examples of smoother

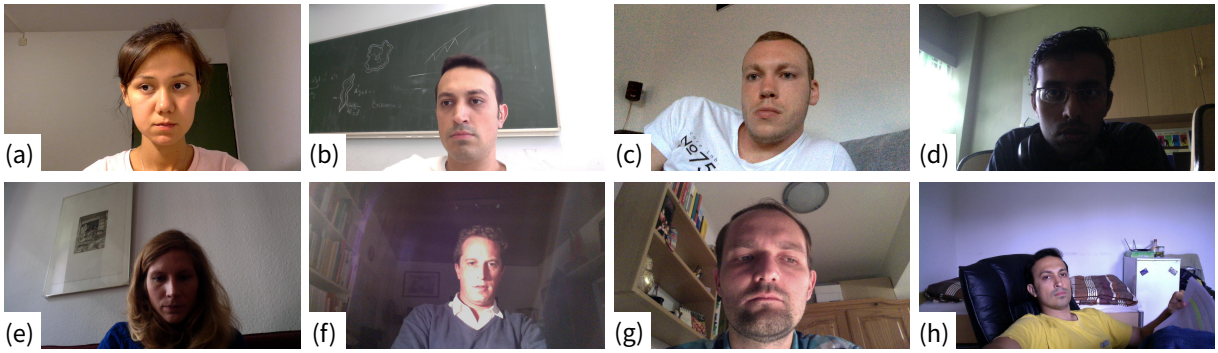


Figure 2.18: Example images from the MPIIGaze dataset (Zhang et al., 2015). As well as images where the face and eyes are clear (a,b,c), this dataset includes back-lit images (d), underexposed images (d,e), strong uneven illumination (f,g), and faces that are far away (h).

gaze movement, the virtual screen targets moved along random trajectories for 2 seconds.

3. **3D floating targets**: To obtain wider gaze angles not possible with a screen, subjects looked at a small ball that was dangled in front of them, hanging on a thin thread from a stick.

To help evaluate robustness against head pose variation, two types of head pose condition were used:

1. **Static head pose**: Subjects kept their head still while looking at gaze targets, though they did not use a chinrest.

2. **Mobile head pose**: To introduce head pose variation, participants were told to perform random head movements while looking at gaze targets. Examples can be seen in Figure 2.17.

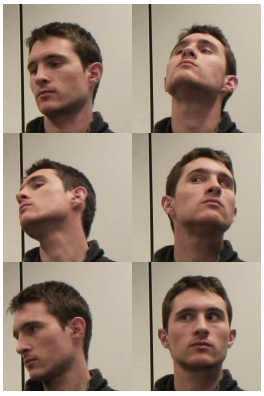


Figure 2.17: Example images from the Eyediap mobile head pose condition.

I decided to use the Eyediap dataset as it contains images and videos taken with typical webcams in a typical indoor environment. Though the Eyediap dataset contains fewer people than the Columbia dataset, it includes more challenging lighting conditions, a smoother distribution of gaze directions, and a wider range head poses.

2.6.3 MPIIGaze

Zhang et al. (2015) noted that previous gaze datasets had been collected under controlled conditions in laboratories, so did not represent the wide range of environments encountered in the real world. They therefore collected the MPIIGaze dataset to study appearance-based gaze estimation in the wild.

MPIIGaze consists of 213,659 images collected from 15 participants (11 male, 6 female) during natural everyday laptop use over three

months. See Figure 2.18 for some examples. The number of images collected by each participant ranged between 34,745 and 1,498. The participants installed custom-written data collection software on their laptops. Every ten minutes the software would request they look at a sequence of 20 on-screen gaze markers. The participants were asked to fixate on each gaze point, and confirm this action by pressing the space bar. A 1280×720px image was recorded for each gaze marker using the laptop's front facing camera. There were no constraints on how, where, or when they should use their laptops. Consequently, MPIIGaze contains significantly more head pose and illumination variation than typical gaze datasets.

I chose to use the MPIIGaze dataset as it represents the most challenging types of image we might expect with visible light remote gaze estimation. The subjects are often quite far from the camera, and the images can be poorly exposed and out of focus. However, the gaze distribution is rather narrow, ranging between $(-20°, +20°)$ horizontally and $(0°, -20°)$ vertically. This means this data does not include examples of someone looking far towards the left or right, and has no images of people looking up. As a result, this dataset might not be very useful for training gaze estimation systems that target scenarios other than laptop use, e.g. mobile phone use, or estimating gaze in archival footage.

2.7 Summary

In this chapter I described gaze estimation. To provide some background, I described the history of the field and how the eye itself works. I then descibed both mainstream and alternative approaches for eye tracking that are used today. Following that, I discussed the specific field of eye tracking research that my work fits into: visible light remote eye gaze estimation. I finally described three benchmark eye gaze datasets that are used in this dissertation for evaluation.

3

Learning-by-synthesis for gaze-estimation

*This chapter presents work that has been published at the **International Computer Vision Conference 2015** in Santiago, Chile (Wood et al., 2015). Tadas Baltrušaitis implemented the eye-shape registration system, and Xucong Zhang implemented the deep neural network for gaze estimation.*

Machine learning methods that learn from large amounts of labelled training data currently perform best for many problems in computer vision, including object detection (Redmon et al., 2015), scene recognition (Zhou et al., 2016), and human pose estimation (Wei et al., 2016). To achieve good performance, these methods require good training data. Normally, such training data would be collected off the Internet and labelled with ground truth by Mechanical Turks. However, this post-hoc labelling is not possible for eye gaze. Given a picture of someone, how can we accurately say what direction they were looking in? While it is possible to roughly classify someone's eye gaze as up, down, left, or right, many applications require more precise measurements. As a result, gaze estimation researchers must collect the images for eye gaze datasets themselves – a time-consuming and expensive process. Participants must be called into a laboratory or hired over a period of time, limiting environment and participant variability. Furthermore, the range of gaze directions is then limited by practical matters, e.g. the size and placement of the screen used for gaze markers.

Mechanical Turks are human workers that can be hired to perform tasks that computers cannot do.

To address these difficulties, computer vision researchers have used *learning-by-synthesis* to generate large amounts training data with computer graphics. The key idea is to learn from synthetic training data,

rather than real world images. The advantages of this approach are that data collection and annotation require less human effort, and image synthesis can be geared to specific application scenarios. The eye is a particularly difficult object to model accurately in 3D given the dynamic shape changes it undergoes with facial motion, and the complex structure of the eyeball itself. For this reason, previous work on learning-by-synthesis for gaze estimation employed only fundamental computer graphics techniques, rendering low-resolution meshes without considering changes in illumination or the varying material properties of the face (Sugano et al., 2014) (see Figure 3.1). In addition, Sugano et al.'s (2014) models are not controllable and the resulting synthesized dataset contains only gaze labels, limiting its usefulness for other computer vision problems, such as facial landmark registration – a crucial step in many gaze estimation pipelines.

Facial landmark tracking is used to align images of eyes as input to appearance-based gaze estimation systems.

In this chapter I present a novel method for rendering a large number of realistic eye region images using a collection of dynamic and controllable eye region models. I provide a comprehensive and detailed description of the model preparation process and rendering pipeline (see Figure 3.2 for an overview of model preparation). Furthermore, while previous work required researchers to create 3D models themselves with photogrammetry, I used readily available 3D models instead. I then present and evaluate two separate systems trained on the resulting data (*SynthesEyes*): an eye region specific facial landmark tracker and an appearance-based gaze estimator. The controllability of the dynamic eye region models allowed me to quickly generate high-quality training data for these two separate tasks.

The specific contributions of this chapter are threefold. I first describe my new technique for generating large amounts of synthetic training data using dynamic eye region models, including a range of realistic appearance variation using image-based-lighting. I then demonstrate the usefulness of SynthesEyes by out-performing state-of-the-art methods for eye-shape registration as well as cross-dataset appearance-based gaze estimation in the wild – a challenging scenario. Finally, to stimulate further research in this area, I have made the dynamic eye region models available upon request.

3.1 Related work

The work in this chapter is related to previous work on learning using synthetic data and computational modelling of the eyes.

Figure 3.1: An example 3D model from the UT Gaze dataset (Sugano et al., 2014). Note the lack of detail in the shape, and the illumination that has been baked into the texture (e.g. the highlight on the nose ridge).

3.1.1 Learning using synthetic data

The performance of learning-based approaches critically depends on how well the test data is covered by the training set. Since it is generally impossible to record enough training data samples to cover all possible test data, synthetic training data has been used instead. Previous work has demonstrated that synthetic training data can be beneficial for a wide range of computer vision tasks including human body pose estimation (Okada and Soatto, 2008; Shotton et al., 2013), object detection and recognition (Fu and Kara, 2011; Yu et al., 2010; Liebelt and Schmid, 2010), and facial landmark localization (Baltrušaitis et al., 2012; Jeni et al., 2015). In many cases, synthetic depth images were used rather than colour images to side step the additional degrees of variation corresponding to changes in object albedo and environmental illumination. Learning-by-synthesis was perhaps most famously used for the Kinect (Shotton et al., 2013), where 3D body models were posed and rendered to train the pose estimation system.

My work in this chapter was inspired by recent related work by Sugano et al. (2014) on learning-by-synthesis for gaze estimation. They prepared a collection of static eye region models of 50 participants looking at 160 on-screen gaze targets using *multi-view stereo* (Furukawa and Ponce, 2010). The participants were photographed by eight cameras simultaneously at SXGA resolution (1280×1024px), and the photos were processed to produce a static 3D mesh of the face. An example can be seen in Figure 3.1. These 3D eye region models were then rendered from new viewpoints to create synthetic training data with additional head pose variation. However, this work suffers several limitations. First, their 3D models are not controllable – we cannot move someone's eyes to change where they are looking. Second, since the source images were low resolution, the resulting meshes fail to accurately capture the details of the eyes. Third, the resulting model textures all have environmental illumination "baked in" so they cannot be re-lit accurately. Instead, the models I use in this chapter are fully

In computer graphics, if lighting is "baked in", it means it is hard-coded into an objects texture.

controllable and have detailed geometry with realistic skin and eyeball materials. This allowed me to generate training data with increased gaze and illumination variation.

Kaneva et al. (2011) used controllable photorealistic models of a city to evaluate different image features. For their comparisons to be valid, they noted that the synthetic data must be as realistic as possible. If the object of interest is highly complex, like the human eye, it is not clear whether we can rely on overly-simplistic 3D models like those by Sugano et al. (2014). In this chapter we investigate if realism is important when generating synthetic data for learning-by-synthesis.

3.1.2 Computational modelling of the eyes

Human eyeballs are extremely complex organs. Fortunately, given that realistic eyes are important for many fields, there is already a large body of previous work on modelling and rendering eyes. Ruhland et al. (2014) provides a recent survey.

As we spend so much time looking at eyes, mistakes in their appearance can cause a computer generated face to appear uncomfortably unfamiliar. Modelling eyes accurately is therefore important for the entertainment industry, who want to portray realistic characters accurately. Recent work by Bérard et al. (2014) represents the state-of-the-art in capturing high quality eye models for actor digital-doubles. They used a hybrid reconstruction method to separately capture both the transparent corneal surface and diffuse sclera in high detail, and recorded deformations of the eyeball's interior structures. These results are impressive, but probably beyond the scope of webcam-based visible light remote gaze estimation. Considering the wide field-of-view, lens blur, and noise patterns typically found in webcams, the details of such super-accurate eye models would not be visible, so would be wasted. Visually-appealing eyes are also important for the video-game industry. Jimenez et al. (2012) recently developed techniques for modelling eye wetness, refraction, and ambient occlusion in a standard rasterization pipeline, showing that approximations are sufficient in many cases.

Aside from visual effects, previous work has used 3D models to examine the eye from a medical perspective. Sagar et al. (1994) built a virtual environment of the eye and surrounding face for mechanically simulating surgery with finite element analysis. Priamikov and Triesch (2014) built a 3D biomechanical model of the eye and its interior muscles to understand the underlying problems of visual perception and motor control. Eye models have also been used to evaluate geometric gaze estimation algorithms, allowing individual parts of an

eye tracking system to be evaluated separately. For example, Świrski and Dodgson (2014) used a rigged head model and reduced eyeball model to render ground truth images for evaluating pupil detection and tracking algorithms.

3.2 Dynamic eye region model

I developed a *dynamic eye region model* which can be randomly posed and rendered to generate fully labeled training images. My goals were realism and controllability, so I combined high quality 3D head scan geometry with my own posable eyeball model. Figure 3.2 shows an overview of the model preparation process.

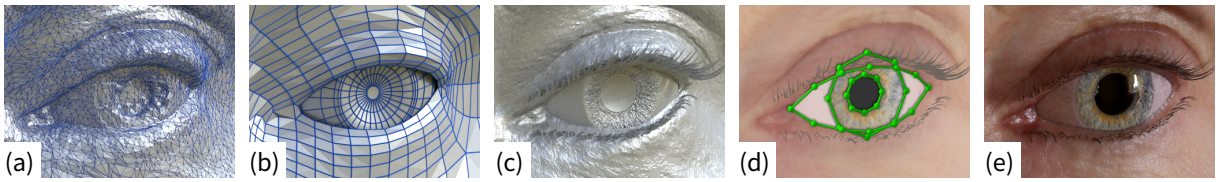


Figure 3.2: An overview of the model preparation process: Dense 3D head scans (1.4 million polygons) (a) are first converted into a topology more suitable for animation (retopology; 9,005 polygons) (b). High resolution skin surface details are restored by displacement maps (c), and 3D iris and eyelid landmarks are annotated manually (d). A sample rendering is shown (e).

For eye gaze training data to be useful, it should represent the types of variety encountered in the real world. This includes **inter-person variety**, i.e. different people have different looking eyes, and **intra-person variety**, i.e. someone's eye changes appearance when they look around or change facial expression. To capture inter-person variety, I use a collection of different head scans featuring different types of eyes. To capture intra-person variety, I model the continuous changes in shape and texture that the face and eyes undergo during eye movement using computer animation. This is more challenging than simply rendering a collection of static models like Sugano et al. (2014), as dynamic geometry must be correctly topologized and rigged to be able to deform correctly.

In this section I describe the process of preparing a dynamic eye region model for synthetic eye image generation. First, I describe the anatomically inspired eyeball model that I place into each head scan. Then I discuss how to acquire 3D head scans and convert each one from a piece of static geometry into a dynamic eye region model that can assume a range of realistic poses.

Figure 3.3: The two-part eyeball model that I used in each dynamic eye region model. Left: the full render. Middle, blue: the inner sclera part that uses a purely diffuse material. Right, red: the outer part that is a little bigger than the inner part, contains the corneal bulge, and is transparent and reflective.

3.2.1 Simplified eyeball model

The eye model I created for this work consists of two parts (see Figure 3.3). The outer part (the red wireframe) approximates the eye's overall shape with two spheres ($r_1 = 12\text{mm}, r_2 = 8\text{mm}$ (Ruhland et al., 2014)), the former representing the main part of the eye, and the latter representing the corneal bulge. To avoid a sharp seam between these two spheres, the two sphere meshes were joined, and the vertices along the seam were smoothed to minimize differences in face-angle. This outer part is transparent, refractive (index of refraction $n = 1.376$), and partially reflective. The sclera's bumpy surface is modeled with smoothed solid noise functions, and applied using a *displacement map* – a 2D scalar function that shifts a surface in the direction of its normal (Lee et al., 2000). The inner part (the blue wireframe) is a flattened sphere – the planar end represents the iris and pupil, and the rest represents the sclera, the white of the eye. There is a 0.5mm gap between the two parts which accounts for the thickness of the cornea. I manually built this eye model using Blender.

The pupil is not actually modelled as a hole in the model, but is shown with a texture map that is black for the pupil region. In this simplified eye model, the pupil and iris are perfect circles that line up with eachother along the eyeball's optical axis. Both the pupil and iris boundaries line up with vertices and edges that form circles in the eyeball model's topology, allowing their shape to be easily controlled with vertex animation.

Eyeballs vary in both shape (pupillary dilation) and texture (iris colour and scleral veins). To model shape variation I use *blend shapes* – an animation technique for interpolating between several different poses created for the same topological mesh (Orvalho et al., 2012). I created blend shapes for dilated and constricted pupils, as well as large and

small irises to account for a small amount (10%) of variation in iris size. Blend shapes are can be mixed, so its possible to model an eye with a small pupil, and a large iris.

Different eyeball textures are generated by alpha compositing eyeball texture images using three separate layers: 1. a *sclera* tint layer (white, pink, or yellow); 2. an *iris* layer with four different photo-textures (amber, blue, brown, grey); and 3. a *veins* layer (blood-shot or clear). I fixed the sclera tint for each head model to match their skin tone, but varied iris texture by randomly choosing from the collection when synthesizing training data. There are methods for procedural iris-synthesis that would have allowed the iris texture to vary in a more detailed, continuous fashion (Lefohn et al., 2003). However, I decided the added complexity would not be worthwhile considering the low resolutions considered in our scenario.

3.2.2 3D head scan acquisition

For an eye region rendering to be realistic, it must also feature realistic nearby facial detail. While previous approaches used models created by artists (Świrski and Dodgson, 2014), I used high-quality head scans captured by a professional photogrammetry studio (10K diffuse colour textures, 0.1mm resolution geometry)[1]. Facial appearance around the eye varies significantly between people as a result of different eye-shapes (e.g. round vs hooded), orbital bone structure (e.g. deep-set vs protruding), and skin detail (wrinkled vs smooth). Therefore the head models I chose (see Figure 3.4) cover a range of different ethnicities and age. As can be seen in Figure 3.2, the cornea of the original head scan has been incorrectly reconstructed by the optical scanning process. This is because the eyeball is made of transparent and reflective material. Since most 3D reconstruction algorithms assume that all surfaces reflect light in a purely-Lambertian way, they do not work well on shiny objects like an eyeball. For images to represent a wide range of gaze directions, the eyeball needed to be posed separately from the face geometry. I therefore removed the original scanned eyeball from the mesh, and placed my own eyeball model in its place.

Nowadays it is possible to easily purchase such scans online from ∼£15 per scan.

Though human skin is not purely Lambertian, in practice it is close enough for these 3D reconstruction methods to work well.

3.2.3 Eye region geometry preparation

While the original head scan geometry is suitable for being rendered as a static model, its high resolution topology is not ideal for changes in shape. Vertical saccades are always accompanied by eyelid motion, so

[1] Ten24 3D Scan Store – http://www.3dscanstore.com/

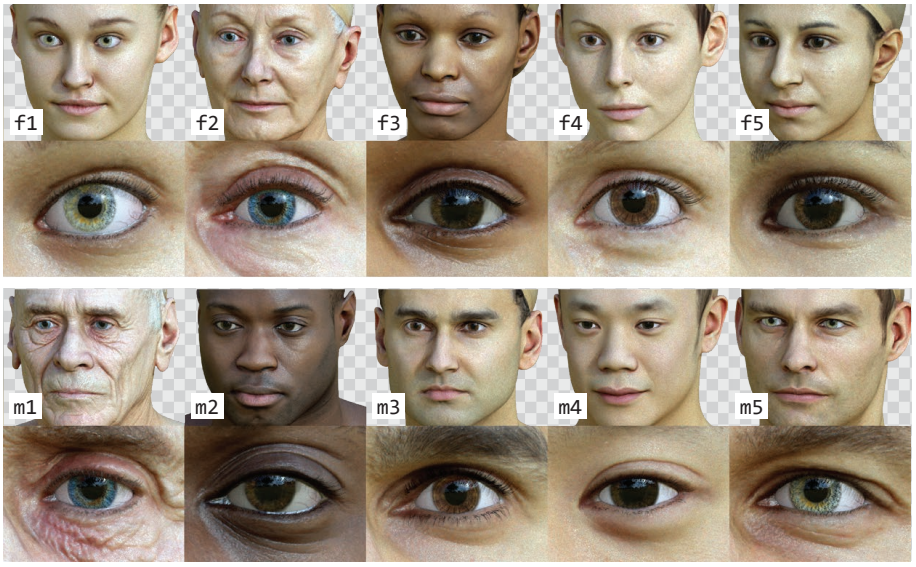


Figure 3.4: The collection of head models (five female `f1–5`, five male `m1–5`) and corresponding images of the eye region. The collection of head models exhibits a good range of variation in eye shape, bone structure, skin smoothness, and skin color.

the eyelids should be posed so they correspond to the gaze direction. This requires computer animation. To do this, I needed a more efficient (low-resolution) geometric representation of the eye region, where edge loops flow around the natural contours of facial muscles. This leads to more realistic animation as mesh deformation matches that of actual skin tissue and muscles (Orvalho et al., 2012).

I therefore *retopologized* the face geometry of each scan to reduce the resolution (number of polygons) of the mesh using a semi-automatic system[2]. This left me with 10 newly topologized models, one for each original scan. I transferred the original high resolution color textures from the high resolution scan onto the new lower resolution retopologized models. As can be seen in Figure 3.2 (b), the resulting edge loops follow the exterior eye muscles, allowing for realistic eye region deformations. This retopologized low-poly mesh ($\sim$10K polys) has lost the skin detail of the original scan, like wrinkles, see Figure 3.2 (c). These were restored with a displacement map computed from the original scanned geometry (Lee et al., 2000).

Although they are two separate organs, there is normally no visible gap between eyeball and skin. However, as a consequence of removing the eyeball from the original scan, the retopologized meshes do not necessarily meet the eyeball geometry. This can be seen in Figure 3.2 (b). To compensate for this, the mesh's eyelid vertices are automatically

[2]ZBrush ZRemesher 2.0, Pixologic, 2015

Figure 3.5: Eyelids are posed to look up or down by interpolating between blend shapes based on gaze direction (model `m2` as example). Note how the eyelid crease expands when the eye looks down.

shifted along their normals at render time so they touch the eyeball. This is called *shrinkwrapping*. This prevented unwanted gaps between the models, even after changes in pose. The face geometry was then assigned physically-based materials, including subsurface scattering to approximate the penetrative light transfer properties of skin, and a glossy component to simulate its oily surface. Blender's inbuilt material shaders were used for this.

3.2.4 Modelling eyelid motion and eyelashes

I model eyelid motion using blend shapes for upwards-looking and downwards-looking eyelids, and interpolating between them based on the global pitch of the eyeball model. This makes our face-model dynamic, allowing it to continuously deform to match eyeball poses. Rather than rendering a single or perhaps several discrete head scans representing a particular gaze vector (Sugano et al., 2014), I instead created training data with a dense distribution of facial deformation. Defining blend shapes through vertex manipulation can be a difficult and time-consuming task but fortunately, only two are required and they have small regions of support. As the tissue around the eye is compressed or stretched, skin details like wrinkles and folds are either attenuated or exaggerated (see Figure 3.5). I modeled this by interpolating between smoothed colour and displacement textures for downwards-looking eyelids, removing any wrinkles. These blend shape and texture modifications were carried out using photos of the same heads looking up and down as references.

To model the eyelashes, I followed the approach of Świrski and Dodgson (Świrski and Dodgson, 2014), and simulated each eyelash hair using directed particle effects. Blender's built-in particle effect system was used for this. Particles were emitted from two control surfaces manually placed underneath the surface of the upper and lower eyelids. I used 180 hairs for the upper eyelash and 100 for the lower eyelash. To make them curl, eyelash particles experienced a slight amount of

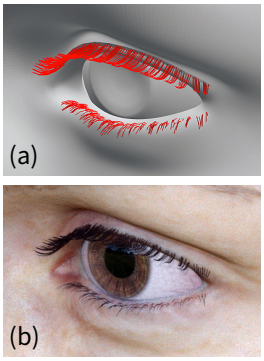


Figure 3.6: Eyelash directed particle effects highlighted in red (a); the rendered result (b).

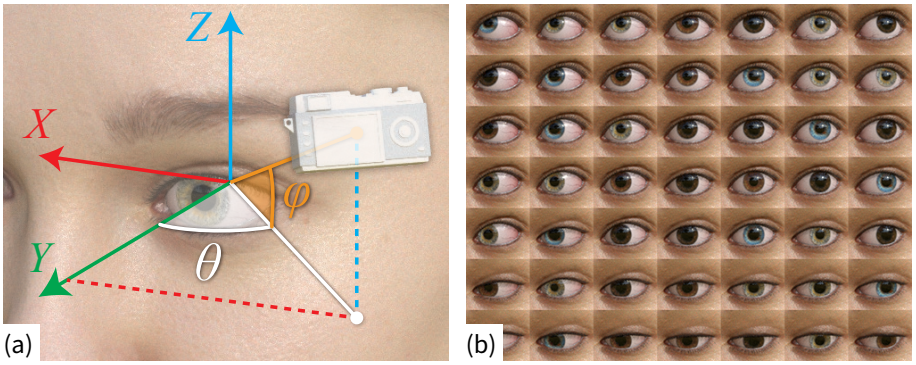


Figure 3.7: (a) The camera is positioned by iterating over spherical coordinates θ, ϕ. This varies head pose. (b) At each camera position, I rendered a large number of eye images corresponding to different gaze directions.

gravity during growth (negative gravity for the upper eyelash).

3.3 Synthesizing training data

Eyes can look very different when viewed from different camera positions and under different illumination. My goal was to render the models from lots of different viewpoints and under lots of different illumination conditions to create a dataset representative of real world variety. In this section I first describe how I posed the virtual camera and models, and explain my simple technique for varying scene illumination using image-based lighting. I then describe my landmark annotation process and finally discuss the details of my rendering setup.

3.3.1 Posing the model

For a chosen eye-region model, each rendered image is determined by parameters $(\mathbf{c}, \mathbf{g}, L, E)$: 3D camera position $\mathbf{c}$; 3D gaze vector $\mathbf{g}$; lighting environment L; and eye model configuration E. Camera positions $\mathbf{c}$ were chosen by iterating over spherical coordinates (θ, ϕ), centered around the eyeball center (see Figure 3.7). Images were rendered with orthographic projection to simulate a small region-of-interest around the eye being cropped from a wide-angle camera image. At each camera position $\mathbf{c}$, I rendered multiple images with different 3D gaze vectors to simulate the eye looking in different directions. Examples with fixed L are shown in Figure 3.7 (b). Gaze vectors $\mathbf{g}$ were chosen by first pointing the eye directly at the camera (simulating eye-contact), and then modifying the eyeball's pitch (α) and yaw (β) angles over a chosen range.

Within E, I randomly configured iris color and pose eyelids according

Figure 3.8: The four HDR environment maps I used alongside renders of the face and eye region. Left: outdoors environments representing bright direct sunlight and soft cloudy light. Right: indoors environments representing a typical bedroom and a strong directional light.

to g. For a generic dataset, I rendered images with up to $45°$ horizontal and vertical deviation from eye-contact, in increments of $10°$. As I posed the model in this way, there was the possibility of rendering "unhelpful" images that either simulate impossible scenarios or are not useful for training. To avoid violating anatomical constraints, I only rendered images for valid eyeball rotations $|\alpha| \leq 25°$ and $|\beta| \leq 35°$. Before rendering, I also verified that the 2D pupil center in the image was within the boundary of the eyelid landmarks. This prevents the rendering of images where too little of the iris is visible.

3.3.2 Creating realistic illumination

One of the main challenges in computer vision is illumination invariance – a good system should work under a range of real-life lighting conditions. I realistically illuminate the models using *image-based lighting*, a technique where high dynamic range (HDR) panoramic images are used to provide light in a scene Debevec (2002). This works by photographically capturing omni-directional light information from the real world, storing it in a equirectangular image, and then projecting it onto a sphere around the object. When a ray hits that texture during rendering, it takes that texture's pixel value as light intensity. At render time I randomly chose one of four freely available HDR environment images[3] to simulate a range of different lighting conditions (see Figure 3.8). The environment is then randomly rotated to simulate a continuous range of head-pose, and randomly scaled in intensity to simulate changes in ambient light. As shown in Figure 3.9, a combination of hard shadows and soft light can generate a range of appearances from only a single HDR environment. This simple and flexible approach creates variability using measured light levels in

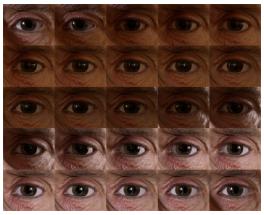


Figure 3.9: A range of lighting conditions can be generated by rotating a single environment map.

[3] `http://adaptivesamples.com/category/hdr-panos/`

real-world environments.

3.3.3 Eye-region landmark annotation

For eye shape registration, I needed additional ground-truth annotations of eye-region landmarks in the training images. As shown in Figure 3.2 (d), each 3D eye-region was annotated once in 3D with 28 landmarks, corresponding to the eyelids (12), iris boundary (8), and pupil boundary (8). The iris and pupil landmarks were defined as a subset of the eyeball geometry vertices, so deform automatically with changes in pupil and iris size. The eyelid landmarks were manually labelled with a separate mesh that follows the seam where eyeball geometry meets skin geometry. This mesh is assigned shape keys and deforms automatically during eyelid motion. Whenever an image is rendered, the 2D image-space coordinates of these 3D landmarks are calculated using the camera projection matrix and saved.

3.3.4 Rendering Images

I use Blender's[4] inbuilt Cycles path-tracing engine for rendering. This renderer traces the paths of many light rays per pixel, scattering light stochastically off physically-based materials in the scene until they reach illuminants. A GPU implementation is available for processing large numbers of rays simultaneously (150/px) to achieve noise-free and photorealistic images. I rendered a generic SynthesEyes dataset of 11,382 images covering $40°$ of viewpoint (i.e. head pose) variation and $90°$ of gaze variation. I sampled eye colour and environmental lighting randomly for each image. Each 120×80px rendering took 5.26s on average using a commodity GPU (Nvidia GTX660). As a result it is possible to specify and render a cleanly-labelled dataset in under a day on a single machine – a fraction of the time taken by traditional data collection procedures (Zhang et al., 2015).

3.4 Experiments – Eye shape registration

As the SynthesEyes pipeline can reliably generate consistent landmark location data, it was used for training a *Constrained Local Neural Field* (CLNF) (Baltrusaitis et al., 2013) model. This system tracks landmarks (interesting points) on a subject's face, e.g. points along the eyelid boundaries. This is an important pre-processing stage for many eye tracking algorithms. Experiments were conducted to evaluate the

[4]The Blender Project – `http://www.blender.org/`

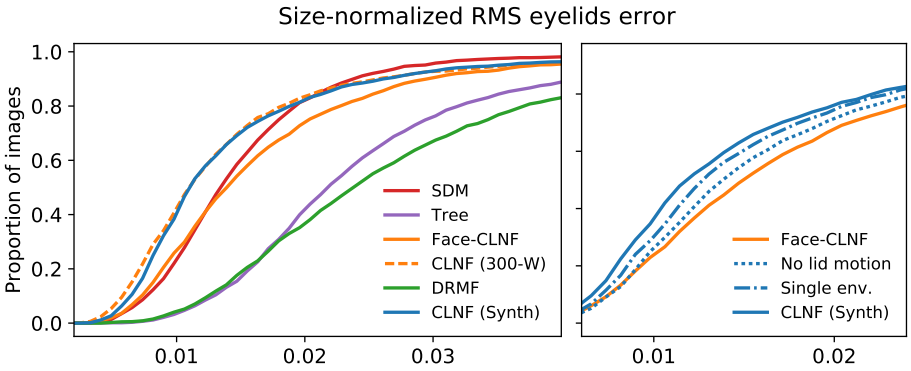


Figure 3.10: An eye region landmark detector trained with synthetic data outperforms the state-of-the-art for eyelid-registration in the wild. The right plot shows how performance degrades for training data without important degrees of variation: realistic lighting and eyelid movement.

generalizability of the SynthesEyes dataset on two different use cases: eyelid registration in the wild and iris tracking from webcams.

3.4.1 Eyelid registration in the wild

We performed an experiment to see how the system generalizes on unseen and unconstrained images. We used the validation datasets from the 300 Faces In the Wild (300-W) challenge (Sagonas et al., 2016) which contain labels for eyelid boundaries. All of the approaches were tested on 830 (out of 1026) test images. Images that did not contain visible eyes (occluded by hair or sunglasses) or where face detection failed were discarded.

CLNF patch experts were trained using the generic SynthesEyes dataset, and a Point Distribution Model (PDM) was constructed by performing Principal Component Analysis on the 3D landmark locations. As the rendered images did not contain closed eyes, extra closed eye landmark labels were generated by moving the upper eyelid down to lower one or meeting both eyelids halfway. The eye shape registration system was initialized using the face-CLNF (Baltrusaitis et al., 2013) facial landmark detector. To measure the difference between using synthetic or real training images, an eyelid-only CLNF model was trained on 300-W images, but used the same PDM used for synthetic data (CLNF 300-W). The proposed approach was compared against the following state-of-the-art facial landmark detectors: CLNF (Baltrusaitis et al., 2013), Supervised Descent Method (SDM) (Xiong and De la Torre, 2013), Discriminative Response Map Fitting (DRMF) (Asthana et al., 2013), and tree based face and landmark detector (Zhu and Ramanan, 2012).

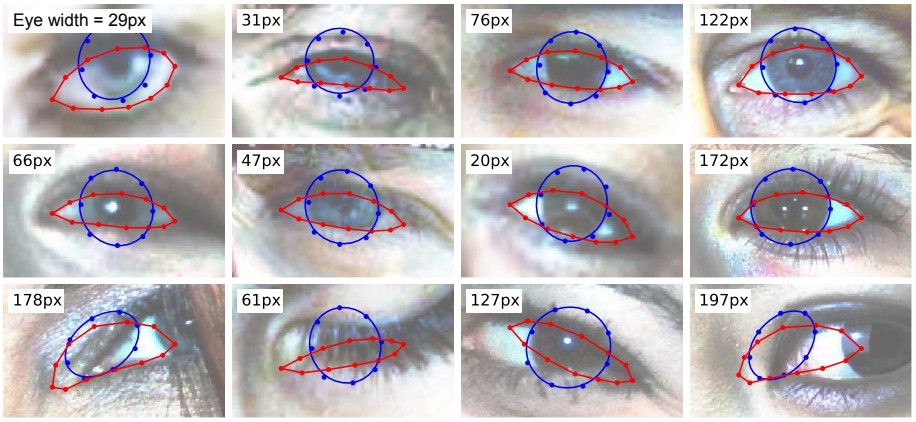


Figure 3.11: Example fits of the SynthesEyes eye-CLNF on in the wild images. The top two rows illustrate successful eye-shape registrations, while the bottom row illustrates failure cases including unmodelled occlusions (e.g. hair) and strong makeup.

The results of this experiments can be seen in Figure 3.10, and example model fits are shown in Figure 3.11. Errors were recorded as the RMS point-to-boundary distance from tracked eyelid landmarks to ground truth eyelid boundary, and were normalized by inter-ocular distance. First, the proposed system CLNF Synth (Mdn $= 0.0110$px) trained using only ten synthetic head models in four lighting conditions results in very similar performance to a system trained on unconstrained in the wild images, CLNF 300-W (Mdn $= 0.0110$px). Second, the results show the eye-specific CLNF outperformed all other systems in eye-lid localization: SDM (Mdn $= 0.0134$px), face-CLNF (Mdn $= 0.0139$px), DRMF (Mdn $= 0.0238$px), and Tree based (Mdn $= 0.0217$px). The first result suggests the importance of high-quality consistent labels. Even though our synthetic dataset has fewer different identities and illumination conditions than 300-W, the synthetic error-free and noise-free labels make the dataset just as valuable for eye region landmark localization.

Learning-by-synthesis also allows us to investigate what parts of the synthesis process are important for generating good training data. Two further eye-specific CLNFs were trained on two different versions of SynthesEyes: one without eyelid motion and one with only one fixed lighting condition. As can be seen in Figure 3.10, not using shape variation (Mdn $= 0.0129$px) and using basic lighting (Mdn $= 0.0120$px) lead to worse performance due to missing degrees of variability in training sets.

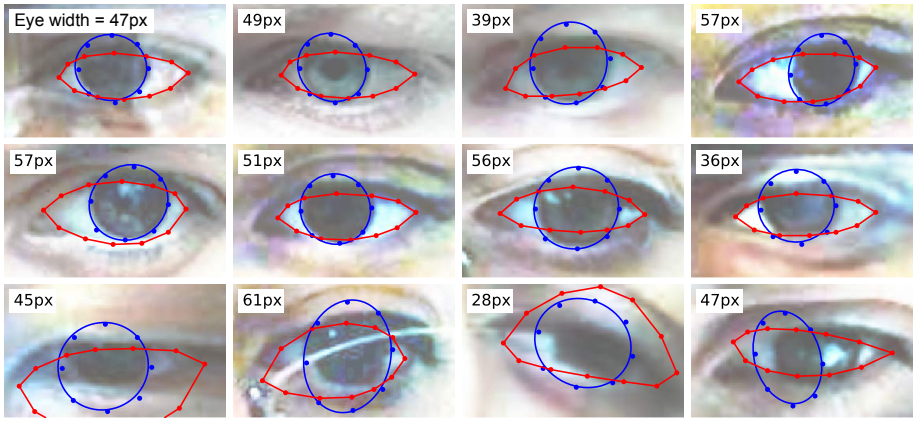


Figure 3.12: Example fits of the SynthesEyes eye-CLNF on webcam images. The top rows show successes, and the bottom failures. Causes of failure included glasses and eye shapes not covered in our training set.

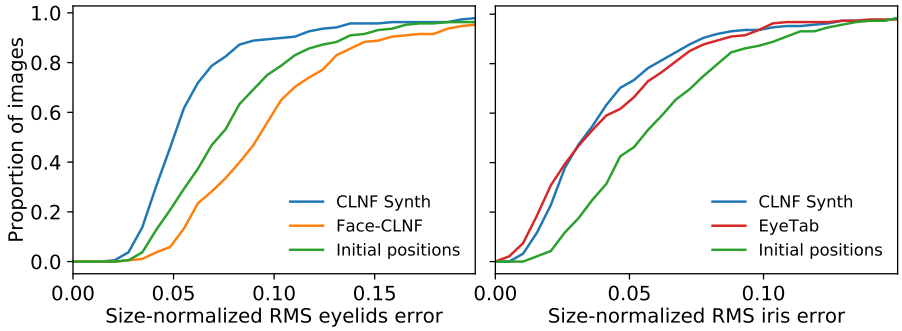


Figure 3.13: The eye region landmark detector was also evaluated on the MPIIGaze dataset (Zhang et al., 2015). The left plot shows how the eye-specific CLNF landmark detector trained with synthetic eye images outperfoms a CLNF model trained for the whole face with real images. The right plot shows the CLNF model performing comparably with the state-of-the-art (Wood and Bulling, 2014) for iris-registration (localising the iris boundary).

3.4.2 Eye-shape registration for webcams

While the 300-W images represent challenging conditions for eyelid registration they do not feature iris labels and are not representative of conditions encountered during everyday human-computer interaction. I therefore annotated sub-pixel eyelid and iris boundaries for a subset of MPIIGaze (Zhang et al., 2015) (188 images). Pupil accuracy was not evaluated as it was impossible to discern in most images.

The eye-specific CLNF (CLNF Synth) was compared against EyeTab (Wood and Bulling, 2014), a recent shape-based approach for webcam gaze estimation that robustly fits ellipses to the iris boundary using image-aware RANSAC (Swirski et al., 2012). The other systems from the previous experiment were not considered as they do not detect

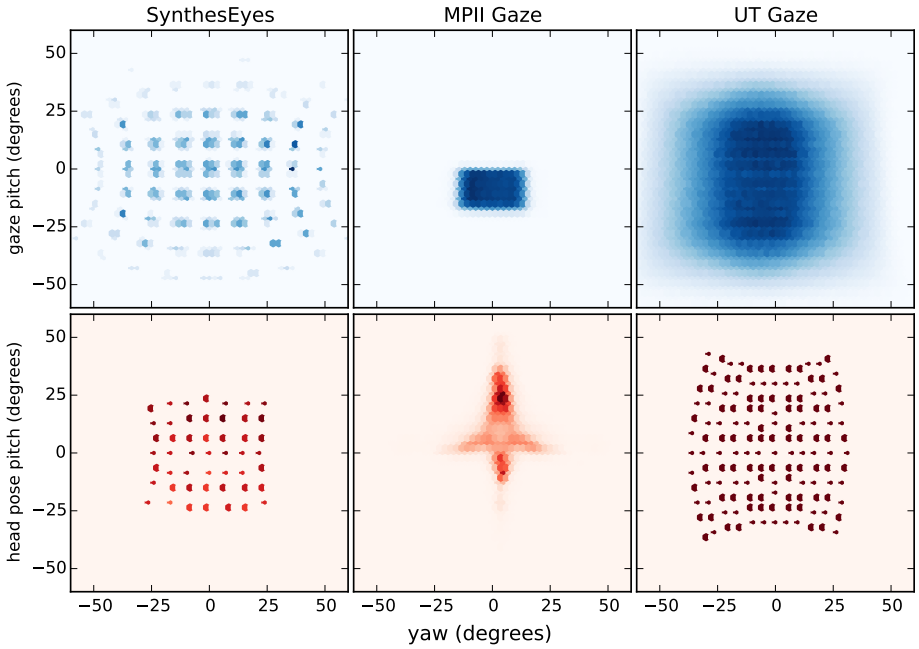


Figure 3.14: The gaze direction (first row) and head pose (second row) distributions of different datasets: SynthesEyes, MPIIGaze Zhang et al. (2015), and UT Multiview Sugano et al. (2014). Note how the gaze and head distributions of MPIIGaze are quite tight compared to the wider distributions of SynthesEyes and UTGaze.

irises. I used a modified version of the author's EyeTab implementation with improved eyelid localization using face-CLNF (Baltrusaitis et al., 2013). The mean position of all 28 eye-landmarks following model initialization was used as a baseline. Eyelid errors were calculated as RMS distances from predicted landmarks to the eyelid boundary. Iris errors were calculated by least-squares fitting an ellipse to the tracked iris landmarks, and measuring distances only to visible parts of the iris. Errors were normalized by the eye-width, and are reported using average eye-width (44.4px) as reference.

As shown in Figure 3.13, the proposed approach (Mdn = 1.48px) demonstrates comparable iris-fitting accuracy with EyeTab (Mdn = 1.44px). However, CLNF Synth is more robust, with EyeTab failing to terminate in 2% of test cases. As also shown by the 300-W experiment, the eye-specific CLNF Synth localizes eyelids better than the face-CLNF. See Figure 3.12 for example model fits.

3.5 Experiments – Gaze estimation

To evaluate learning-by-synthesis with SynthesEyes for appearance-based gaze estimation, a *cross-dataset* experiment was performed (as

described by Zhang et al. (2015)). The idea is to train and test a gaze estimation system on different datasets. This is a very important challenge, as it represents the ability of a gaze estimation system to generalize beyond the idiosyncrasies of the particular camera and environment used to create a gaze dataset.

I synthesized training images using the same camera settings as in the UT dataset (Sugano et al., 2014). The head pose and gaze distributions for the three datasets are shown in Figure 3.14. The same convolutional neural network (CNN) model as described by (Zhang et al., 2015) was trained on both synthetic datasets and tested on MPIIGaze. As shown in Figure 3.15, the CNN model trained on the generic SynthesEyes dataset achieved similar performance ($\mu = 13.91°$) as the model trained on the UT dataset ($\mu = 13.55°$). This confirms that our approach can synthesize data that leads to comparable results with previous synthesis procedures (Sugano et al., 2014). Note from Figure 3.15 that there is still a performance gap between this cross-dataset and the within-dataset training (red line).

While it is in general important to cover a wide range of head poses to handle arbitrary camera settings, if the target setting is known in advance, e.g. laptop gaze interaction as in case of MPIIGaze, it is possible to target data synthesis to the expected head pose and gaze ranges. To investigate the benefits of targetted learning-by-synthesis, I rendered an additional dataset (SynthesEyes targeted) for a typical laptop setting ($10°$ pose and $20°$ gaze variation). For a comparison, the entire UT dataset was re-sampled to create a subset (UT subset) that has the same gaze and head pose distribution as MPIIGaze. To make a comparison assuming the same number of participants, the UT dataset was divided into five groups with ten participants each, and the errors were averaged across these five groups. As shown in the third and fourth bars of Figure 3.15, having similar head pose and gaze ranges as the target domain improves performance compared to the generic datasets. A Wilcoxon signed-rank test showed that the CNN trained with the targetted SynthesEyes dataset achieves a statistically significant performance improvement over the CNN trained with the UT subset ($W = 4.01\text{e}8, p < 1\text{e}{-5}$).

These results suggest that neither SynthesEyes nor the UT dataset alone capture all variations present in the test set, but different ones individually. For example, while SynthesEyes covers more variations in lighting and facial appearance, the UT dataset contains real eye movements captured from more participants. Recent work by Fu and Kara (2011) and Peng et al. (2014) demonstrated the importance of fine-tuning models

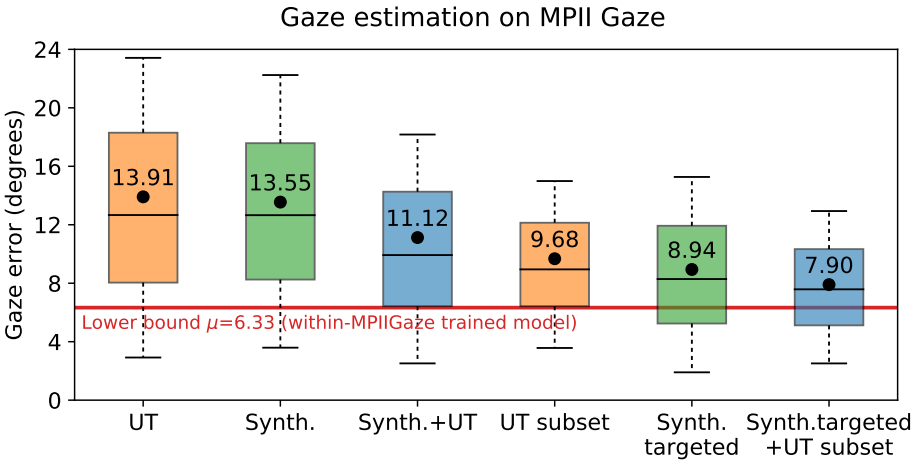


Figure 3.15: Test performance on MPIIGaze. X axis represents training set used. Dots are mean errors, and red line represents a practical lower-bound (within-dataset cross-validation score). Note how combining synthetic datasets for training lead to improved performance (blue plots).

initially trained on synthetic data on real data to increase performance. Therefore a final experiment was conducted to evaluate the performance of training and fine-tuning using both datasets (see Figure 3.15). The same CNN model was trained on the SynthesEyes dataset and then fine-tuned using the UT dataset. This fine-tuned model achieved better performance in both settings (untargeted $\mu = 11.12°$, targeted $\mu = 7.90°$). A Wilcoxon signed-rank test showed that the CNN trained with targetted data and finetuned with UT data achieves a statistically significant performance improvement over the previous state-of-the-art in cross-dataset appearance based gaze estimation, UT subset (Zhang et al., 2015) $(W = 1.93\text{e}8, p < 1\text{e}{-5})$. This indicates a promising avenue for future investigation into pre-training with synthetic data, and fine-tuning with real data.

Person-specific appearance Appearance-based gaze estimation performs best when trained and tested on the same person, as the training data includes the same eye appearances that occur during testing. However, eye images from SynthesEyes and MPIIGaze can appear different due to differences in eye-shape and skin color. To examine the effects of this, an additional experiment was conducted where ten separate systems were trained (one for each SynthesEyes eye model) and tested on each participant in MPIIGaze, recording average error for each participant. The results can be seen in Figure 3.16.

This plot illustrates which SynthesEyes models were useful for training and which ones were not. It can be seen that training with certain eye

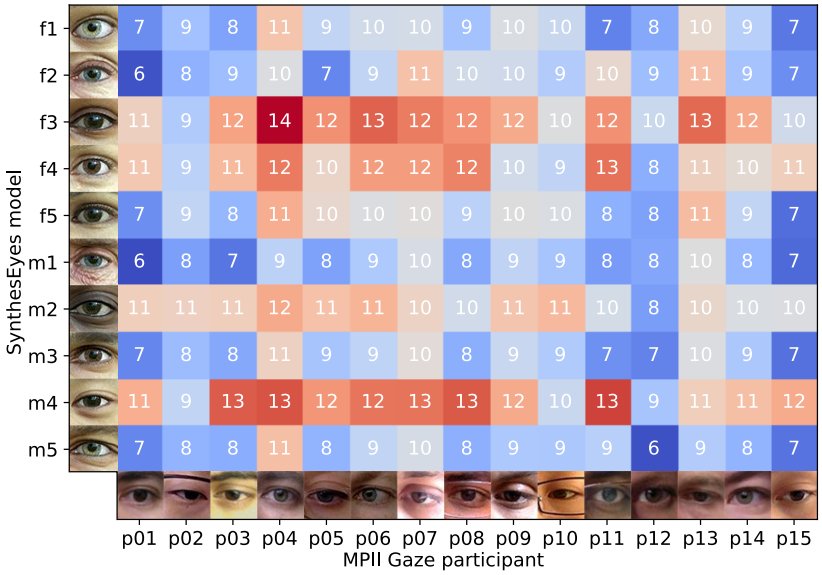


Figure 3.16: Per–eye-model gaze estimation mean errors on MPIIGaze in degrees. Red represents worst scores. Note how some eye-models have proved more useful than others for training. For example, a CNN trained with images from `f3`, a dark-skinned female with brown eyes, perform particularly badly on images of `p04`, a fair-skinned male with blue eyes.

models lead to poor generalization, for example `f3`, `m2`, and `m4`, perhaps due to differences in skin-tone and eye-shape. Similarly, learning with an Asian-shaped eye-region model (`m4`) leads poor performance for non-Asian-shape eye-region test participants. Also, total errors for some target participants are lower than for others, perhaps because of simpler eye-region shape that is matched to the training images. Although intuitive, these experiments further confirm the importance of correctly covering appearance variations in the training data. They also open up potential directions for future work, including person-specific adaptation of the renderings and gaze estimation systems.

3.6 Summary

In this chapter I presented my first investigations into learning-by-synthesis for appearance-based gaze estimation. Rather than present a new algorithm for gaze estimation itself, I described in detail how to synthesize perfectly labelled realistic images of the human eye. At the core of my method is a computer graphics pipeline that uses a collection of dynamic eye-region models obtained from head scans to generate images for a wide range of head poses, gaze directions, and illumination conditions.

I demonstrated that systems trained with SynthesEyes data can outperform state of the art methods for eye-shape registration and cross-dataset appearance-based gaze estimation in the wild. These results are promising and underline the significant potential of such learning-by-synthesis approaches particularly in combination with recent large-scale supervised methods.

4

Eye region morphable model

In the previous chapter I showed how I trained eye tracking systems with rendered images. Though this system performed well, its training data was still lacking in a certain respect: a poor range of **inter-person variability**. It contained only ten virtual participants, far fewer than is normally considered sufficient for a system to generalize well to unseen people. For comparison, training databases for face analysis tasks generally contain hundreds of different people (Sagonas et al., 2016), e.g. 337 participants in Multi-PIE (Gross et al., 2007), and 126 participants in FFRGC-V2 (Phillips et al., 2005).

Having too few training participants can be problematic for learning methods that are prone to over-fitting. For example, if a neural network learns to predict which specific training participant of out ten it is observing, it can learn participant-specific features that reduce its error at training-time. But these participant-specific features will not help it generalize to unseen test participants. On the other hand, having lots of training participants that all look similar will not help either. For example, if a training set contains only Caucasians, the resulting system might not learn features that apply to African or Asian people.

In this chapter I explain how I built a generative eye region model, allowing me to produce a large range of inter-person variability. I first explain what morphable models are, and why previous models are unsuitable for synthesising realistic eye images. I then describe a new morphable model that correctly treats the face and eyes as separate parts, the first of its kind. Crucially, this model allows independent eyeball movement – the primary indicator of gaze direction. The aim of this model was to allow me to take learning-by-synthesis a step further, generating better training datasets with an improved range

of eye appearance variation (Chapter 5). Furthermore, by fitting our model to images, I was also able to explore analysis-by-synthesis methods for gaze estimation for the fist time (Chapter 6).

4.1 3D morphable models

A *3D morphable model* (3DMM) is a statistically-derived generative model of the face. It comprises linear models of shape and texture which describe a face as the average face plus a weighted sum of basis faces that represent modes of variation (see Figure 4.2). They are constructed from a training set of face scans that have been brought into *correspondence* – meshes share the same topology, and vertices share the same anatomic semantics. The average face and these modes of variation are extracted using dimensionality reduction, typically Principal Components Analysis (PCA) (Blanz and Vetter, 1999). They are often combined with models of scene illumination and camera projection to analyse faces in images in an illumination-invariant and pose-invariant manner.

In past literature, the term *albedo* has been used instead of *texture*. In this dissertation they mean the same thing.

Since their introduction nearly two decades ago (Blanz and Vetter, 1999), 3DMMs have become a well established tool for many face-related tasks in computer vision. Their popularity can be attributed to two key benefits. 1. They provide a **powerful prior** for recovering 3D facial shape and texture in data-deficient scenarios, e.g. 2D images – an ill-posed problem (Blanz and Vetter, 1999; Romdhani and Vetter, 2005; Aldrian and Smith, 2013). Given an image of a face, these systems use model fitting to recover the most likely 3DMM parameters that represent that face. 2. They provide a **compact representation** of a face by encoding it using the learned shape and texture bases. This dimensionality reduction is key in making certain face-analysis tasks like face recognition (Paysan et al., 2009) and age estimation (Booth et al., 2016) more robust to changes in illumination and pose.

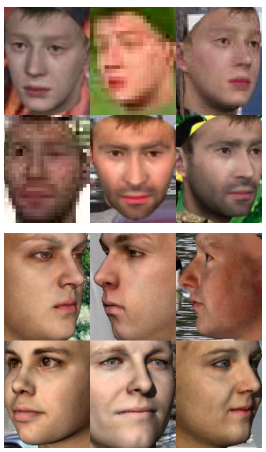


Figure 4.1: Synthetic training data used by Rätsch et al. (2012) (top) and Scherbaum et al. (2013) (bottom). Note the neutral facial expressions and gaze.

4.1.1 Learning-by-synthesis with 3DMMs

Due to their generative nature, 3DMMs are well-suited to learning-by-synthesis. New faces can be generated by perturbing the average face with random amounts of the basis faces. In this way, researchers have used 3DMMs to generate training sets with a guaranteed amount of variation – more convenient than manually collecting a large amount of data (see Figure 4.1). For example, Huang et al. (2003) captured faces from images, and re-rendered them under different poses and illumination for training a component-based face recognition system.

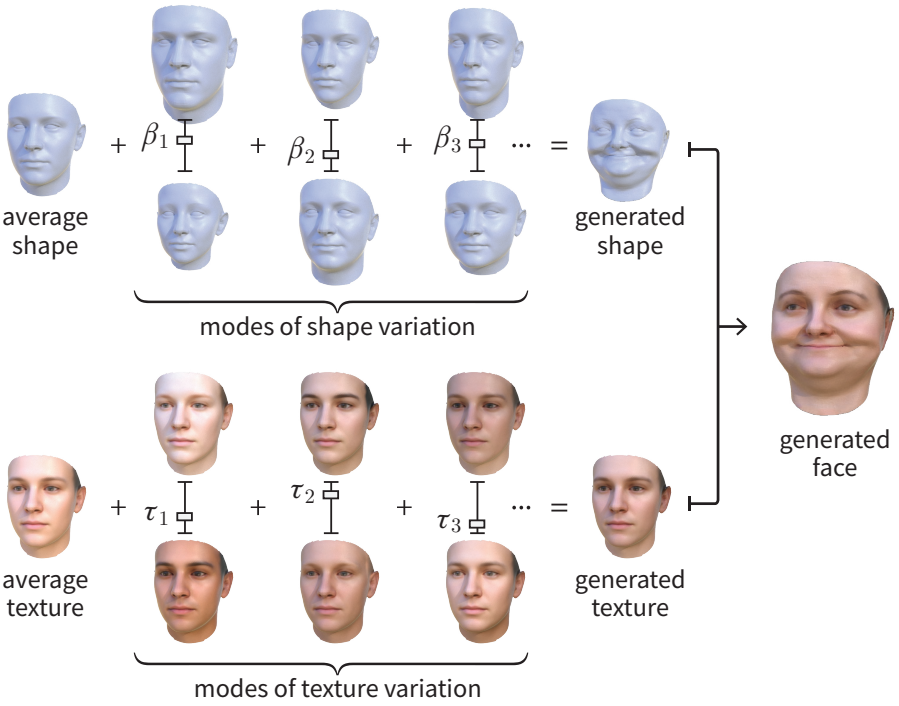


Figure 4.2: How a 3D morphable model works. A face is represented as the average face, plus a weighted sum of basis faces representing modes of facial variation. The primary mode of shape variation captures the differences between big and small faces, and the primary mode of texture variation captures the differences between light and dark skin.

Learning-by-synthesis has also been applied to head-pose estimation, with both Rätsch et al. (2012) and Fanelli et al. (2013) rendering training images of heads under various rotations in RGB and depth-only respectively. Scherbaum et al. (2013) trained a face detector with 3DMM images, and stressed the benefits of having full control over the variation in a training set. This allowed them to avoid potential biases in pose or appearance that might be present for datasets collected from the internet e.g. most face images present them facing towards the camera. These previous systems focussed on full-face analysis tasks, and avoided faces with non-neutral expression.

Inspired by these previous works, my plan was to use a 3DMM to generate images of eyes looking in different directions for training a gaze estimation system with learning-by-synthesis. So instead of rendering a discrete collection of ten head models as before (Chapter 3), I could instead generate a continuous stream of eye region models, and render these to produce training data with improved facial appearance variation. Unfortunately, previous 3DMMs have failed to accurately model the eyes, making them unsuitable for our task.

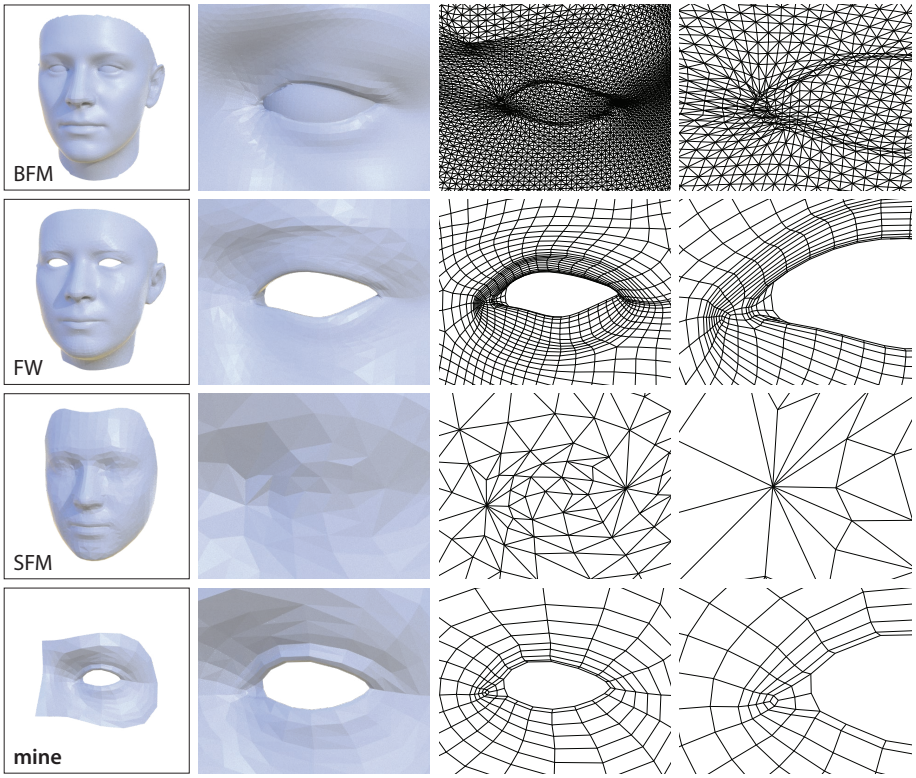


Figure 4.3: A comparison of previous models to my new eye region model. Note how details like the inner eye corner and eyelid margins are not present in the SFM and BFM. For the FW model, these small details have been copied from a template rather than fit to scan data. In contrast, my model cleanly captures these details with an efficient topology well suited for animation.

4.1.2 Previous face models

Though much time has passed since they were first presented, relatively few 3DMMs are available to the public. This is because they are difficult to construct. Collecting the 3D face scans is time consuming and expensive, and accurately bringing the scans into correspondence is a challenging task. I now present the most commonly used and most recent face models available, and explain why they are not suitable for synthesizing eye images.

Basel Face Model (BFM) (Paysan et al., 2009)

The BFM is currently the most widely used 3DMM. It was released to the public in 2009 to help bring 3DMM techniques mainstream, since previous models were not freely available. Limitations of the scanning equipment meant that the authors failed to accurately capture eye detail, and so replaced the eye with a smooth proxy mesh (see Figure 4.3, top row). Since this proxy covers the caruncle and eyelid margins, eye images rendered with the BFM miss these important details and appear

unnatural. Additionally, the BFM's mesh is extremely high resolution (>50K vertices), making it unsuitable for real time manipulation.

Figure 4.4: Raw scan data used in FW. Note how eye details have been smoothed out.

FaceWarehouse (FW) (Cao et al., 2014b)

Cao et al. (2014b) released FaceWarehouse to provide a comprehensive database of 3D facial expressions to the community. While not itself a 3DMM, the database includes registered scans of 150 individuals with various facial expressions – data that has since been used to build linear face models (Thies et al., 2015; Cao et al., 2014a). The scans were brought into correspondence by deforming a template mesh to match Kinect data. However, Kinect face scans are low resolution (see Figure 4.4), so eyes details in the FW model cannot be accurate. Any details visible in Figure 4.3 (second row) have instead been copied across from the source face template. Furthermore, as only a Kinect was used, the corresponding color data is low quality, preventing us building a good texture model.

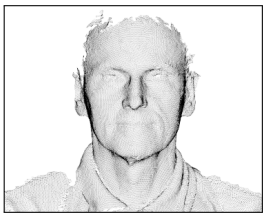


Figure 4.5: Raw scan data used in SFM. Note that eye details are barely visible.

Surrey Face Model (SFM) (Huber et al., 2016)

Huber et al. (2016) recently released the multi-resolution SFM as an alternative to the BFM. The SFM includes lower resolution models, better suited towards real time applications. However, all eye features have been smoothed away, leaving flat patch where the eyeball should be (see Figure 4.3 third row). As no detailed parts of the eye are discernible it is inappropriate for synthesizing close-up eye images. Furthermore, the source scans for the SFM remain of similar quality to previous models (see Figure 4.5), so there is no hope of accurate eye detail being captured.

4.1.3 Challenges of capturing the eye region in 3D

The issues encountered by previous models can be attributed to two challenges of the eye region: its **structure** and **motion**.

First, I will explain why the eye's structure makes it difficult to reconstruct in 3D. The eyeball is a complex organ, composed of multiple layers of transparent, refractive, and reflective material. This prevents the use of traditional 3D reconstruction techniques like photogrammetry that assume opaque non-transparent surfaces. As a result the corneal bump cannot be reconstructed, and the iris appears sunken and blurry. It therefore makes sense to remove the eyeball from each face scan (as in FW) as it is inaccurate. Furthermore, the eye is surrounded by eyelashes – hundreds of tiny self-similar hairs that can confound reconstruction methods. These cannot be discriminated correctly by photogrammetry, leading to a clumpy and noisy reconstruction of

the eyelashes and eyelids. It is therefore important to use scans of the highest quality to have a chance of correctly modelling eye detail.

Second, I will explain why eyeball motion has been largely ignored by previous face models. To model facial expressions, some previous facial models have been extended to capture expressions that form through deforming the skin (e.g. mouth opening, eyebrow raising) (Thies et al., 2016). Models like the BFM and SFM include the eyes as part of their face mesh, allowing them to be fit to images of faces with neutral gaze. This fixed facial topology does not allow eyes to move independently, so prevents these models being used to model represent non-neutral gaze expressions. While skin-only facial expressions can be modelled accurately by deforming the face mesh, this is not appropriate for eye movement. This is because eyeballs rotate freely in their sockets, so should not be attached to the eyelids of the face mesh. Previous work has side-stepped eye movement by removing the eyes from the mesh, and ignoring them during model-fitting (Thies et al., 2016; Cao et al., 2014a). Instead, eyeballs should be treated as separate objects from the skin, allowing them to be fit separately.

4.2 My multi-part eye region model

Taking these challenges into account, I designed a new multi-part 3DMM of the eye region. It captures shape and texture variation of both the facial eye region and eyeball, while also allowing for articulated independent eyeball motion. It consists of two main parts: a generative model of the facial eye region, and an articulated model of the eyeball. In the following sections I describe the parts that make up our model, how I pose them to simulate eye gaze, and how I render them to synthesize eye images.

4.3 Part 1: the facial eye region

Pictures of the eye also contain a portion of the face. In this section I describe the largest part of my model – the facial eye region.

4.3.1 Data acquisition

In order to build my generative morphable model, I needed 3D data of the eye region. I acquired high resolution head scans from the same online store as used in Chapter 3. An example can be seen on the left of Figure 4.6. Short scanning time is critical for capturing facial detail, as

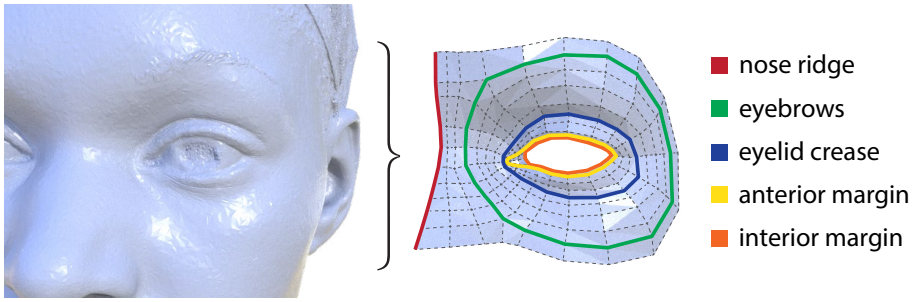


Figure 4.6: Retopologizing an original cleaned head scan (5M vertices) into my more efficient generic eye region mesh (229 vertices) The coloured strokes show important edge loops that I manually positioned to ensure mesh topology matches real life anatomic structure, allowing for realistic deformation.

even tiny facial movements can lead to a blurry reconstruction. While previous work used scanners with $\sim$1s acquisition time (Paysan et al., 2009), the models I used were captured in under $^1/_{10,000}$ of a second, resulting in high detail (0.1mm resolution geometry). I acquired $c = 22$ scans from an online store[1] (7 female, 15 male) covering different ages, eye shapes, facial bone structure, and skin tone.

4.3.2 Scan registration

The first step of building a 3DMM is bringing the raw scan data into correspondence. To do this, I retopologized each original high-resolution mesh so that semantically identical points (e.g. points along the interior margin or nose ridge) shared the same vertex in a lower resolution domain or topology. In Chapter 3 I retopologized each head-scan separately, resulting in ten new topologies for ten initial meshes. This then required each mesh to be animated separately, a time-consuming task. To bring the new set of scans into correspondence I instead registered all eye regions with a single generic topology, as shown in Figure 4.6.

I carefully designed this topology so the edge loops (sets of connected edges) would match the real life anatomic structure, e.g. the oculus orbicularis, while also faithfully capturing the original shape of the scanned eye region. These edge loops allow more realistic animation as mesh deformation matches that of real flesh and muscles (Orvalho et al., 2012). The topology does not include the eyeball, as we wish to pose that separately. I also uv-unwrapped the topology so each vertex is associated with a uv coordinate – this allows us to apply textures to the eye region (Figure 4.7). Once I had manually registered each

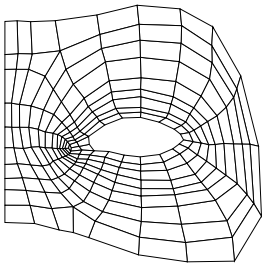


Figure 4.7: The uv map of the facial eye region.

[1] Ten24 3D Scan Store – `http://www.3dscanstore.com/`

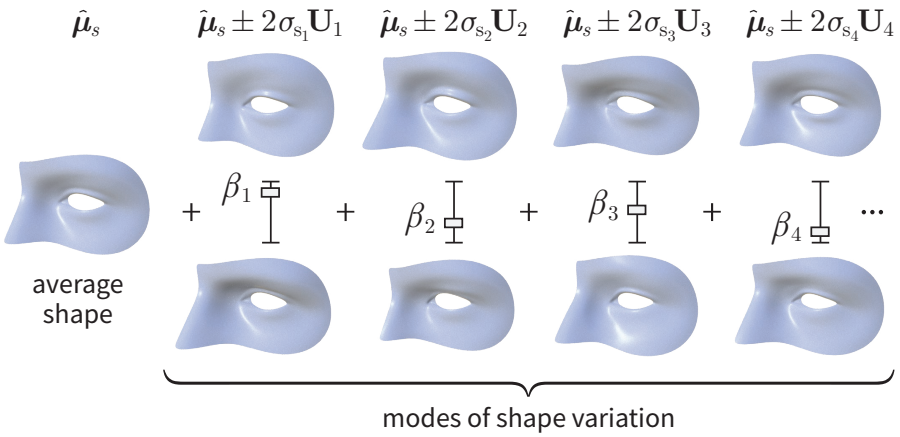


Figure 4.8: The linear shape model $\mathcal{M}_{geo}$. The mean eye region shape $\hat{\boldsymbol{\mu}}_s$ is shown along with the first four modes of shape variation. The first mode $\mathbf{U}_1$ varies how inset / protruding the eye is. The shapes for $\mathbf{U}_1$ correspond to $\mathcal{M}_{geo}([+2, 0, \ldots])$ and $\mathcal{M}_{geo}([-2, 0, \ldots])$.

scan, I brought the textures into correspondence by transferring each original texture into into a new texture map defined in this uv space.

Following registration, we have c sets of points that represent shape, and c sets of RGB values that represent texture. Let us write shape as vector $\mathbf{s}$ (n vertices), and texture as vector $\mathbf{t}$ (m texels), encoded as $3n$ and $3m$ dimensional vectors respectively:

$$\mathbf{s} = [x_1, y_1, z_1, x_2, \ldots y_n, z_n]^T \in \mathbb{R}^{3n} \tag{4.1}$$

$$\mathbf{t} = [r_1, g_1, b_1, r_2, \ldots g_m, b_m]^T \in \mathbb{R}^{3m} \tag{4.2}$$

where x_i, y_i, z_i is the 3D position of the ith vertex, and r_j, b_j, g_j is the color of the jth texel.

4.3.3 Linear shape model $\mathcal{M}_{geo}$

We now have a training set of $c = 22$ shape vectors $\mathbf{s}_i, i \in [1, 22]$ that form a distribution in $3n$ dimensional space. Our aim is to model this distribution, so we can generate new face shapes that are similar to those in the training set, and examine observed shapes to determine their plausibility (Cootes et al., 2001).

Let us assume that facial shape $\mathbf{s}$ is distributed as a multivariate normal random variable

$$\mathbf{s} \sim \mathcal{N}_{3n}\left(\boldsymbol{\mu}_s,\ \boldsymbol{\Sigma}_s\right) \tag{4.3}$$

where $\boldsymbol{\mu}_s \in \mathbb{R}^{3n}$ is the mean 3D shape, and $\boldsymbol{\Sigma}_s \in \mathbb{R}^{3n \times 3n}$ is the covariance matrix.

Parameter estimation We now search for $\hat{\boldsymbol{\mu}}_s$ and $\hat{\boldsymbol{\Sigma}}_s$: unbiased estimators for $\boldsymbol{\mu}_s$ and $\boldsymbol{\Sigma}_s$ respectively. Let us gather the c shape vectors $\mathbf{s}_1, \mathbf{s}_2 \ldots \mathbf{s}_c$ into shape data matrix $\mathbf{S}$,

$$\mathbf{S} = \begin{bmatrix} x_{11} & y_{11} & z_{11} & x_{12} & \ldots & x_{1n} & y_{1n} & z_{1n} \\ x_{21} & y_{21} & z_{21} & x_{22} & \ldots & x_{2n} & y_{2n} & z_{2n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{c1} & y_{c1} & z_{c1} & x_{c2} & \ldots & x_{cn} & y_{cn} & z_{cn} \end{bmatrix} \in \mathbb{R}^{c \times 3n} \tag{4.4}$$

$\hat{\boldsymbol{\mu}}_s$ can then be calculated as the column-wise mean of $\mathbf{S}$,

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{c} \sum_{i=1}^{c} \mathbf{S}_i \tag{4.5}$$

and we can then derive an estimate of the covariance matrix $\hat{\boldsymbol{\Sigma}}_s$

$$\hat{\boldsymbol{\Sigma}}_s = \frac{1}{c-1} \left(\mathbf{S} - \hat{\boldsymbol{\mu}}_s\right) \left(\mathbf{S} - \hat{\boldsymbol{\mu}}_s\right)^T \tag{4.6}$$

Model definition We seek a shape model $\mathcal{M}_{geo}$ that allows us to generate face shapes $\mathbf{s}^*$. To simplify the problem, we apply Principal Component Analysis (PCA) to the data to reduce its dimensionality from $3n$ to $c-1$. First, let us define a generative, parameterized shape model of the form $\mathbf{s}^* = \mathcal{M}_{geo}(\boldsymbol{\beta})$ as follows:

$$\mathcal{M}_{geo}\left(\boldsymbol{\beta}\right) = \hat{\boldsymbol{\mu}}_s + \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma}_s)\,\boldsymbol{\beta} \tag{4.7}$$

$$= \hat{\boldsymbol{\mu}}_s + \sum_{i=1}^{c-1} \beta_i\, \sigma_{s_i} \mathbf{U}_i \tag{4.8}$$

where $\mathbf{U} \in \mathbb{R}^{3n \times 3n}$ is a matrix representing the modes of shape variation, $\boldsymbol{\sigma}_s \in \mathbb{R}^{3n}$ are the standard deviations of each of those modes, and $\boldsymbol{\beta} \in \mathbb{R}^{3n}$ is a vector of shape parameters. As $\hat{\boldsymbol{\Sigma}}_s$ is a real symmetric matrix, $\mathbf{U}$ can be determined through the eigen decomposition of the covariance matrix $\hat{\boldsymbol{\Sigma}}_s$:

$$\hat{\boldsymbol{\Sigma}}_s = \mathbf{U}\boldsymbol{\Lambda}_s\mathbf{U}^T \tag{4.9}$$

where $\mathbf{U}$ is an orthogonal matrix whose i^{th} column is the eigenvector $\mathbf{U}_i$ of $\hat{\boldsymbol{\Sigma}}_s$, and $\boldsymbol{\Lambda}_s$ is a diagonal matrix whose diagonal entries $\Lambda_{s_{ii}}$ are the eigenvalues λ_{s_i} of $\hat{\boldsymbol{\Sigma}}_s$ (both sorted so that $\lambda_{s_i} \geq \lambda_{s_{i+1}}$).

These eigenvectors $\mathbf{U}_i$ represent the orthonormal basis vectors that best describe variation in the model distribution – these are the principal components. The variance of the i^{th} mode of variation across the training set is given by the corresponding eigenvalue λ_i. We scale each mode by its standard deviation $\sigma_{s_i} = \sqrt{\lambda_i}$ so $\mathcal{M}_{geo}$ is more evenly parameterized across $\boldsymbol{\beta}$.

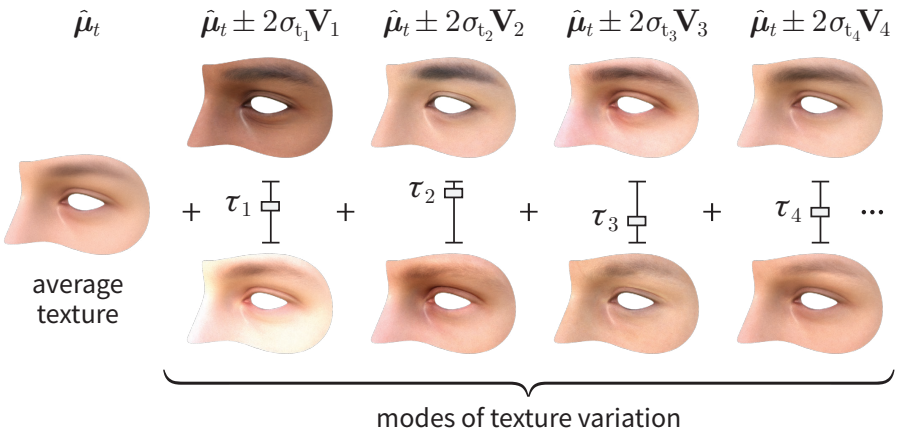


Figure 4.9: The linear texture model $\mathcal{M}_{tex}$ shown rendered onto the average shape. The mean eye region texture $\hat{\boldsymbol{\mu}}_t$ is shown with the first four modes of texture variation. The first mode $\mathbf{V}_1$ varies between light and dark skin. The textures for $\mathbf{V}_1$ correspond to $\mathcal{M}_{tex}([+2, 0, \ldots])$ and $\mathcal{M}_{tex}([-2, 0, \ldots])$.

It can be shown that only the first $c-1$ eigenvectors $\mathbf{U}_i$ have non-zero eigenvalues: $\lambda_i = 0$ for $i \geq c$ (Cootes et al., 2001). We therefore only need to consider the first $c-1$ columns of $\mathbf{U}$ and the first $c-1$ parameters in β when generating shapes $\mathbf{s}^*$.

Examples Figure 4.8 shows the mean shape $\hat{\boldsymbol{\mu}}$ along with visualizations of the four most important degrees of shape variation.

4.3.4 Linear texture model $\mathcal{M}_{tex}$

We derive the generative texture model $\mathcal{M}_{tex}$ in a similar way to $\mathcal{M}_{geo}$. We start with a training set of c texture vectors $\mathbf{t}_i$ that form a distribution in $3m$ dimensional space. We assume that facial texture $\mathbf{t}$ is distributed as a multivariate normal random variable

$$\mathbf{t} \sim \mathcal{N}_{3m}\left(\boldsymbol{\mu}_t,\ \boldsymbol{\Sigma}_t\right) \tag{4.10}$$

where $\boldsymbol{\mu}_t \in \mathbb{R}^{3m}$ is the mean facial texture, and $\boldsymbol{\Sigma}_t \in \mathbb{R}^{3m \times 3m}$ is the covariance matrix.

Parameter estimation To find unbiased estimators $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{\Sigma}}_t$, we first gather the facial textures into a texture data matrix $\mathbf{T} \in \mathbb{R}^{c \times 3m}$ whose i^{th} row corresponds to the texture vector $\mathbf{t}_i$. $\hat{\boldsymbol{\mu}}_t$ and $\hat{\boldsymbol{\Sigma}}_t$ can then be calculated as follows:

$$\hat{\boldsymbol{\mu}}_t = \frac{1}{c}\sum_{i=1}^{c} \mathbf{T}_i \tag{4.11}$$

$$\hat{\boldsymbol{\Sigma}}_t = \frac{1}{c-1}\left(\mathbf{T} - \hat{\boldsymbol{\mu}}_t\right)\left(\mathbf{T} - \hat{\boldsymbol{\mu}}_t\right)^T \tag{4.12}$$

Model definition We now seek a shape model $\mathcal{M}_{tex}$ that allows us to generate face textures as $\mathbf{t}^* = \mathcal{M}_{tex}(\boldsymbol{\tau})$. Again, we use PCA to simplify the problem. Our parameterized generative texture model is defined as follows:

$$\mathcal{M}_{tex}(\boldsymbol{\tau}) = \hat{\boldsymbol{\mu}}_t + \mathbf{V}\,\mathrm{diag}(\boldsymbol{\sigma}_t)\,\boldsymbol{\tau} \tag{4.13}$$

$$= \hat{\boldsymbol{\mu}}_t + \sum_{i=1}^{c-1} \tau_i \sigma_{t_i} \mathbf{V}_i \tag{4.14}$$

where $V \in \mathbb{R}^{3m \times 3m}$ is a matrix representing the modes of texture variation, $\boldsymbol{\sigma}_t \in \mathbb{R}^{3n}$ are the standard deviations of each of those modes, and $\boldsymbol{\tau} \in \mathbb{R}^{3m}$ is a vector of model parameters. $\mathbf{V}$ is found through the eigen decomposition of $\hat{\boldsymbol{\Sigma}}_t$,

$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T \tag{4.15}$$

where $\mathbf{V}$ is an orthogonal matrix whose i^{th} column is the eigenvector $\mathbf{V}_i$ of $\hat{\boldsymbol{\Sigma}}_t$, and Λ_t is a diagonal matrix whose diagonal entries $\Lambda_{st_{ii}}$ are the eigenvalues λ_{t_i} of $\hat{\boldsymbol{\Sigma}}_t$ (both sorted so that $\lambda_{t_i} \geq \lambda_{t_{i+1}}$). As before, we scale each mode by its standard deviation σ_{s_i}, and we only consider the first $c-1$ columns of $\mathbf{V}$ and the first $c-1$ parameters in $\boldsymbol{\tau}$.

Examples Figure 4.9 shows the mean texture $\hat{\boldsymbol{\mu}}_t$ along with visualizations of the four most important degrees of texture variation.

4.4 Part 2: the eyeball

Eye gaze direction is determined by the orientation of the eyeball. My multi-part model therefore includes an eyeball model that can be rotated independently from the rest of the face to simulate changes in gaze direction.

4.4.1 Eyeball shape

The visible region of the eyeball is composed of three parts: the white *sclera*, the transparent *cornea*, and the colored *iris*. As is standard in computer graphics (Ruhland et al., 2014), I approximate the eye's shape with two spheres: a large one representing the sclera ($r = 12mm$), and a small one representing the corneal bulge ($r = 8mm$). The boundary between these two spheres is the *limbus* – the outer edge of the iris. This boundary where the spheres meet was smoothed using vertex averaging to provide a continuous transition between cornea and sclera. The resulting mesh represents the exterior surface of the eyeball, and comprises 546 vertices in total.

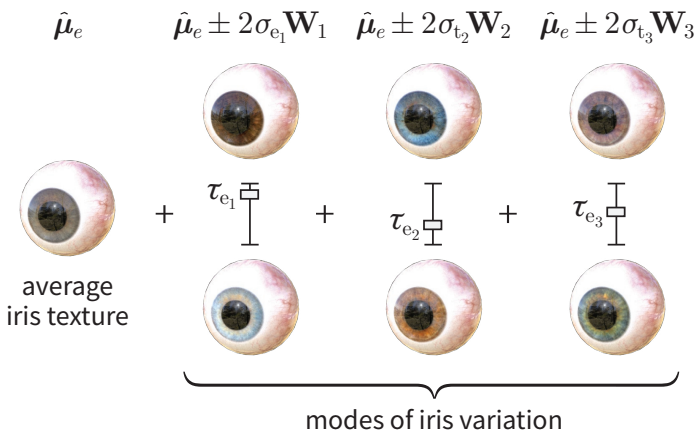


Figure 4.10: The linear iris texture model $\mathcal{M}_{eye}$. The mean iris texture $\hat{\mu}_{t_{eye}}$ is shown with the three modes of iris texture variation. The primary mode W_1 varies between light and dark coloured eyes.

Eyeballs vary in shape, both between people and over time. Iris size variation is modelled by simply scaling all iris vertices around their centre using procedural animation. Pupillary dilation and contraction is modelled by altering texture lookups, scaling them towards or away from the pupil centre.

4.4.2 Iris texture model $\mathcal{M}_{eye}$

Eyeballs also vary in colour between people. This variation is modelled using a linear texture model $\mathcal{M}_{eye}$ built in the same way as $\mathcal{M}_{tex}$. I collected and aligned a set of high resolution eye images to use as training data (see Figure 4.11). Following PCA on this source data, we can generate new eyeball textures $\mathbf{t}^*_{eye}$ in the following way:

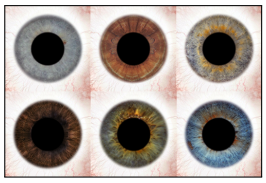


Figure 4.11: Manually aligned source images for the generative iris texture model.

$$\mathcal{M}_{eye}\left(\boldsymbol{\tau}_{eye}\right) = \hat{\boldsymbol{\mu}}_e + \mathbf{W}\,\mathrm{diag}(\boldsymbol{\sigma}_e)\,\boldsymbol{\tau}_e \tag{4.16}$$

where $\boldsymbol{\tau}_e$ is a vector of eye texture parameters, $\hat{\boldsymbol{\mu}}_e$ is the estimated average eyeball texture, $\mathbf{W}$ is a matrix representing orthonormal modes of eye texture variation, and $\boldsymbol{\sigma}_{t_e}$ are the standard deviations of each of those modes. Figure 4.10 shows some examples of the mean eye texture along with the primary modes of variation.

4.5 Posing our model

The multi-part model is defined with neutral gaze, representing someone looking directly ahead. The world origin and eyeball origin are aligned, and the eyeball gazes along the Z axis. The eye region model must therefore be posed to simulate different gaze directions.

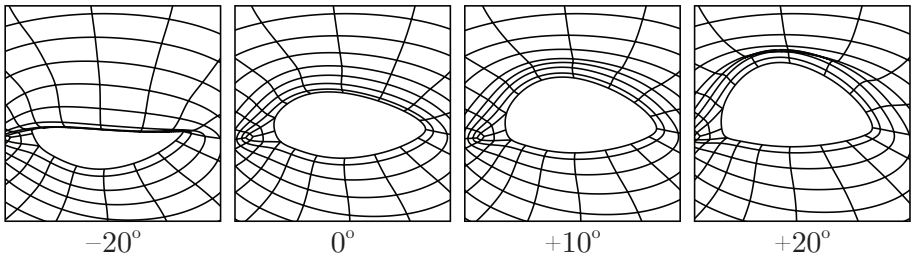


Figure 4.12: Eyelid motion corresponding to eye gaze pitch of $-20°$, $0°$, $+10°$, and $+20°$. The face mesh is posed using procedural animation.

4.5.1 Eyeball rotation

The eyeball model is rotated by applying rotation transforms to its model-to-world transform $\mathbf{M}_{eye}$. Gaze direction can be represented in terms of eyeball pitch θ_{pitch} and yaw θ_{yaw} angles, so the model-to-world matrix can be defined as

$$\mathbf{M}_{eye} = \mathbf{R}_Y\left(\theta_{yaw}\right)\mathbf{R}_X\left(\theta_{pitch}\right) \tag{4.17}$$

Where $\mathbf{R}_Y\left(\theta_{yaw}\right)$ is a rotation about the Y axis of θ_{yaw}, and $\mathbf{R}_X\left(\theta_{pitch}\right)$ is a rotation about the X axis of θ_{pitch}.

4.5.2 Procedural eyelid motion

When the eyeball moves, the eyelids move with it. As the facial eye region model represents neutral gaze ($0°$ eyeball pitch and yaw), we must animate the mesh to represent lid motion.

There are two main methods for animating faces: 1) skeletal animation: a mesh's surface is deformed by posing an underlying skeleton, and 2) blend shapes: different deformed versions of a mesh are stored as separate keyframes, and are interpolated between. However, these methods are generally applied to a specific instance of a face, not a generative face model. I instead use **procedural animation** to position eyelid vertices using mathematical functions rather than artist-driven facial rigs. This way we can ensure consistent animation across different facial shapes.

Eyelid movement can be broadly described as a rotation, with different parts of the lid having different rotational axes (Malbouisson et al., 2005). I therefore model eyelid rotation by applying different rotation transforms to the different vertices around the eye. Following the edge-loops in our topology, the rotation of the j^{th} vertex in the i^{th} edge loop $\boldsymbol{v}_{ij}$ is defined follows:

$$\boldsymbol{v}'_{ij} = \boldsymbol{R}\left(\boldsymbol{a}_i, \boldsymbol{p}_{ij}, \theta_{ij}\right)\boldsymbol{v}_{ij} \tag{4.18}$$

Where $\boldsymbol{R}(\boldsymbol{a}, \boldsymbol{p}, \theta)$ describes a rotation of θ around axis $\boldsymbol{a}$ about pivot $\boldsymbol{p}$. Each edge loop's rotational axis $\boldsymbol{a}_i$ is defined as the offset between its eye corner vertices, and pivot $\boldsymbol{p}_{ij}$ interpolates between eye corners and eyeball center to ensure vertices near the eye corners are not displaced too far. Angles θ are defined separately for upper and lower lids using measurements taken from an empirical study (Malbouisson et al., 2005). Rotational amount θ decays for the outer edge loops to simulate elastic stretching of the surrounding skin and flesh. Examples of this animation can be seen in Figure 4.12.

4.5.3 Shrinkwrapping the eye region

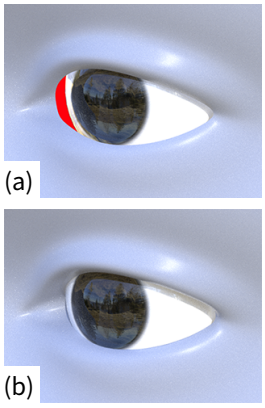


Figure 4.13: (a): Gaps (red) can appear in the model. (b): Shrinkwrapping fixes this.

Although I aligned the source face scans with the eyeball model as best as I could, there is no guarantee that the facial mesh will neatly meet the eyeball, and in some cases it may intersect it. This is unrealistic. To avoid this behaviour, the inner edge loops must be *shrinkwrapped* (see Figure 4.13) onto the eyeball surface, filling in any unwanted gaps between the mesh and eyeball.

Vertices in the innermost edge loop (the interior margin) are projected directly onto the eyeball surface. The resting place of each vertex is computed by intersecting a ray with the eyeball; the ray starts from the vertex's original position, and has direction pointing towards the eyeball centre. These intersection calculations are efficient as we can approximate the eyeball mesh with two sphere primitives. Finally, vertices in outer edge loops are displaced towards their inner edge loop neighbours simulate skin elasticity. This displacement is performed as a weighted average of the original vertex position and the position of its neighbour in the inner edge loop.

4.6 Rendering the model

This chapter concerned how the multi-part model is specified in terms of geometry, texture, and animation. This 3D geometry was rendered with several different tools and methods, depending on what was required of it. The following were used:

Cycles – A photorealistic unbiased path tracing renderer integrated in Blender. Cycles was used to render the SynthesEyes dataset (Chapter 3), and produce many of the figures and diagrams in this dissertation.

Unity – A game engine that provided a simple API for controlling the generative models, as well as a physically-based rasterizing

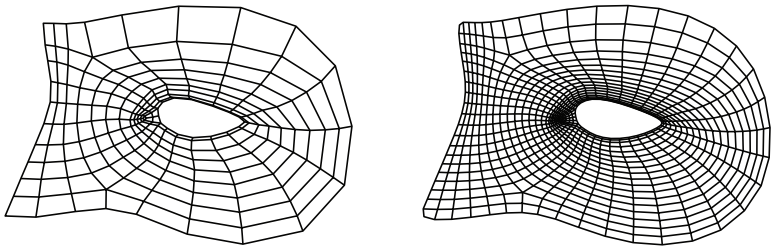


Figure 4.14: Since the shape model can appear blocky, I smooth its surface using subdivision. Here is an example of a single step of Catmull and Clark (1978) subdivision as performed by Blender.

renderer for increased throughput compared to Cycles. Unity was used to generate a larger eye image dataset in (Chapter 5).

DirectX – A low-level graphics API that let me synthesise eye images with maximum efficiency. Without this speed, analysis-by-synthesis would not have been feasible (Chapter 6).

Future chapters will present how these three renderers were used in more detail. The rest of this section describes three additional graphics techniques that were used to improve realism: smoother skin using subdivision surfaces, physically based iris refraction, and particle-driven eyelashes.

4.6.1 Smoother skin using subdivision surfaces

Skin has a smooth surface. Since the facial eye region mesh is low resolution, it may appear unnatural blocky when rendered from close up. This blockiness can be avoided by subdividing the facial mesh, smoothing its surface and edges by increasing its polygon count. Applying a subdivision step to a mesh involves introducing new vertices and repositioning old vertices. The positions of these vertices are calculated as a weighted average of the original vertices in a local neighbourhood.

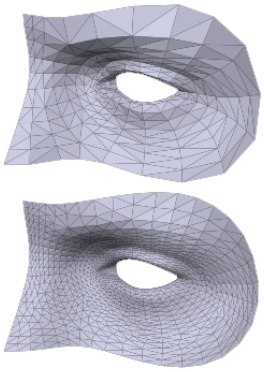


Figure 4.15: Top: the facial eye region after triangulation. Bottom: the triangulated mesh following a step of Loop (1987) subdivision.

Converting the facial quad-mesh to triangles doesn't affect the shape model as it concerns vertex position only.

Two different subdivision schemes were used, depending on how the model was rendered.

Catmull-Clark subdivision As shown in Figure 4.6, the facial mesh consists of quadrilateral polygons, or "quads". We can therefore use the popular quad-based Catmull and Clark (1978) subdivision scheme, which is provided by Blender (Figure 4.14). This is the subdivision method used when rendering with Cycles.

Loop subdivision Although our topology is originally composed of quads, it must be converted to triangles before it can be rendered with GPU rasterization (e.g. with Unity). Once the quad mesh has

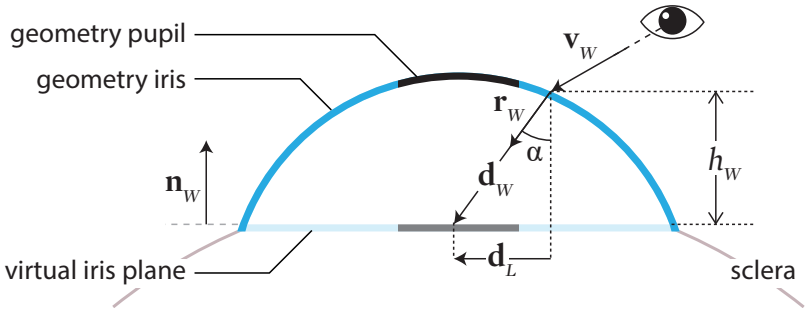
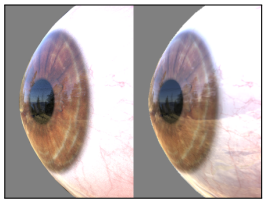


Figure 4.16: The layered structure of the eyeball is simulated in real time using a special shader. The idea is to calculate a uv-space texture offset $\mathbf{d}_L$ that corresponds to corneal refraction.

been triangulated, I apply Loop (1987) subdivision (Figure 4.15). If we want to render multiple images of a subdivision surface, it would be computationally wasteful to recalculate the subdivision stencils for each vertex every frame. Instead, I precomputed the resulting mesh topology and per-vertex stencil weights for a single step of Loop (1987) subdivision, and apply them whenever facial shape changes.

4.6.2 Physically based iris refraction



Blender Unity

Figure 4.17: Left: two part eyeball rendered offline with Blender Cycles. Right: physically correct refraction in real time with Unity.

When using Cycles, I use a two-part eyeball model similar to the one in Chapter 3 to represent the refractive part of the eyeball accurately as possible. However, when rendering with Unity or DirectX, ray-traced refraction is not natively available to us. For these real time engines I instead implemented physically accurate corneal refraction using a custom *fragment shader* – a GPU program that processes each pixel in a graphics pipeline.

I followed the approach of Jimenez et al. (2012), and altered each texture look-up with a calculated texture-space offset $\mathbf{d}_L$ (see Figure 4.16). For each pixel on the surface of the cornea:

$$\text{refracted pixel color} = \text{EyeTexture}(\mathbf{uv} + \mathbf{d}_L) \quad \text{where} \tag{4.19}$$

$$\mathbf{d}_L = \left(\mathbf{M}^{-1}\mathbf{d}_W\right)_{xy} \quad \text{and} \quad \mathbf{d}_W = \frac{h_W}{-\hat{\mathbf{n}}_W \cdot \hat{\mathbf{r}}_W} \tag{4.20}$$

$\mathbf{uv}$ is the original texture coordinate, $\mathbf{M}$ is the eyeball model-to-world transform, $\mathbf{d}_W$ is the world-space vector between the corneal surface and virtual iris, h_W is the world-space height between cornea and iris, $\hat{\mathbf{n}}_W$ is the eyeball's gaze direction in world-space, and $\hat{\mathbf{r}}_W$ is the world-space refracted view direction through the iris.

An example is shown in Figure 4.17. The eyeball on the left was rendered with Cycles, taking several seconds. The eyeball on the right

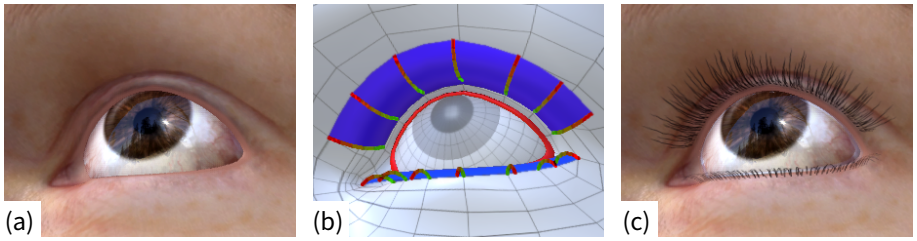


Figure 4.18: (a) The eye rendered without eyelashes. (b) The interpolated eyelash mesh in blue, and guide hairs in green-to-red. (c) The final render with the semi-transparent eyelash mesh.

was rendered with Unity, taking only a few milliseconds. Clearly, the iris refraction shader is a good approximation of ray-traced refraction. We can therefore model the complex multi-layered structure of the eyeball using a single 3D mesh, in real time.

4.6.3 Eyelashes

In images of eyes, eyelashes appear as dark edge-like structures and can provide a visual cue when someone is looking downwards. In Unity and DirectX, these were modelled using directed particle effects. Hair particles start at the eyelid boundaries and grow outwards away from the eyeball, curling up or down depending on the eyelid. Hair particles are checked for collisions with the facial mesh during hair growth, and are redirected to avoid clipping. Computing hundreds of hair strands is expensive, so I instead grow ten guide hairs (five for each eyelid) and interpolate a piece of textured transparent eyelash geometry to them (see Figure 4.18). While not as realistic as the hair particles used by Cycles (Chapter 3), this approach is much faster.

4.7 Summary

In this chapter I described my generative 3D morphable model of the eye region. In subsection 4.1.2, I explained why previous 3DMMs were unsuitable for my work, and in section 4.3 and section 4.4 I described the two main parts of my model: the face and eyeball. This model plays a key role in the rest of the dissertation. In Chapter 5 I use it to generate more varied training data for learning-by-synthesis. In Chapters 6 and 7 I fit the model to images using analysis-by-synthesis.

5

Learning an appearance based gaze estimator from one million synthetic images

This chapter presents work published at the ***2016 Symposium on Eye Tracking Research & Applications*** *in Charleston, South Carolina, USA (Wood et al., 2016b). Tadas Baltrušaitis implemented the k-Nearest-Neighbour gaze estimation system.*

In this chapter I present *UnityEyes*, a research tool for rapidly synthesizing large numbers of eye region images for use as training data. UnityEyes couples a generative model of the human eye region with a real-time rendering engine. As described in Chapter 4, the eye region model is derived from high-resolution 3D face scans and captures variation in eye region shape, texture, and pose. This addresses one limitation of the work in Chapter 3 – too little (or too sparsely sampled) inter-person variability. However, the work in Chapter 3 suffers another limitation: the **rate at which data can be generated**.

For the SynthesEyes dataset, it took ∼5s to render each image using Cycles, Blender's photorealistic ray tracing rendering engine. This means it would be possible to generate about ∼15,000 images in about a day on a typical PC. For many problems, this amount of data is deemed sufficient, e.g. the eye region landmark tracker trained on SynthesEyes data (section 3.4). However, we know that other machine learning methods work better with more data, e.g. deep learning (LeCun et al., 2015). While it would be possible to generate massive

datasets using the SynthesEyes pipeline by running it for long periods of time, or renting a server farm, these options might not be available or convenient for a typical researcher.

UnityEyes trades a little realism for a lot of speed. By using rasterization instead of ray tracing, we can synthesize eye images over $200\times$ faster than before. This allowed me to synthesize a massive dataset of over 1,000,000 images that densely covers a wide range of different gaze directions and eye region appearances. I demonstrate the importance of covering this variability by showing competitive performance for device and person-independent appearance-based gaze estimation, despite only using a light-weight k-Nearest-Neighbour classifier. Furthermore, this massive dataset was used to estimate gaze in images from the 300-W dataset (Sagonas et al., 2016): an extremely challenging scenario that no previous work in gaze estimation has attempted to address.

Rasterization is much faster than ray tracing because GPUs contain special hardware dedicated to polygon rasterization.

The UnityEyes rendering tool has been made publicly available for the benefit of the research community.[1]

5.1 Related work

The work in this chapter is related to two types of previous work: rendering eyes in real time, and using morphable models for gaze estimation.

5.1.1 Rendering eyes in real time

For the work in Chapter 3, I studied how professional computer graphics artists approached the challenge of rendering a realistic eye. With the features provided by modern ray tracing renderers like Cycles, it is possible to model the complex structures of the eye region accurately enough to achieve photorealistic results. However, the offline renderers used by traditional computer artists are slow.

Figure 5.1: Jimenez et al.'s (2012) realistic eyeball effects turned off (a) and on (b).

Therefore for this chapter, I turned to previous work on rendering eyes in real time. Real time eyes have one major application: video games. Jimenez et al. (2012) was the first to describe the process in detail. Their comprehensive approach considers eyeball reflections, eye wetness, reflection occlusion, view refraction, light refraction (caustics), two-layer eye shading, eye redness, and ambient occlusion. Jimenez et al.'s (2012) work has since been built upon by other game development studios (Karis et al., 2016). These references were critical when it came to developing the UnityEyes eye shading pipeline.

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

5.1.2 Using morphable models for gaze estimation

My work is not the first to use morphable models for gaze estimation. Previous work used morphable face models to *frontalize* faces – transforming input images in a way to undo distortions from head pose. For example, Egger et al. (2014) fit the Basel Face Model (BFM) (Paysan et al., 2009) to input images to obtain a dense 3D reconstruction of a face. They then obtain aligned eye images by "unwrapping" the face into a canonical pose-invariant image. These aligned eye images are then used in a typical appearance-based gaze estimation system (regression from HOG features using random forests). Mora and Odobez (2014) used the BFM in a similar fashion, using it to un-rotate images of a person's head to align eye images. While frontalization is also possible using sparse methods like a typical facial landmark tracker (Baltrušaitis et al., 2016), the results may not be as accurate (though they will be faster).

The work in this chapter is different. It is the first to use morphable models for learning-by-synthesis for gaze estimation. The goal being, if we can generate *enough* training samples, we won't need frontalization. Given an eye image under unconstrained head pose, the machine learning system will hopefully have already seen a similar image at training time.

5.2 Generative eye region model

In this chapter, I used an early version of the model described in Chapter 4. To vary eye region shape, I used the PCA shape model $\mathcal{M}_{geo}$ as described in section 4.3:

$$\mathcal{M}_{geo}\left(\boldsymbol{\beta}_{face}\right) = \hat{\boldsymbol{\mu}}_{geo} + \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma}_{geo})\,\boldsymbol{\beta}_{face} \tag{5.1}$$

where $\hat{\boldsymbol{\mu}}_{geo}$ is the mean shape, $\mathbf{U}$ are the modes of shape variation, and $\boldsymbol{\sigma}_{geo}$ are the standard deviations of these modes.

For texture, I used a simpler exemplar-based approach. When generating an eye region, UnityEyes picks a texture uniformly at random from the 22 textures extracted from the face scans. The full set of textures can be seen in Figure 5.2. This approach was used because the linear texture model $\mathcal{M}_{tex}$ had not yet been built. However, since the source actors for the scans were diverse, the collection of textures provided a decent range of variability.

Otherwise, the model used is as described in Chapter 4. This includes procedural animation for eyelid motion and eyelid shrinkwrap-

Figure 5.2: The 22 eye region textures that are used in UnityEyes. When generating an eye region, a texture from this collection is chosen at random.

ping (section 4.5), smoother skin using subdivision surfaces (subsection 4.6.1), physically correct eyeball refraction (subsection 4.6.2), and directed particle effects for the eyelashes (subsection 4.6.3).

5.3 Synthesizing eye images

My goal was to rapidly create large, realistic, and varied datasets of eye images. Though the generative eye region model captures 3D shape variation, images of eyes also exhibit variation depending on pose and environmental illumination. In this section I describe the rendering engine used by UnityEyes, as well as how I parameterized the scene to produce illumination and pose variation.

5.3.1 Rendering the models

I used *Unity 5.2*[2] to render the eyeball and generative eye region model. Unity is a cross-platform game engine that provides support for various graphics APIs (Direct3D on Windows, OpenGL on Linux and Mac, etc.). It includes a state of the art graphics pipeline that supports physically-based shading and global illumination – important features for rendering consistently realistic images. It also provides a convenient API for modifying and manipulating 3D data. This was used to implement the generative shape model. As Unity was built for games,

[2] `https://unity3d.com/`

```
{
  # Landmarks in screen-space
  interior_margin_2d: [ (202.7042, 186.4788), ... ],
  caruncle_2d: [ (191.9471, 175.4047), ... ],
  iris_2d: [ (213.3930, 195.4109), ... ],
  eye_details: {
    look_vec: (-0.363, 0.093, -0.927), # in camera-space
    pupil_size: 0.05249219,
    iris_size: 0.9090334,
    iris_texture: eyeball_amber
  },
  lighting_details: ...   # Illumination details
  eye_region_details: ... # Shape PCA details
  head_pose: (351.2107, 161.3652, 0.0000)
}
```

Figure 5.3: Example JSON meta-data associated with a rendered eye image. Of particular note, `eye_details.look_vec` encodes the optical axis gaze direction in camera space, and `head_pose` encodes the rotational differences between camera and head. The 2D screen-space landmarks should be used for post-processing the images, e.g. for aligning them.

it has been thoroughly optimised and supports efficient deployment of custom shaders. This allowed me to write eyeball and skin material shaders that were compatible with the rest of Unity's physically based shading pipeline.

In its default configuration, UnityEyes is set up to render images at 400×300px resolution, and save them to disk. This took 23ms per image using a commodity GPU (Nvidia GTX660) and SSD: a 200× speedup over Chapter 3. The bottleneck is writing images to storage; image rendering itself takes only 3.6ms. This speed up is attributable to the difference in how Unity and Cycles render images. Unity's rasterizing renderer uses the full power of the GPU's dedicated rasterization hardware to draw triangular meshes. Cycles's path tracing renderer stochastically simulates the behaviour of millions of rays of light per-image – a much greater workload. Though Cycles has been carefully optimized to use GPU hardware as best as possible, a massive gulf in speed still exists between ray tracing and rasterization.

As well as saving the rendered images, UnityEyes outputs JSON-formatted metadata files. See Figure 5.3 for an example. These describe the scene of each image fully, including gaze direction, eye region shape parameters, and lighting information, as well as 2D facial landmarks in

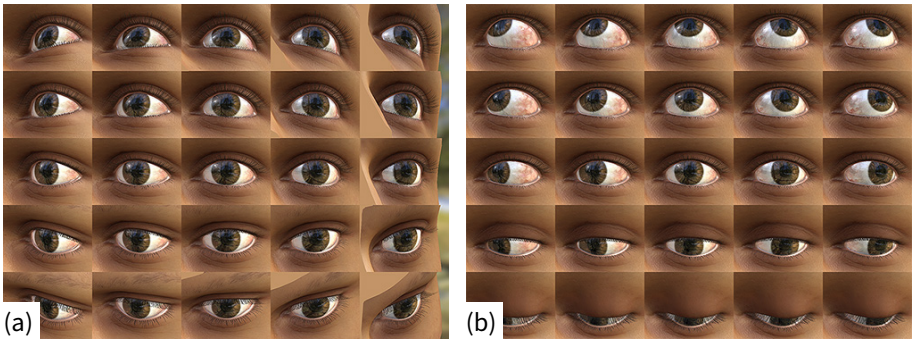


Figure 5.4: (a) Views of the same eye with the same gaze direction from 25 different camera positions. (b) Views of the same eye with varying gaze directions from a single camera position.

screen space (e.g. eye-corners and eye-centre).

5.3.2 Illuminating the models

A major source of error for appearance-based gaze estimation is illumination variation Zhang et al. (2015). The eye-region can appear very different depending on how it is illuminated – the brow or nose can produce cast shadows that obscure the eye, and un-even lighting can cause large variations in image intensity across the face. One of the main benefits of learning-by-synthesis is the fact that we can easily simulate these effects.

Spherical harmonics are orthogonal basis functions for representing functions defined over a sphere. They can be used to efficiently encode ambient diffuse illumination.

In Chapter 3, I used image-based lighting (IBL) to illuminate my collection of models with different environmental conditions. IBL in Unity can simulate ambient light via *spherical harmonics* (Ramamoorthi and Hanrahan, 2001) and can be used to generate reflections, but cannot be used to cast shadows. I therefore included a separate randomized directional light source that was used to introduce cast shadows and highlights around the eye region. This directional light simulates bright light sources like the sun.

For SynthesEyes, I picked from a collection of four panoramic HDR photographs when choosing environmental lighting. With UnityEyes I expanded this collection to 20 for increased variation, and randomly varied their rotation and exposure levels. Given an environment map and directional light, Unity's rendering engine then calculates surface shading, shadows, reflections, and ambient occlusion for the eye region and eyeball all in real-time.

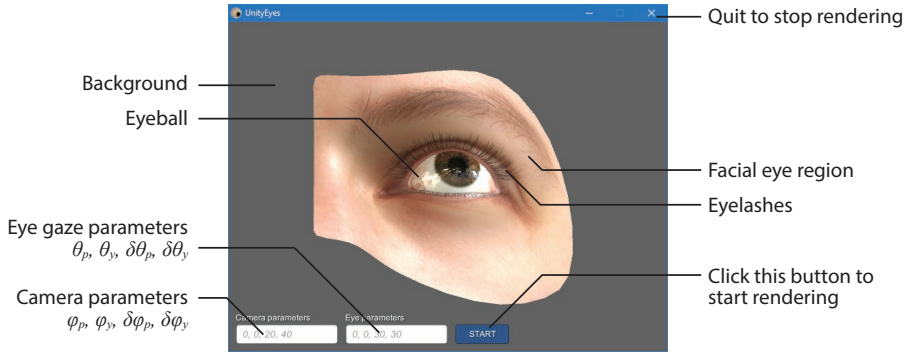


Figure 5.5: UnityEyes running on Windows 10 at 640×480px.

5.3.2 Posing the models

One of the advantages of generating a dataset using computer graphics is being able to precisely position objects in the scene without the practical difficulties of real-life image capture. Including images of the eye taken under different head poses is one approach for head pose independent gaze estimation. In Chapter 3, I iterated over a range of head poses and gaze directions in fixed discrete increments. This is not dissimilar to how real-life datasets were captured (Smith et al., 2013; Sugano et al., 2014). For UnityEyes, I instead specified the transforms of camera and eyeball randomly and continuously, allowing us to synthesize training data that densely covers the range of possible head poses. The generative eye region was positioned at the scene origin pointing forwards, and defines neutral head pose. For each rendered image, I randomly positioned the camera using spherical coordinates and pointed it towards the eyeball centre, simulating different head poses. I also randomly varied eyeball pitch and yaw as deviation from neutral gaze. Like Chapter 3, UnityEyes assumes an orthographic camera to simulate cropping a region of interest from a wide-angle image. Example renders with varying camera position and gaze are shown in Figure 5.4.

5.4 Using UnityEyes

I have made UnityEyes freely available for the benefit of the research community. This section briefly explains how to use the tool, though more information is available online[3]. On starting UnityEyes, users first choose a resolution for rendering images. The application then starts in *interactive mode* where users can pan around using the left

[3]UnityEyes – a tool for rendering eye images. `http://www.cl.cam.ac.uk/research/rainbow/projects/unityeyes/tutorial.html`

mouse button, rotate the eyeball with the middle mouse button, and adjust the zoom using the scroll wheel. As can be seen in Figure 5.5, the 3D eye region is rendered against a 50% grey background. Once the *start* button is pressed, the application will enter *rendering mode* where it will continuously randomize scene parameters, and save images and JSON metadata files. The parameters that are randomized include eye region shape, texture, pose, and environmental illuminaiton.

5.4.1 Targetting a scenario

In Chapter 3 I showed that that targetting a specific scenario with learning-by-synthesis can improve results. UnityEyes therefore allows users to specify a gaze distribution through parameters $\{\theta_p, \theta_y, \delta\theta_p, \delta\theta_y\}$ where eyeball pitch and yaw (in degrees) are modelled as uniform random variables $U(\theta_p - \delta\theta_p, \theta_p + \delta\theta_p)$ and $U(\theta_y - \delta\theta_y, \theta_y + \delta\theta_y)$. Changes in head pose are simulated by rotating the camera around the eye region. Variance in the camera position is definined in a similar way using parameters $\{\phi_p, \phi_y, \delta\phi_p, \delta\phi_y\}$. Scene parameters (including gaze) are varied randomly so it is not required to set a target number of rendering frames. Whether UnityEyes is run for one hour or one day, the resulting training data should still cover the desired distribution.

5.5 Experiments

Experiments were performed to assess both the quality of the images rendered by UnityEyes and their suitability for appearance based gaze estimation. In this section I first briefly describe the test datasets and the methodology used to match synthetic images to test data. I then present three experiments: one on matching eye images in the wild, one on gaze estimation with the MPIIGaze dataset, and one measuring the benefits of using a morphable shape model. Note that in these experiments I generated data with generic scene parameters, and did not perform any scenario-specific targeting as was done in Chapter 3.

Datasets The datasets from the 300-W challenge Sagonas et al. (2016) were used. These included **AFW** (Zhu and Ramanan, 2012), **IBUG** (Sagonas et al., 2013) and **LFPW+Helen** (Belhumeur et al., 2013; Le et al., 2012), containing 135, 337, 600, and 554 images respectively. The 300-W datasets feature uncontrolled images of faces *in the wild*: in indoor and outdoor environments, under varying illuminations, in presence of occlusions, under different poses, and from different quality cameras. These datasets were used to assess the re-

alism of the generated data only as they do not have gaze direction annotations.

To quantitatively evaluate UnityEyes for appearance based gaze estimation we used a subset of **MPIIGaze** (Zhang et al., 2015) that had manually annotated eye corners (1,500 eye images). The **SynthesEyes** dataset (12,000 synthesized images) as generated in Chapter 3 was used as a comparison against the new UnityEyes dataset.

Methodology In all of the experiments, eye images were cropped to 60×38px, and aligned using a similarity transform. For the 300-W and MPIIGaze datasets, the annotated eye corner locations included in the dataset were used for alignment. To align the UnityEyes images I used the same landmark conventions, with landmarks sampled from the 3D mesh. The estimated gaze vectors were rotated around the z-axis in-line with the similarity transform.

To match images, all images were first resized to a fixed ROI size with number of pixels P. They were then converted to floating-point grayscale, and normalised so each had zero mean and unit variance. Finally, image matching was performed using nearest-neighbour to choose the image i from the training set I_{train} that most closely matches the test image I_{test}.

$$i = \underset{i}{\operatorname{argmin}} \left(\frac{1}{P} \sum_{p=0}^{P} |I_{train_i}(p) - I_{test}(p)| \right)$$

The *pixel error* for image matching was computed using the mean absolute difference between the normalised images. The eye gaze error was computed as the angle between ground truth gaze direction and estimated gaze direction in degrees. The overall eye gaze error for an entire dataset was reported as the median of these gaze errors.

5.5.1 Matching eye images in the wild

In the first experiment, the realism of UnityEyes was tested by matching in the wild eye images to those generated by UnityEyes. Over a million images were rendered and matched these to the four in the wild test sets. The pixel-wise errors for UnityEyes ($\text{M} = 0.522, \text{SD} = 0.080$) is slightly higher than when matching from LFPW and Helen trainsets to the test sets ($\text{M} = 0.511, \text{SD} = 0.087$). A paired samples t-test reveals this difference is statistically significant; $t(1243) = 5.11, p < 0.001$. This suggests that there are still some differences between images in the wild, and synthetic images generated by UnityEyes. Paired

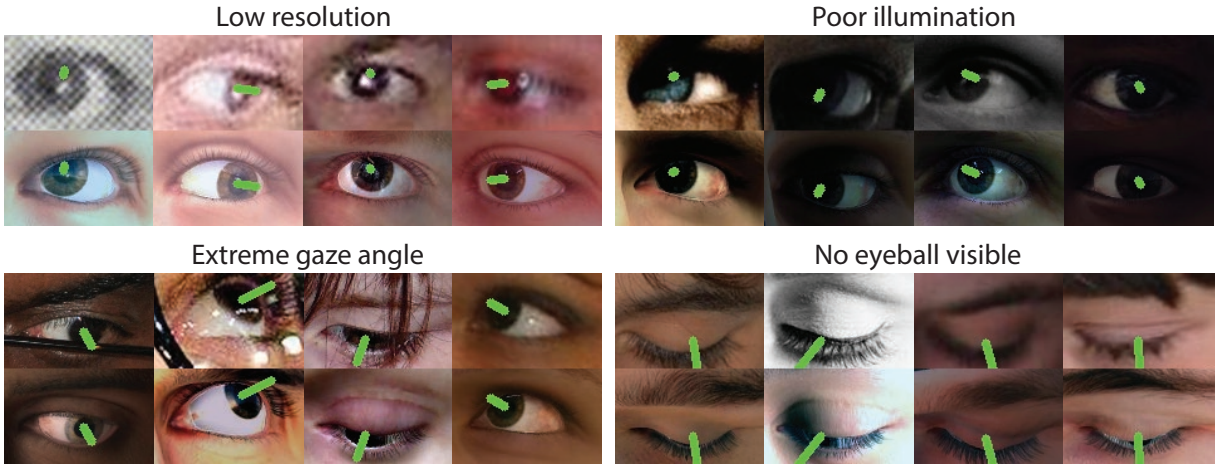


Figure 5.6: Nearest-neighbour pairs showing in-the-wild images from 300-W (top) and our renders (bottom) along with estimated gaze (green). Examples of successful fits on images with low resolution, poor illumination, extreme gaze angles, and occluded eyeballs are shown. No previous appearance-based gaze estimation system has attempted 3D gaze estimation for such extreme gaze angles before because they lacked ground truth data for training. Synthetic training data from UnityEyes has made this possible.

sampled t-tests showed that image matching using SynthesEyes is significantly less accurate ($\mathrm{M} = 0.607, \mathrm{SD} = 0.085$) when compared to UnityEyes; $t(1243) = -57.64, p < 0.001$; or the in the wild trainsets; $t(1243) = -46.96, p < 0.001$. This shows that UnityEyes images are closer to real-world captured images than SynthesEyes images.

Some example matches can be seen in Figure 5.6. The top rows show successful nearest-neighbour matches – these allowed gaze to be estimated for unseen people in unconstrained lighting conditions. Failure cases in the bottom row include un-modelled occlusions (e.g. hair) and appearance variation (make-up). These qualitative examples show the benefit of UnityEyes over previous methods, as it is now possible to estimate gaze for extreme gaze angles for which training data could not previously be collected. Indeed, in some cases the eye is not visible at all – a situation not addressed by previous work in gaze estimation. As nearest-neighbour image matching was carried out in a normalized grey-scale space, colours were corrected to aid visual comparison by matching the mean and standard deviations of the RGB channels.

Similarly for MPIIGaze, the UnityEyes generated dataset achieves better image matching performance ($\mathrm{M} = 0.467, \mathrm{SD} = 0.080$) than SynthesEyes ($\mathrm{M} = 0.510, \mathrm{SD} = 0.093$) and in the wild ($\mathrm{M} = 0.539, \mathrm{SD} = 0.086$) and even within dataset matching ($\mathrm{M} = 0.512, \mathrm{SD} = 0.095$). Paired sampled t-tests showed that UnityEyes was significantly better at matching MPIIGaze images compared to SynthesEyes; $t(1405) =$

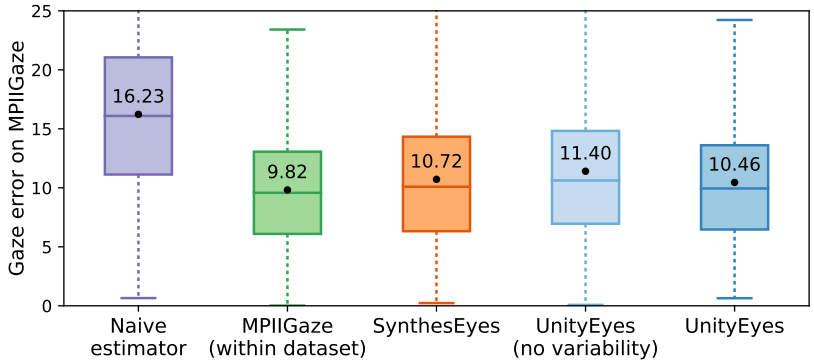


Figure 5.9: Box-and-whisker plots showing gaze errors (degrees) tested on MPIIGaze using a k-NN estimator. Boxes show the upper and lower quartiles, and whiskers show the full range of the gaze errors. Dots and text show mean values. X-axis represents training set. The final two box plots show the benefits of using the generative shape model (blue) over one with no variation (light blue).

Model	Gaze error
CNN with UT (Zhang et al., 2015)	13.91°
CNN with SynthesEyes (Wood et al., 2015)	13.55°
CNN with SynthesEyes+UT (Wood et al., 2015)	11.12°
k-NN with UnityEyes	**10.46°**

Table 5.1: A comparison between UnityEyes and previous work for cross-dataset gaze estimation on MPIIGaze. Previous work used convolutional neural networks trained with UT Gaze (Sugano et al., 2014) and SynthesEyes datasets. Note that UnityEyes achieves better results despite using a very simple learning approach.

$-12.67, p < 0.001$; in the wild trainsets; $t(1405) = -45.78, p < 0.001$; and within-dataset MPIIGaze images; $t(1405) = -12.91, p < 0.001$.

Training data amount analysis It was possible to generate over a million training images in less than 12 hours on commodity hardware. The effectiveness of this amount of data is shown in Figure 5.7 and Figure 5.8. It can be seen that both the image pixel error and gaze estimation errors decrease with an increased number of training images.

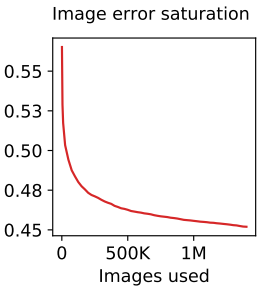


Figure 5.7: Image error for nearest-neighbour matching UnityEyes images against MPIIGaze.

5.5.2 Gaze estimation

In the second experiment, a k-Nearest-Neighbour gaze estimation system trained with UnityEyes was evaluated. For each real image in MPIIGaze, we found the top k matches from UnityEyes. The average of the k gaze directions from UnityEyes was used as the gaze estimate. Gaze error was calculated as the difference in degrees between this estimate and the ground truth. To compare with other datasets, we

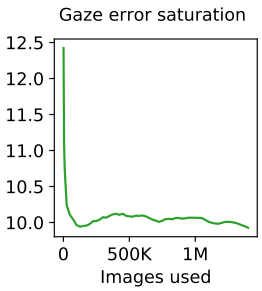


Figure 5.8: Gaze error for nearest-neighbour matching UnityEyes images against MPIIGaze. Lowest error achieved at 1.4M images.

also trained similar k-NN systems on held-out within-dataset images (MPIIGaze) and SynthesEyes images (Chapter 3). The error rates in degrees can be seen in Figure 5.9.

The results show that the UnityEyes dataset contains images that can be used to predict gaze with accuracy comparable to the SynthesEyes dataset. A paried samples t-test found no significant difference in the performance between UnityEyes and SynthesEyes data; $t(1405) = -1.51, p = 0.130$. However, UnityEyes images can be generated much faster than SynthesEyes images.

Furthermore, a simple k-Nearest-Neighbour approach achieves comparable performance to state-of-the-art deep learning based methods for cross-dataset appearance based eye gaze estimation without dataset targeting (Zhang et al., 2015; Wood et al., 2015) (see Table 5.1).

5.5.3 Shape variance

In the third experiment I measured the usefulness of our morphable eye region shape model $\mathcal{M}_{geo}$. Two different UnityEyes datasets were generated: a regular dataset using the full capabilities of the generative shape model, and a dataset with no shape variation, containing just the mean 3D shape $\boldsymbol{\mu}_{geo}$. Both datasets contained uniform variation in eye region texture.

The results demonstrate that using our morphable model shape variation was beneficial for both the pixel errors on the MPIIGaze dataset ($\text{M} = 0.467, \text{SD} = 0.080$ vs $\text{M} = 0.477, \text{SD} = 0.083$) and gaze estimation on the same dataset ($\text{M} = 10.455, \text{SD} = 5.408$ vs $\text{M} = 11.404, \text{SD} = 6.038$). According to paired samples t-tests, both image error and gaze error differences were statistically significant: $t(1405) = -3.35, p < 0.001$ for image error, and $t(1405) = -7.22, p < 0.001$ for gaze error. This is shown in Figure 5.9.

5.6 Summary

In this chapter I described UnityEyes – a research tool for rapidly synthesizing eye images for use as training data. It produces images with greater variation than before through the use of a generative shape model (Chapter 4), while also being much faster than previous work through the use of a real time game engine (Unity). In section 5.5 I described several experiments using datasets generated by UnityEyes. I showed that a massive dataset of synthetic eye images could be used to

estimate gaze for extremely challenging in the wild images from the 300-W datasets. These included images with extreme gaze directions and eyeball occlusions – situations that no previous appearance-based gaze estimation system has addressed. Through an evaluation on MPII, I demonstrated the importance of a good training set by out-performing previous cross-dataset deep learning systems using only a lightweight k-NN gaze estimation system trained with UnityEyes.

This chapter marks the end of my investigations into learning-by-synthesis. In Chapter 6 I will describe a new approach for person- and device-independent gaze estimation: analysis-by-synthesis.

6

Analysis-by-synthesis for gaze estimation

*This chapter includes work published at the **European Computer Vision Conference 2016** in Amsterdam, The Netherlands (Wood et al., 2016a). Tadas Baltrušaitis assisted with the gradient descent implementation.*

Look at the two faces in Figure 6.1. You would probably say that the faces appear to be looking in different directions. The face on the left appears to be gazing up, towards the right. The face on the right appears to be looking directly at you. What might be surprising is that *the eyes in each face are exactly the same*. If you don't believe this, try covering up the bottom halves of the faces. This suggests that we take information about the whole face into account when estimating gaze. If we want a computer system to estimate gaze like a human, we must allow it to use information outside the eyes alone.

Figure 6.1: The famous Wollaston (1824) illusion. Where is each face looking?

Despite this, appearance-based gaze estimation systems have traditionally only used the eye itself (Tan et al., 2002; Williams et al., 2006). As these basic systems do not consider the rest of the face, they break down under head movement. More recent work has attained better results by using an image of the entire face as an input. These methods become robust against variations in head pose and illumination by learning from large amounts of data (Zhang et al., 2015; Krafka et al., 2016). However, they still struggle to generalize beyond the scenario they were trained in. Instead of trying to *implicitly* learn a resilience against such modes of variation, can we model their effect on the facial eye region *explicitly*?

In this chapter, I revisit model-based gaze estimation, presenting the first approach that uses *analysis-by-synthesis*. Analysis-by-synthesis attempts to understand the world by rendering a generative model, and comparing the synthesized output to an observed image. The difference between synthetic and observed images is called the *reconstruction error*. If the reconstruction error is low, we can say that our internal model has explained the observed image well. Analysis-by-synthesis is an old technique that has made a come-back in recent years, being successfully applied to hand-tracking (Joseph Tan et al., 2016), body-tracking (Loper and Black, 2014), and face-tracking (Thies et al., 2015). In this chapter I will show how I applied analysis-by-synthesis to gaze estimation. The idea being that if we can encode all manner of eye region variation within an internal scene model, we should be able to build a generic gaze estimator.

I first present an overview of how analysis-by-synthesis for gaze estimation works. Following that, I describe the synthesis stage of my system, explaining how the scene is parameterized and rendered. Next I describe the analysis stage, explaining the formulation of the reconstruction energy and fitting strategy. Finally, I present experiments on the Columbia (Smith et al., 2013) and EYEDIAP (Funes Mora et al., 2014) gaze datasets, showing state-of-the-art performance in cross-dataset appearance-based gaze estimation.

6.1 Overview

There are two steps to an analysis-by-synthesis method: analysis and synthesis. The key idea is to "fit" a model to observed data. Given an observed image I_{obs}, we wish to produce a synthesized image I_{syn} that best matches it. Once I_{obs} and I_{syn} are matched, we call the final model parameters the "fit". The fitted eyeball parameters can then be used to

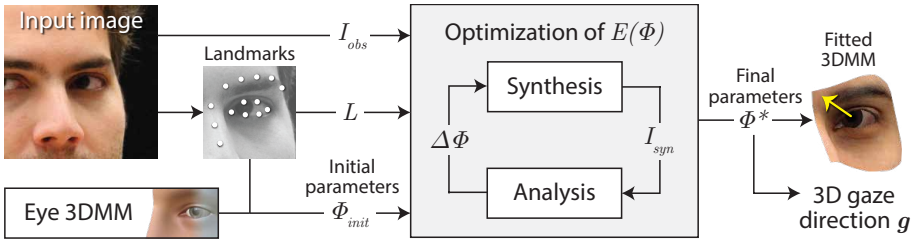


Figure 6.2: An overview the fitting process: First, facial landmarks L are localised and used to initialze our 3DMM (Chapter 4). Analysis-by-synthesis is then used to render an I_{syn} that best matches I_{obs}. Eye gaze $\boldsymbol{g}$ can then be extracted from fitted parameters $\boldsymbol{\Phi}^*$.

estimate gaze direction.

Synthesis (section 6.2) This stage generates synthetic images I_{syn} using the morphable model from Chapter 4 and a custom-written, highly optimized DirectX rendering framework. The full scene model is described by a set of parameters $\boldsymbol{\Phi}$ that cover both geometric (shape, texture, and pose) and photometric (illumination and camera projection) variation.

Analysis (section 6.3) This stage has two tasks: comparing how closely I_{syn} has matched I_{obs}, and adjusting model parameters $\boldsymbol{\Phi}$ to achieve a better fit. Observed and synthetic images are compared using an objective function $E(\boldsymbol{\Phi})$ which considers both a dense measure of appearance similarity, as well as a holistic measure of facial feature-point similarity (see Equation 6.7). $\boldsymbol{\Phi}$ is updated using gradient descent.

6.2 Synthesizing eye images

The goal is to use a 3D eye region model to synthesize an image which matches an input eye image. To render synthetic views of the eye region, I used the multi-part model described in Chapter 4. This model was posed in a scene, illuminated, and then rendered using a model of camera projection. Our total set of model and scene parameters $\boldsymbol{\Phi}$ are:

$$\boldsymbol{\Phi} = \{\boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\theta}, \boldsymbol{\iota}, \boldsymbol{\kappa}\}, \tag{6.1}$$

where $\boldsymbol{\beta}$ are the shape parameters, $\boldsymbol{\tau}$ the texture parameters, $\boldsymbol{\theta}$ the pose parameters, $\boldsymbol{\iota}$ the illumination parameters, and $\boldsymbol{\kappa}$ the camera parameters. In this section I describe each part of our model in turn, and the parameters that affect it.

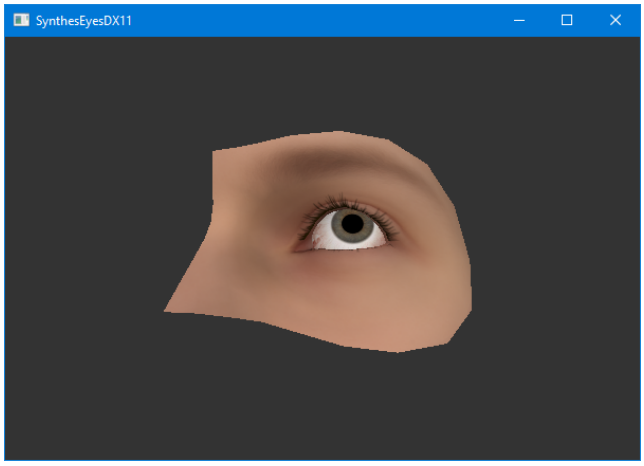


Figure 6.3: A screenshot of the DirectX version of the model.

6.2.1 Morphable facial eye region model

The first part of the model is a 3DMM of the eye region. This serves as a prior for facial appearance. The facial eye region is controlled with parameters $\boldsymbol{\beta}_{face} \subset \boldsymbol{\beta}$ and $\boldsymbol{\tau}_{face} \subset \boldsymbol{\tau}$. For the work in this chapter, I used generative models of both facial shape $\mathcal{M}_{geo}$ and texture $\mathcal{M}_{tex}$ as they were described in Chapter 4.

$$\mathcal{M}_{geo}\left(\boldsymbol{\beta}_{face}\right) = \hat{\boldsymbol{\mu}}_{geo} + \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma}_{geo})\,\boldsymbol{\beta}_{face} \tag{6.2}$$

$$\mathcal{M}_{tex}\left(\boldsymbol{\tau}_{face}\right) = \hat{\boldsymbol{\mu}}_{tex} + \mathbf{V}\,\mathrm{diag}(\boldsymbol{\sigma}_{tex})\,\boldsymbol{\tau}_{face} \tag{6.3}$$

$\hat{\boldsymbol{\mu}}_{geo}$ and $\hat{\boldsymbol{\mu}}_{tex}$ are the mean shape and texture. $\mathbf{U}$ and $\mathbf{V}$ are the orthonormal modes of shape and texture variation. $\boldsymbol{\sigma}_{geo}$ and $\boldsymbol{\sigma}_{tex}$ are the standard deviations of these modes of variation. Please see section 4.3 for the full derivation.

From our set of $c = 22$ scans, 90% of shape and texture variation can be encoded in 8 shape and 7 texture coefficients. This reduction in dimensionality is important for fitting our model efficiently.

Additionally, as eyelashes can provide a visual cue to gaze direction, they are modelled using a semi-transparent mesh controlled by a simple hair simulation (see subsection 4.6.3).

6.2.2 Parametric eyeball model

The second part of the multi-part model is the eyeball which is represented with a separate mesh. The eyeball model is almost exactly the same as the one described in Chapter 4. To vary iris texture, I use another linear model $\mathcal{M}_{iris}$,

$$\mathcal{M}_{iris}\left(\boldsymbol{\tau}_{iris}\right) = \hat{\boldsymbol{\mu}}_{iris} + \mathbf{W}\,\mathrm{diag}(\boldsymbol{\sigma}_{iris})\,\boldsymbol{\tau}_{iris} \tag{6.4}$$

where $\hat{\boldsymbol{\mu}}_{iris}$ is the mean iris texture, $\mathbf{W}$ are the modes of iris variation, $\boldsymbol{\sigma}_{iris}$ are the standard deviations of iris variation, and $\boldsymbol{\tau}_{iris} \subset \boldsymbol{\tau}$ are the iris texture parameters. Please see subsection 4.4.2 for more details.

As the "white" of the eye is not purely white, variations in sclera colour are modelled by multiplying the non-iris parts of the eyeball texture with a RGB tint colour parameter $\boldsymbol{\tau}_{tint} \in \mathbb{R}^3$.

Changes in iris size are modelled by scaling vertices on the iris boundary about the iris centre as specified by iris diameter β_{iris}.

6.2.3 Posing the model

Both global and local pose information is stored in parameter vector $\boldsymbol{\theta}$. The two parts of the model are defined in a local coordinate system with origin at eyeball centre, so I use the model-to-world transforms $\mathbf{M}_{face}$ and $\mathbf{M}_{eye}$ to position them in a scene. The facial eye region component has six degrees of freedom in translation $\boldsymbol{\theta}_T \in \mathbb{R}^3$ and rotation $\boldsymbol{\theta}_R \in \mathbb{R}^3$. These are encoded as 4×4 homogenous transformation matrices $\mathbf{T}$ and $\mathbf{R}$, so model-to-world transform $\mathbf{M}_{face} = \mathbf{TR}$. The eyeball's position is anchored to the face model, but it can rotate separately through local pitch and yaw transforms $\mathbf{R}_x(\theta_p)$ and $\mathbf{R}_y(\theta_y)$, giving $\mathbf{M}_{eye} = \mathbf{TR}_x\mathbf{R}_y$.

When the eye looks up or down, the eyelid moves to follow it. Eyelid motion is modelled using procedural animation and shrinkwrapping as described in subsection 4.5.2 and subsection 4.5.3.

6.2.4 Scene illumination

As we focus on a small, localized region of the face, I assume a simple illumination model where lighting is distant and surface materials are purely Lambertian. The illumination model therefore consists of an ambient light with colour $\mathbf{l}_{amb} \in \mathbb{R}^3$, and a directional light with colour $\mathbf{l}_{dir} \in \mathbb{R}^3$ and 3D direction vector $\mathbf{L}$. Colours are encoded as RGB triples, and the light direction is represented by pitch and yaw angles. This scene model does not consider specular effects, global illumination, or self-shadowing, so illumination depends only on surface normal and albedo. Radiant illumination $\mathcal{L}$ at a point on the model surface with normal $\mathbf{N}$ and albedo $\mathbf{c}$ is calculated as:

$$\mathcal{L}(\mathbf{n}, \mathbf{c}) = \mathbf{c}\,\mathbf{l}_{amb} + \mathbf{c}\,\mathbf{l}_{dir}\,(\mathbf{N} \cdot \mathbf{L}) \tag{6.5}$$

While this model is simple, I found it to be sufficient. If I were to consider a larger facial region, I would explore more advanced material or illumination models, as seen in previous work (Thies et al., 2015).

6.2.5 Camera projection

To analyse how images are formed, a model of camera projection is required. For simplicity, I fix an axis-aligned camera at the world origin, and set the world-to-view transform as the identity $\mathbf{I}_4$. I assume knowledge (or an estimate) of intrinsic camera calibration parameters $\boldsymbol{\kappa}$, and use these to build a projection transform matrix $\mathbf{P}$. The key parameters here are aspect ratio and field of view. These are not optimized over. A point in our model can then be transformed into image space using the model-view-projection transform $\mathbf{PM}_{\{face|eye\}}$.

6.2.6 DirectX rendering framework

To actually synthesize the images, I developed a new DirectX rendering engine from scratch. The goal was to be able to render and analyse images **as quickly as possible**. This is the reason I switched from Unity – though $\sim$300fps may be good for learning-by-synthesis, it is not fast enough for analysis-by-synthesis. Furthermore, interfacing between Unity and other software for optimization would be tricky and computationally expensive. I decided that a complete solution with DirectX and C++ was preferable.

As it typical of rendering engines, a major consideration is limiting the amount of CPU$\leftrightarrow$GPU communication. Through careful book-keeping, I ensure only a minimal set of GPU data is updated per-frame, based on what parameters in $\boldsymbol{\Phi}$ have changed. The most expensive operation in our fitting framework is transmitting I_{syn} from the GPU to CPU in order to analyse it. This is a bottleneck as it causes a GPU pipeline stall. This is alleviated by only copying only the valid foreground pixels P across during calculations. As a result, we can render I_{syn} at over 5000fps.

6.3 Analysis-by-synthesis for gaze estimation

Given an observed image I_{obs}, the aim is to produce a synthesized image $I_{syn}\left(\boldsymbol{\Phi}^*\right)$ that best matches it. 3D gaze direction $\boldsymbol{g}$ can then be extracted from eyeball pose parameters. I search for optimal model parameters $\boldsymbol{\Phi}^*$ using *analysis-by-synthesis*. To do this, I iteratively render a synthetic image $I_{syn}\left(\boldsymbol{\Phi}\right)$, compare it to I_{obs} using our energy function, and update $\boldsymbol{\Phi}$ accordingly. Let us cast this as an unconstrained energy minimization problem for unknown $\boldsymbol{\Phi}$.

$$\boldsymbol{\Phi}^* = \underset{\boldsymbol{\Phi}}{\operatorname{argmin}}\ E(\boldsymbol{\Phi}) \tag{6.6}$$

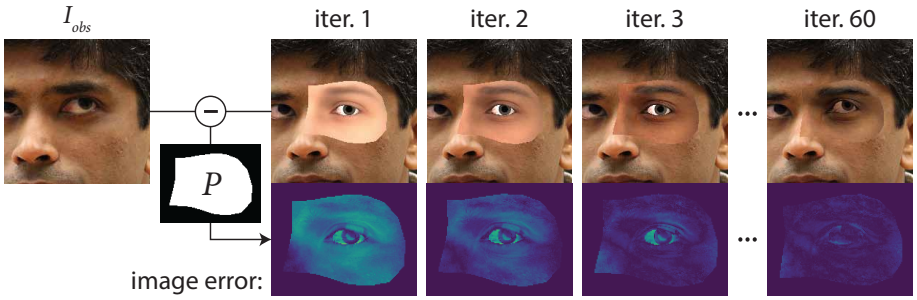


Figure 6.4: E_{image} measures dense image-similarity as the mean absolute error between I_{obs} and I_{syn}, over a mask of rendered foreground pixels P (white). Errors for background pixels (black) are ignored.

6.3.1 Objective function

The reconstruction energy is formulated as a combination of a dense *image similarity metric* E_{image} that minimizes difference in image appearance, and a sparse *landmark similarity metric* E_{ldmks} that regularizes the fit using reliable facial feature points, and weight λ controlling their relative importance.

$$E(\mathbf{\Phi}) = E_{image}(\mathbf{\Phi}) + \lambda \cdot E_{ldmks}(\mathbf{\Phi},\ L) \tag{6.7}$$

Image similarity metric The primary goal is to minimise the difference between I_{syn} and I_{obs}. This can be seen as an ideal energy function: if $I_{syn} = I_{obs}$, the model must have perfectly fit the data, so virtual and real eyeballs should be aligned. I approach this by including a dense photo-consistency term E_{image} in the energy function. However, as the 3DMM in I_{syn} does not cover the entire of I_{obs}, we must split the image into two regions: a set of rendered foreground pixels P that we compute error over, and a set of background pixels that we ignore (see Figure 6.4). Image similarity is then computed as the mean absolute difference between I_{syn} and I_{obs} for foreground pixels $p \in P$.

$$E_{image}(\mathbf{\Phi}) = \frac{1}{|P|} \sum_{p \in P} \left| I_{syn}(\mathbf{\Phi}, p) - I_{obs}(p) \right| \tag{6.8}$$

Landmark similarity metric The face contains important *landmark* feature points that can be localized reliably. These can be used to efficiently consider the appearance of the whole face, as well as the local appearance of the eye region. I use a state-of-the-art face tracker (Baltrusaitis et al., 2013) to localize 14 landmarks L around the eye region in image-space (see Figure 6.5). For each landmark $l \in L$ I compute a corresponding synthesized landmark l' from our 3DMM. The sparse landmark-similarity term is calculated as the distance between

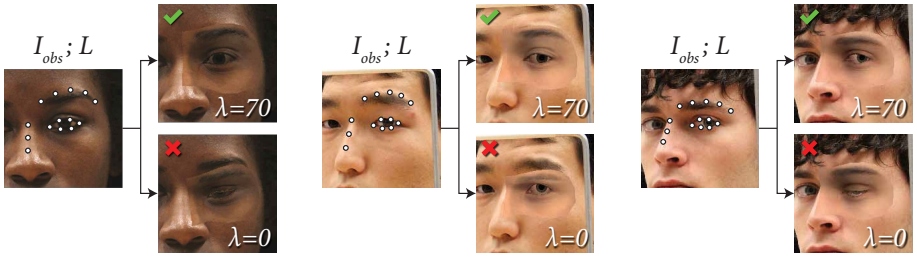


Figure 6.5: I_{obs} with landmarks L (white dots), and model fits with the landmark similarity term (top), and without (bottom). Note how it prevents erroneous drift in global pose, eye region shape, and local eyelid pose.

both sets of landmarks, normalized by the foreground area to avoid bias from image or eye region size. This acts as a regularizer to prevent fitted pose θ from drifting too far from a reliable estimate.

$$E_{ldmks}(\mathbf{\Phi},\ P) = \frac{1}{|P|}\sum_{i=0}^{|L|} \|l_i - l'_i\| \tag{6.9}$$

6.3.2 Optimization procedure

Fitting the model is a challenging non-convex, high-dimensional optimization problem. To approach it I use gradient descent (GD) with an annealing step size. Calculating analytic derivatives for a scene as complex as the eye region is challenging due to occlusions. I therefore use numeric central derivatives ∇E to guide our optimization procedure:

$$\mathbf{\Phi}_{i+1} = \mathbf{\Phi}_i - \boldsymbol{t} \cdot r^i \, \nabla E(\mathbf{\Phi}_i) \tag{6.10}$$

where $\mathbf{t} = [t_1 ... t_{|\mathbf{\Phi}|}]$ are per-parameter step-sizes, and r the annealing rate. The gradient $\nabla E(\mathbf{\Phi}_i)$is given by

$$\nabla E(\mathbf{\Phi}_i) = \left(\frac{\partial E}{\phi_1}, \frac{\partial E}{\phi_2}, \cdots, \frac{\partial E}{\phi_{|\mathbf{\Phi}|}} \right) \quad \text{and} \tag{6.11}$$

$$\frac{\partial E}{\phi_j} = \frac{E(\mathbf{\Phi}_i + h_j) - E(\mathbf{\Phi}_i - h_j)}{2h_j} \tag{6.12}$$

where $\mathbf{h} = [h_1 ... h_{|\mathbf{\Phi}|}]$ are per-parameter numerical values. $\mathbf{t}$ and $\mathbf{h}$ were calibrated by performing a simple grid-search experiment. Alternate optimization techniques were explored, including LBFGS (Liu and Nocedal, 1989) and rprop (Riedmiller and Braun, 1992). These were found to be unstable, perhaps due to the use of numerical rather than analytical derivatives.

Computing gradients ∇E is expensive, requiring rendering and differencing two images per parameter. Their efficient computation is made possible through the use of the tailored GPU DirectX rasterizer. Transmitting I_{syn} from GPU to CPU is a bottleneck as it causes a GPU pipeline stall. This was alleviated by copying only the valid foreground pixels P across during calculations. Figure 6.4 shows convergence for a typical input image, with I_{obs} size 800×533px, and I_{syn} size 125×87px. The optimization procedure converges after 60 iterations for 39 parameters, taking 3.69s on a typical PC (3.3Ghz CPU, GTX 660 GPU).

I explored foward-only differences as their use would have saved much computation. However, gradient estimates from forward-differences did not converge as well.

Initialization As the proposed method performs *local* optimization, an initial starting point is required. Of course, the closer the initial model configuration is to the global optimum, the better. I use 3D eye corner landmarks and head-pose rotation from the face tracker (Baltrusaitis et al., 2013) to initialize $\mathbf{T}$ and $\mathbf{R}$. I then use 2D iris landmarks and a simple single sphere eyeball model to initialize gaze direction (Baltrušaitis et al., 2016). $\boldsymbol{\beta}$ and $\boldsymbol{\tau}$ are initialized to $\mathbf{0}$, and $\boldsymbol{\iota}$ colors $\mathbf{l}_{amb}$ and $\mathbf{l}_{dir}$ are set to $[0.8, 0.8, 0.8]$.

6.3.3 Extracting gaze direction

Our task is estimating 3D gaze direction $\boldsymbol{g}$ in camera-space. Once our fitting procedure has converged, $\boldsymbol{g}$ can be extracted by applying the eyeball model transform to a vector pointing along the optical axis in model-space: $\boldsymbol{g} = \boldsymbol{M}_{eye}\,[0, 0, -1]^T$.

6.4 Experiments

In order to analyze the effectiveness of my approach at estimating eye gaze I evaluated it on two publicly available datasets: Columbia (Smith et al., 2013) and Eyediap (Funes Mora et al., 2014). These datasets were chosen as they show the full face, as required for our facial-landmark based initialization.

Columbia contains of images of 56 people looking at a target grid on the wall (see subsection 2.6.1). Example fits can be seen in Figure 6.6, right. These experiments used a subset of 34 people (excluding those with eyeglasses) with 20 images per person, resulting in 680 images. As the images were taken by a high quality camera (5184×3456px), they were downsampled to 800×533px for faster processing.

Eyediap contains 94 video sequences of 16 participants looking at two types of targets: *screen targets* and *floating target* (see subsection 2.6.2).

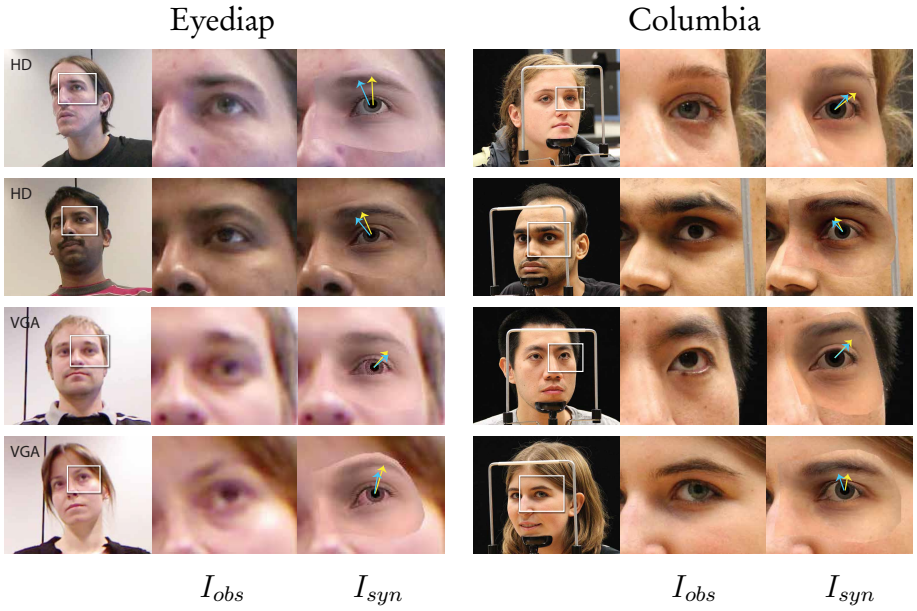


Figure 6.6: Example model fits on the Eyediap (Funes Mora et al., 2014) (HD and VGA) and Columbia (Smith et al., 2013) gaze datasets, showing estimated gaze (yellow) and labelled gaze (blue).

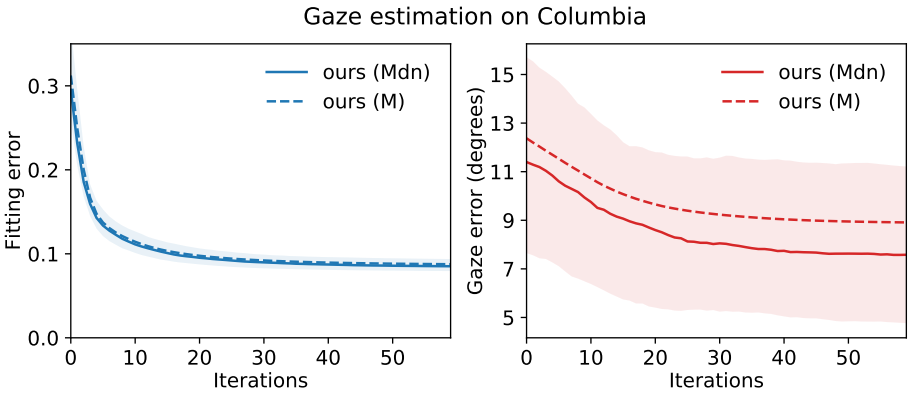


Figure 6.7: Fitting error (blue) and gaze estimation error (red) on the Columbia gaze dataset. Filled region shows inter-quartile range. Note both errors improve with the number of GD iterations.

Example fits can be seen in Figure 6.6, left. I extracted images from the VGA and HD videos for our experiment – 622 images with screen targets and 500 images with floating targets. In both cases, the system used a gradient descent step size of 0.0025 with an annealing rate of 0.95 that started after 10^{th} iteration.

6.4.1 Gaze estimation

In the first experiment I evaluated how well the model predicts 3D gaze direction for Columbia using the proposed analysis-by-synthesis approach. The results are shown in Figure 6.7, giving average gaze

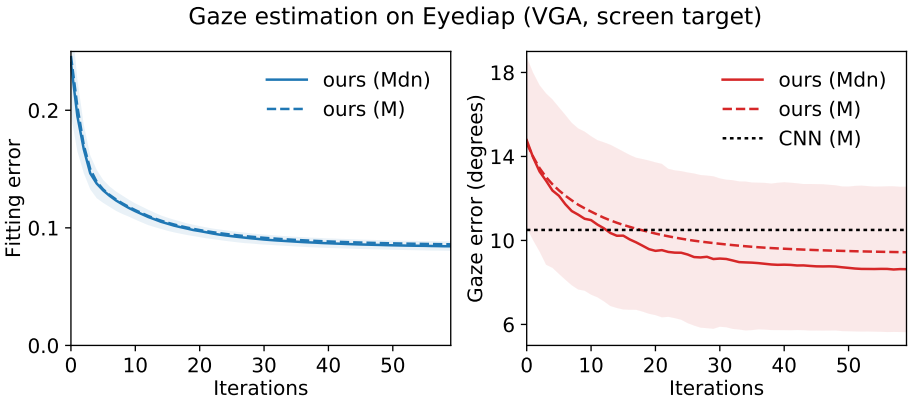


Figure 6.8: Fitting (blue) and gaze estimation (red) error on VGA videos from the Eyediap dataset. Screen target videos were used. We outperform a state-of-the-art CNN trained on UTGaze data (Zhang et al., 2015).

Model	Gaze error
CNN (UT) (Zhang et al., 2015)	10.5°
RF (UT) Sugano et al. (2014)	12.0°
k-NN (UT) (Zhang et al., 2015)	12.2°
ALR (UT) (Lu et al., 2014)	12.7°
SVR (UT) (Schneider et al., 2014)	15.1°
k-NN (UnityEyes) (Wood et al., 2016b)	19.9°
Analysis-by-synthesis (mine)	**9.44°**

Table 6.1: We outperform a range of state-of-the-art cross-dataset methods trained on UTGaze (UT) and synthetic data (UnityEyes), including Convolutional Neural Networks (CNN), Random Forests (RF), k-Nearest-Neighbour (kNN), Adaptive Linear Regression (ALR) and Support Vector Regression (SVR).

error of Mean (M) = 8.87°, Median (Mdn) = 7.54° after convergence. As the proposed method does not impose a prior on predicted gaze distribution, it can produce outliers with extreme error, so we believe its performance is best represented by a median (Mdn) average. Note how the decrease in fitting error corresponds to a monotonic decrease in mean and median gaze errors. Furthermore, the proposed approach outperforms the geometric approach used to initialize it (Baltrušaitis et al., 2016), a recently proposed k-Nearest-Neighbour approach (Wood et al., 2016b) (M = 19.9°, Mdn = 19.5°) and a naïve model that always predicts forwards gaze (M = 12.00°, Mdn = 11.17°).

The results for gaze estimation on Eyediap VGA images can be seen in Figure 6.8. As before the decrease in pixel error corresponds in the decrease in gaze errors. Furthermore, the final gaze estimation error on the Eyediap *screen* condition (M = 9.44°, Mdn = 8.63°) outperfoms that reported in literature previously – 10.5° using a Convolutional Neural

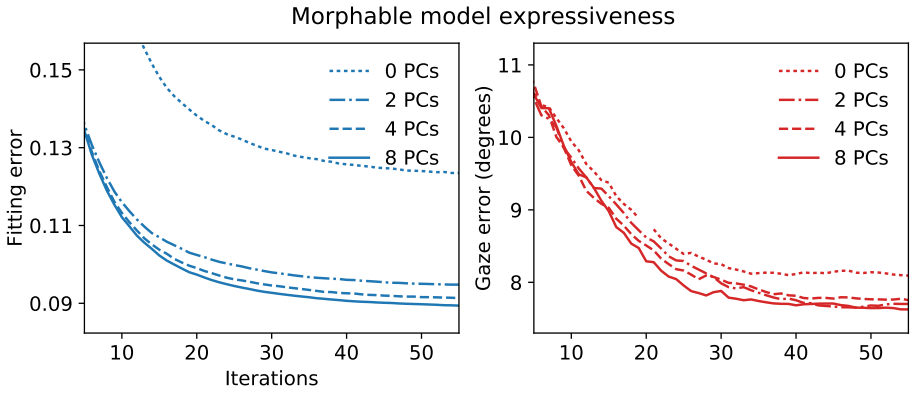


Figure 6.10: As we include more shape and texture and shape principal components (PCs) in the facial morphable model, we see decreases in both fitting and gaze error.

Network (Zhang et al., 2015) (see Table 6.1 for other comparisons). Analysis-by-synthesis also outperforms the model used to initialize the fitting, a kNN model ($\mathrm{M} = 21.49°, \mathrm{Mdn} = 20.93°$), and a naïve model ($\mathrm{M} = 12.62°, \mathrm{Mdn} = 12.79°$).

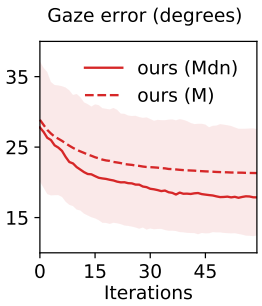


Figure 6.9: Analysis-by-synthesis can also generalize to the more challenging Eyediap floating target condition.

The results for Eyediap's floating targets (see Figure 6.9) are less accurate but still improve upon the baseline. Zhang et al. (2015) could not evaluate their approach for floating targets due to head pose variations not present in their training set. Despite a drop in accuracy, analysis-by-synthesis can still generalize to this difficult scenario and outperforms a kNN model ($\mathrm{M} = 30.85°, \mathrm{Mdn} = 28.92°$), and a naïve model ($\mathrm{M} = 31.4°, \mathrm{Mdn} = 31.37°$).

I performed a similar experiment for Eyediap images taken with the HD camera. These exhibit stronger head pose than the VGA images. The system achieved an average gaze error of $\mathrm{M} = 11.0°, \mathrm{Mdn} = 10.4°$ for screen targets and $\mathrm{M} = 22.2°, \mathrm{Mdn} = 19.0°$ for floating targets. Despite extreme head pose and gaze range, analysis-by-synthesis still performs comparably with the state-of-the-art and outperforms a kNN model ($\mathrm{M} = 29.39°, \mathrm{Mdn} = 28.62°$ for screen, and $\mathrm{M} = 34.6°, \mathrm{Mdn} = 33.19°$ for floating target), and a naïve model ($\mathrm{M} = 22.67°, \mathrm{Mdn} = 22.06°$ for screen, and $\mathrm{M} = 35.08°, \mathrm{Mdn} = 34.35°$ for floating target).

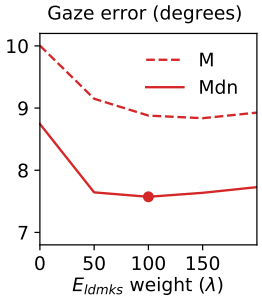


Figure 6.11: The effect of the landmark regularization term λ which decreases the error by not allowing the fit to drift.

6.4.2 Morphable model evaluation

In addition to evaluating the system's gaze estimation capabilities, experiments were performed to measure the expressive power of our morphable model and the effect of including E_{ldmks} in our objective function.

First, the importance of our facial point similarity weight (λ) was

assessed using the Columbia dataset. The same fitting strategy was used as before, but λ was varied. Results can be seen in Figure 6.11; the optimum choice has been highlighted. It is clear that λ has a positive impact on gaze estimation accuracy, by not allowing fits to drift too far from the reliable estimates and by reducing the variance of the error.

Second, I wanted to see if modelling more degrees of shape and appearance variation led to better image fitting and gaze estimation. I therefore varied the number of shape (β) and texture (τ) principal components (PCs) that our facial morphable model was allowed to use during fitting on Columbia, and recorded results for 0, 2, 4 and 8 PCs. Both the texture and shape PCs were varied together, picking the same number for both. As seen in Figure 6.10 (left), increasing the number of PCs lead to better image fitting error, as I_{syn} matches I_{obs} better when allowed more variation. A similar downward trend can be seen for gaze error, suggesting better modelling of nearby facial shape and texture is important for correctly aligning the eyeball model, and thus determining gaze direction.

6.5 Summary

In this chapter I presented a new method for visible light remote gaze estimation: analysis-by-synthesis. The idea is to fit an underlying scene model to an image, and take the fitted 3D eyeball orientation as the estimated gaze direction. This scene model includes the generative eye region model from Chapter 4, and photometric models of illumination and camera projection.

In section 6.2 I described the synthesis stage: how the scene model is parameterized by $\mathbf{\Phi}$ and how it's rendered using a tailored DirectX rasterizer. In section 6.3 I described the analysis stage: how the model is fit to an image by minimizing a reconstruction energy $E(\mathbf{\Phi})$ using numerical gradient descent.

By explicitly modelling how variations in camera type and head pose affect an image, analysis-by-synthesis can generalize to different usage scenarios in ways that traditional appearance-based gaze estimation cannot. In section 6.4 I showed that analysis-by-synthesis performed competitively with a state of the art deep learning cross-dataset approach (Zhang et al., 2015), while also being able to operate beyond the limited gaze range afforded by the deep learning system's training set. In Chapter 7 I will describe how I extended the system in this chapter to make it more performant using an improved fitting strategy, and more robust using a binocular eye region model.

7

Analysis-by-synthesis in real time

In the previous chapter, I presented a new approach for gaze estimation: analysis-by-synthesis. Though it showed promising results for person-independent and device-independent gaze estimation, it still lagged behind previous work with respect to processing speed. The system in Chapter 6 took $\sim$4 seconds to process an image. While this might be acceptable in some cases, many applications for gaze estimation require real time performance.

A further limitation of the work in Chapter 6 was the fact it only considered one eye at a time: the right eye. While the tracked facial landmarks helped regularize the fit in terms of head pose, the right eye might not always be visible. In these cases, the system in Chapter 6 will fail. A natural extension therefore is to track both eyes together. Since we know faces are roughly symmetrical, we can use information about one eye to help fit the other.

In this chapter I first describe the extensions made to the model in order to track both eyes at once. I then detail the second-order fitting strategy that was chosen to obtain better fits quicker: the Gauss-Newton algorithm. Finally, through experiments on both Columbia and Eyediap datasets, I demonstrate improved performance over Chapter 6 with respect to both gaze estimation accuracy and speed.

7.1 Binocular eye region tracking

The general analysis-by-synthesis approach remains the same as in Chapter 6: given an image frame I_{obs}, we wish to recover a set of

optimal parameters $\boldsymbol{\Phi}^*$ that best explains it in terms of our eye region model. We search for $\boldsymbol{\Phi}^*$ by iteratively rendering a synthetic eye region image I_{syn}, comparing it to I_{obs} using our reconstruction energy E (defined in Equation 7.2), and updating $\boldsymbol{\Phi}$ accordingly.

7.1.1 Binocular eye region model

This model is *binocular* as it considers two eyes rather than one.

In this chapter I use a binocular version of the model in Chapter 4. This model is used to render images for tracking the eye region with analysis-by-synthesis. It contains four main parts: the left and right facial eye regions, and the left and right eyeballs. It is specified by parameter vector $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \{\boldsymbol{\beta}, \boldsymbol{\tau}, \boldsymbol{\theta}, \boldsymbol{\iota}\}, \tag{7.1}$$

where $\boldsymbol{\beta}$ are the set of shape parameters, $\boldsymbol{\tau}$ the texture parameters, $\boldsymbol{\theta}$ the pose parameters, and $\boldsymbol{\iota}$ the illumination parameters. I now describe each parameter below.

Shape $\boldsymbol{\beta}$ The shape of each half of the binocular eye region is described by the same linear Principal Component Analysis (PCA) model $\mathcal{M}_{geo} \in \mathbb{R}^{3n}$ as in Chapter 6. We assume faces are symmetrical, so the shapes of both eye regions are controlled with a single set of coefficients $\boldsymbol{\beta}_{face} \in \mathbb{R}^{16}$, As before, an additional parameter β_{iris} controls iris size by scaling vertices on the iris boundary.

Texture $\boldsymbol{\tau}$ As before, I use a linear PCA texture model $\mathcal{M}_{tex} \in \mathbb{R}^{3m}$ of the facial eye region. It is controlled with texture coefficients $\boldsymbol{\tau}_{face} \in \mathbb{R}^8$. In this chapter, iris variation is modelled differently than before. A white "base" iris texture is used, and tinted with an RGB colour $\boldsymbol{\tau}_{iris}$. I decided to use this simpler RGB model over the PCA model as I found the detail of the PCA model wasn't necessary. Furthermore, the PCA model had trouble fitting to certain eye colours. Sclera color variation is modelled with another RGB parameter $\boldsymbol{\tau}_{tint}$

Pose $\boldsymbol{\theta}$ These describe both global and local pose. Globally, the two parts of the eye region are positioned with rotation $\boldsymbol{\theta}_{\mathbf{R}}$ and translation $\boldsymbol{\theta}_{\mathbf{T}}$. The interocular distance is controlled via θ_{iod} Eyeball positions are fixed in relation to the eye regions. The local pose parameters determine gaze. The general gaze direction is given by pitch and yaw angles θ_p and θ_y, and vergence is controlled with θ_v. When the eyeball looks up or down, the eyelids follow it. As in Chapter 6, eyelid motion is modelled using procedural animation, parameterized by θ_{lid}.

Illumination $\boldsymbol{\iota}$ I use the same illumination model as in Chapter 6: ambient light coupled with a single directional light. The ambient light

Parameter			#
Shape $\boldsymbol{\beta}$	$\boldsymbol{\beta}_{face}$	Eye region shape PCA coefficients	16
	β_{iris}	Iris size (mm)	1
Texture $\boldsymbol{\tau}$	$\boldsymbol{\tau}_{face}$	Eye region texture PCA coefficients	8
	$\boldsymbol{\tau}_{iris}$	Iris colour (RGB)	3
	$\boldsymbol{\tau}_{tint}$	Sclera tint colour (RGB)	3
Pose $\boldsymbol{\theta}$	$\boldsymbol{\theta}_{\mathbf{R}}$	Eye region rotation (radians)	3
	$\boldsymbol{\theta}_{\mathbf{T}}$	Eye region translation (mm)	3
	θ_{iod}	Interocular distance (mm)	1
	$\theta_p, \theta_y, \theta_v$	Eye gaze pitch, yaw, vergence (radians)	3
	θ_{lid}	Eyelid pitch (radians)	1
Illumination $\boldsymbol{\iota}$	$\boldsymbol{\iota}_{amb}$	Ambient light colour (RGB)	3
	$\boldsymbol{\iota}_{dir}$	Directional light colour (RGB)	3
	$\boldsymbol{\iota}_R$	Directional light rotation (pitch, yaw)	2
			50

Table 7.1: The total set of model parameters that are optimized over to reach a model fit.

has intensity $\boldsymbol{\iota}_{amb} \in \mathbb{R}^3$, and the directional light has intensity $\boldsymbol{\iota}_{dir} \in \mathbb{R}^3$ and direction defined by rotation $\boldsymbol{\iota}_R \in \mathbb{R}^2$ (pitch and yaw angles). We assume all surfaces are Lambertian.

In total we therefore have $17 + 14 + 11 + 8 = 50$ parameters of $\boldsymbol{\Phi}$ to optimize over. See Table 7.1 for the full list.

Rendering the model

Once our model has been configured with parameters $\boldsymbol{\Phi}$, we render synthetic images $I_{syn}(\Phi)$ using a DirectX-based rasterizer. We fix our virtual camera at the world origin, and assume knowledge (or estimate) of camera intrinsic parameters that define the camera projection.

These camera parameters are the same as $\boldsymbol{\kappa}$ in Chapter 6.

Realistically rendering eyes is a challenge. As before, I use physically correct corneal refraction techniques to better model the eyeball's layered transparent structure (see subsection 4.6.2). I implemented two additional optional effects to improve the realism of our output. First, for smoother skin, I implemented a single step of Loop (1987) subdivision with precomputed stencils for efficiency (see subsection 4.6.1). Second, I approximate ambient occlusion shadowing on the eyeball using a single-pass analytic technique: I project the positions of eyelid vertices into eyeball uv space, fit a 2D cubic polynomial to them, and apply per-pixel ambient occlusion as a function of distance to each eyelid polynomial. This lets us model the shadows around the interior eye margin using a cheap and simple single-pass solution.

These effects are optional as they slow rendering (and thus fitting) down. They can be used if speed is not a primary concern.

7.1.2 Energy formulation

A good energy function is critical to the success of any analysis-by-synthesis method. In this chapter, I modified the energy function of Chapter 6 to include additional priors. The proposed energy $E(\mathbf{\Phi})$ is a weighted sum of several terms, each encoding a different requirement of our model fit. Each term is expressible as a sum-of-squares, allowing me to minimize $E(\mathbf{\Phi})$ using the Gauss-Newton algorithm.

$$E(\mathbf{\Phi}) = \underbrace{E_{img}(\mathbf{\Phi}) + E_{ldmks}(\mathbf{\Phi})}_{\text{Data terms}} + \underbrace{E_{stats}(\mathbf{\Phi}) + E_{pose}(\mathbf{\Phi})}_{\text{Prior terms}} \tag{7.2}$$

The data terms E_{img} and E_{ldmks} (see Figure 7.2) guide our model fit using image pixels and facial landmarks, while the prior terms E_{stats} and E_{pose} penalize unlikely facial shape and texture, and eyeball orientations. I now describe each term in detail.

Image similarity $\boldsymbol{E_{img}}$ The primary goal is to minimize the photometric reconstruction error between I_{syn} and I_{obs}. The data term E_{img} expresses how well the fitted model explains I_{obs} by densely measuring pixel-wise differences across the images using a robust mean squared error. Image similarity is promoted with the term

$$E_{img}(\mathbf{\Phi}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \rho\big(\big| I_{syn}(p) - I_{obs}(p) \big| \big)^2 \tag{7.3}$$

where $\mathcal{P} \subset I_{syn}$ represents the set of rendered foreground pixels belonging to our 3D model. The background pixels are ignored. The robust function $\rho(e) = \min(\sqrt{T}, e)$, for threshold T, alleviates the effects of outliers; this is important for recovering iris colour in the presence of strong specular highlights on the eye.

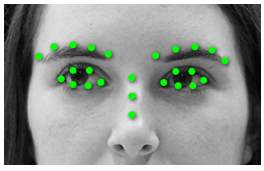


Figure 7.1: The 25 facial landmark points that are used in E_{ldmks}.

Landmark similarity $\boldsymbol{E_{ldmks}}$ The face contains several landmark feature points that can be tracked reliably. I therefore regularize the dense data term (E_{img}) using a sparse set of landmarks $\mathcal{L}$ provided by a state-of-the-art face tracker (Baltrušaitis et al., 2016). This is a common regularization technique for monocular facial capture methods (Thies et al., 2016). $\mathcal{L}$ consists of 25 points that describe the eyebrows, nose and eyelids. For each 2D tracked landmark $l \in \mathcal{L}$, I also compute a corresponding synthesized 2D landmark l' as a linear combination of projected vertices in our shape model. Facial landmark similarities are then incorporated into our energy using

$$E_{ldmks}(\mathbf{\Phi}) = \lambda_{ldmks} \cdot \frac{1}{|\mathcal{P}|} \sum_{i=0}^{|\mathcal{L}|} \|l_i - l'_i\|^2 \tag{7.4}$$

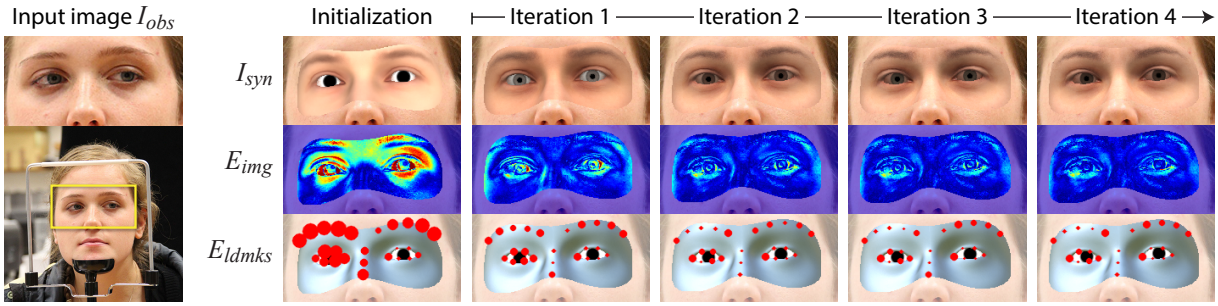


Figure 7.2: The 3D eye region model is fit to an image by minimizing a reconstruction energy $E(\Phi)$. The two main energy terms are a dense photometric error term E_{img} and a sparse landmark similarity term E_{ldmks}. This figure shows the energies decreasing over four iterations of the Gauss Newton algorithm – the false colour heat map shows pixel-wise differences decreasing, and the shrinking red circles shows landmark location differences decreasing.

As landmark distances $\|l_i - l'_i\|$ are measured in image-space, the energy is normalized by dividing through by foreground area $|\mathcal{P}|$ to avoid bias from eye region size in the image. The importance of E_{ldmks} is controlled with weight λ_{ldmks}.

Statistical prior E_{stats} Unlikely facial shapes and texture are penalized using a statistical prior Blanz and Vetter (1999). As we assume a normally distributed population, our PCA model parameters should be close to the mean $\mathbf{0}$. This prior is encoded in the term

$$E_{stats}(\mathbf{\Phi}) = \lambda_{geo} \cdot \sum_{i=0}^{|\boldsymbol{\beta}|} \beta_i^2 + \lambda_{tex} \cdot \sum_{i=0}^{|\boldsymbol{\tau}|} \tau_i^2 \tag{7.5}$$

Recall that $\beta_i \in \boldsymbol{\beta}$ and $\tau_i \in \boldsymbol{\tau}$ are scaled by their respective standard deviations in our model. This energy helps the model fit avoid degenerate facial shapes and texture, and guides its recovery from poor local minima found in previous frames. The penalties for unlikely shape and texture are weighted separately with λ_{geo} and λ_{tex}.

Pose prior E_{pose} The final energy penalizes mismatched parameters for eyeball gaze direction and eyelid position. The eyelids follow eye gaze, so if the eyeball is looking upwards, the eyelids should be rotated upwards, and visa versa. Pose consistency is enforced with

$$E_{pose}(\mathbf{\Phi}) = \lambda_{pose} \cdot \|\theta_{lid} - \theta_p\|^2 \tag{7.6}$$

where θ_{lid} is the eyelid pitch angle of our model's face parts, and θ_p is the gaze pitch angle of our eyeball parts. Its relative importance is controlled by weight λ_{pose}.

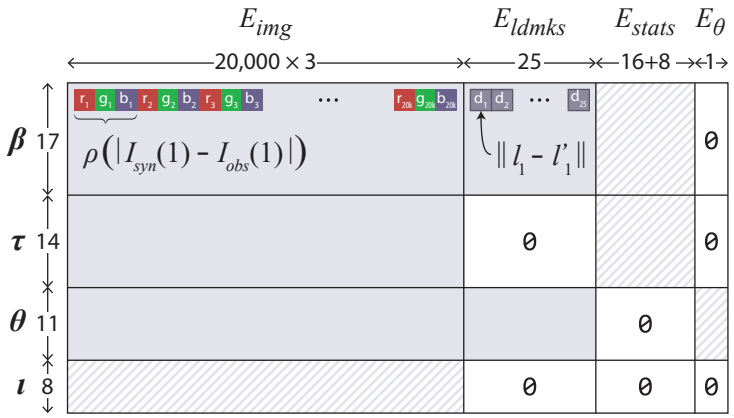


Figure 7.3: The non-zero structure of the Jacobian $\mathbf{J_r}$ for a 200×100px eye region. E_θ is E_{pose}. Dashed regions represent sparse blocks.

7.1.3 Optimization procedure

In Chapter 6 I optimized the reconstruction energy using gradient descent – a first order fitting method. In this chapter I use an annealed form of the Gauss-Newton algorithm – a second order optimization approach. The parameter update for $\mathbf{\Phi}$ is as follows:

$$\mathbf{\Phi}^{i+1} = \mathbf{\Phi}^i - \eta^i \left(\mathbf{J_r}^T \mathbf{J_r}\right)^{-1} \cdot \mathbf{J_r}^T \mathbf{r} \tag{7.7}$$

where $\mathbf{r}$ is the vector of energy function residuals, $\mathbf{J_r}$ the Jacobian matrix of residuals $\mathbf{r}$ evaluated at $\mathbf{\Phi}^i$, $\mathbf{J_r}^T \mathbf{J_r}$ the approximation to the Hessian matrix, and η the annealing rate. Figure 7.2 shows four iterations of our model fit.

To compute the Jacobian (see Figure 7.3) I use numerical central derivatives. This is an expensive operation, requiring two images to be rendered for every parameter. The system is made performant by calculating $\mathbf{J_r}$ and $\mathbf{J_r}^T \mathbf{J_r}$ entirely on the GPU, avoiding expensive pipeline stalls from cross-system data transfer. Since image rendering is a key operation for our system, the tailor-written DirectX rasterizer from Chapter 6 is critical. To further lighten the computational load of our numerical derivatives, it is possible to mask out a subset of $\mathbf{\Phi}$ when tracking in a video, so optimize over a smaller set of parameters frame-to-frame.

Initialization The energy landscape of $E(\mathbf{\Phi})$ is riddled with local minima, so its important to start from a good initialization. The face tracker provides 3D estimates for the facial landmark positions. Global translation parameters are initialized to the mean landmark position, and global rotation parameters are initialized using the the Kabsch (1976) algorithm, registering them to the 2D landmarks. Other parameters are initialized to $\mathbf{0}$ by default, except for interocular distance and

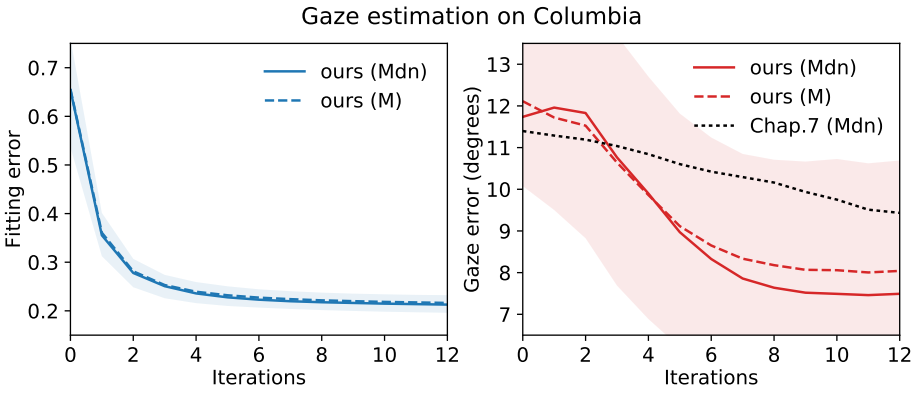


Figure 7.4: Fitting error and gaze error for the Columbia dataset decrease with the number of fitting iterations. The filled regions represent interquartile range. The second-order optimization strategy converges faster than gradient descent (dashed black line).

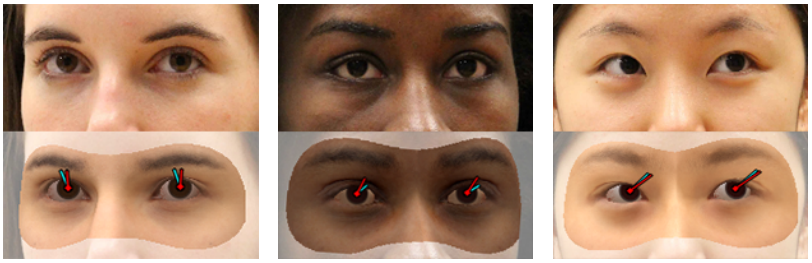


Figure 7.5: Example model fits on the Columbia gaze dataset (Smith et al., 2013) showing true gaze direction (red) and estimated gaze direction (cyan).

iris size, for which we use anthropomorphic averages, and illumination, for which we experimentally chose a basic setup. When tracking in video, we exploit temporal similarities by initializing Φ_{init} with Φ^* from the previous frame.

7.2 Experiments

In order to measure the benefits of the improved binocular eye region model and fitting procedure, I conducted gaze estimation experiments on images from the Columbia (Smith et al., 2013) and MPIIGaze (Zhang et al., 2015) datasets, and videos from the Eyediap (Funes Mora et al., 2014) dataset. To assess how suitable this system is for processing videos, I also measured runtime on Eyediap videos.

7.2.1 Gaze estimation on Columbia

A gaze estimation experiment was performed on the Columbia dataset. The same subset of images as used in subsection 6.4.1. Results are

shown in Figure 7.4, and example model fits can be seen in Figure 7.5. Photometric error and gaze estimation error decrease with the number of model fitting iterations. When run for 12 Gauss Newton iterations, the binocular analysis-by-synthesis system achieves a final error of $\text{M} = 8.03°, \text{Mdn} = 7.51°$, an improvement over the monocular system of Chapter 6 ($\text{M} = 8.87°, \text{Mdn} = 7.54°$).

Error in gaze pitch and yaw refer to errors in estimating the vertical and horizontal components of eye gaze respectively.

If we examine the pitch and yaw components of gaze separately, the proposed binocular system outperforms recent work (Jeni and Cohn, 2016) in terms of gaze yaw ($3.13°$ vs $3.51°$), though perform worse in terms of gaze pitch ($6.92°$ vs $4.27°$). This result is promising since the analysis-by-synthesis approach is dataset agnostic, while Jeni and Cohn's (2016) system was trained on the Columbia dataset specifically. Furthermore, if we compare the dashed black line in Figure 7.4 with the solid red line, we can see that the second-order optimization strategy leads to faster convergence than the first-order method used in Chapter 6, despite performing a similar amount of work per iteration.

7.2.2 Gaze estimation on Eyediap

In the previous chapter, I explored fitting the 3DMM to individual image frames extracted from Eyediap videos. This was because the fitting strategy was too slow to conveniently process videos. With the GPU-assisted Gauss Newton approach, fitting on videos is now possible. Gaze was estimated for the first 500 frames of each of the 50 videos. Example frames from three videos can be seen in Figure 7.6.

For each frame, a single true gaze vector was obtained by averaging the two look vectors for each eye. A corresponding estimated gaze vector was found by averaging the 3D look vectors of the left and right eyeball in the eye region model. The results can be seen in Figure 7.7. It can be seen that even after only five Gauss Newton (GN) steps with the binocular model ($\text{M} = 9.03°, \text{Mdn} = 8.29°$), we reach a lower error than with 60 Gradient Descent (GD) steps with the monocular one ($\text{M} = 9.44°, \text{Mdn} = 8.63°$). If we allow our approach to run for ten iterations per frame, error decreases further to $\text{M} = 8.13°, \text{Mdn} = 7.12°$.

When compared to recent appearance-based approaches, this result is extremely promising. The right plot in Figure 7.7 shows three deep learning approaches as presented by Zhang et al. (2016). These deep learning approaches were all trained on images from Eyediap itself – they are not dataset independent. My binocular analysis-by-synthesis approach achieves competitive results, though it operates in a completely dataset independent manner.

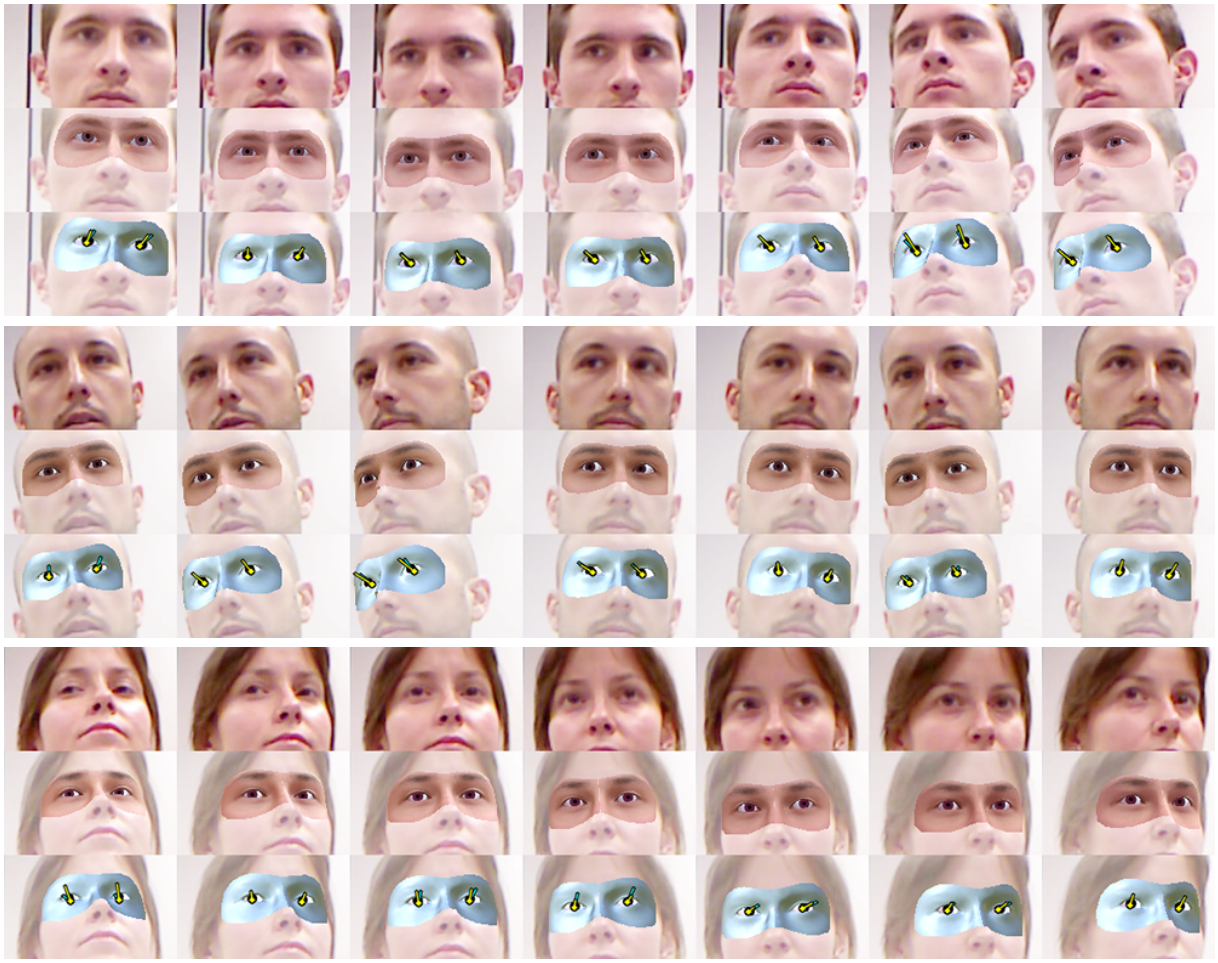


Figure 7.6: Improvements in speed over Chapter 6 make the method in this chapter is suitable for gaze estimation in videos. Here are some example frames from three Eyediap videos featuring head movement. In each row, top row is input image, middle row is overlaid model fit, and bottom row shows model shape with gaze. Yellow lines are estimated gaze, cyan is ground truth gaze.

7.2.3 Gaze estimation on MPIIGaze

A final gaze estimation experiment was performed on the challenging MPIIGaze dataset. From a subset of 22500 images, the face tracker (Baltrušaitis et al., 2016) managed to find a face in 17993 images. These images were used as input. To better compare against previous work, all participants were used, including those who wore glasses. Example model fits can be seen in Figure 7.8. As with Eyediap, a single true gaze vector and a single estimated gaze vector were calculated for each image by averaging the gazes of each eye.

When used with 5 GN iterations, an average gaze error of $\mathrm{M} = 14.18°, \mathrm{Mdn} = 13.56°$ was achieved. If allowed 10 GN iterations, this gaze error improves to $\mathrm{M} = 10.71°, \mathrm{Mdn} = 10.01°$. This is slightly worse than the cross-dataset appearance-based system shown in Chap-

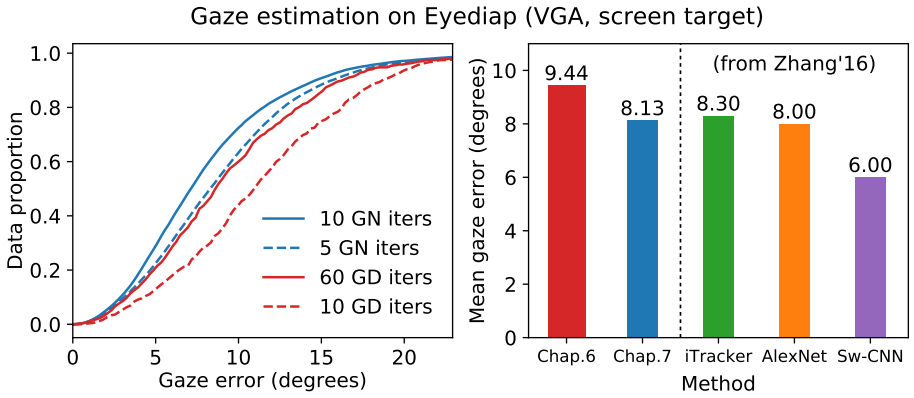


Figure 7.7: For Eyediap, the new fitting strategy achieves better results in just five Gauss Newton iterations (red) compared to 60 Gradient Descent iterations (blue). Furthermore, a mean gaze error of $8.13°$ compares well to recent Eyediap-specific deep learning approaches (green, orange, purple).

Figure 7.8: True gaze is shown in cyan, and estimated gaze in yellow. The top row of images shows successful fits from the top 10% of results. The bottom row shows failure cases from the bottom 10%. As can be seen, the fitting approach is prone to failure under challenging illumination conditions.

ter 5 ($\mathrm{M} = 10.46°$), and a long way from the $\mathrm{M} = 4.8°$ that can be achieved with dataset-specific approaches (Zhang et al., 2016). However, I believe this result is promising since analysis-by-synthesis system is generic. Furthermore, we can dissect its failure cases in ways not possible with black-box appearance-based approaches. For example, by examining the bottom row of Figure 7.8 we can see that shadows from the brow ridge and unusual upwards facing lighting conditions are present in the failure cases. This could guide possible extensions to the model to help address these limitations.

7.2.4 Runtime

To determine if the new fitting strategy is suitable for real time use, I measured runtime performance on Eyediap videos. A typical desktop

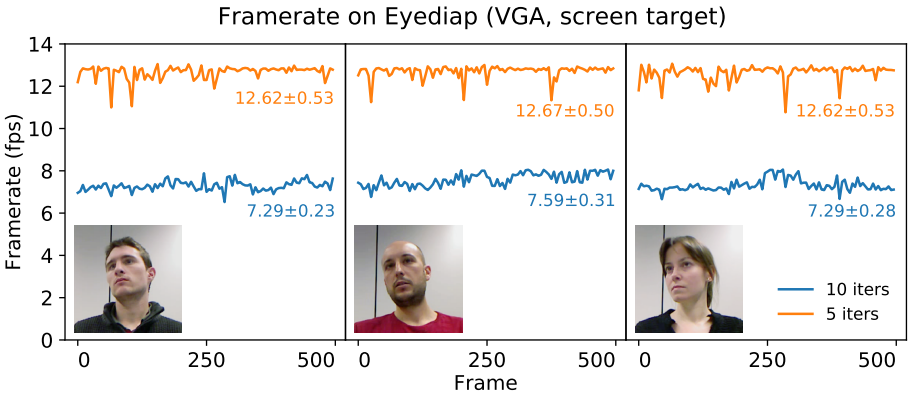


Figure 7.9: FPS over three Eyediap videos. With ten Gauss Newton iterations, interactive frame rates are possible. When limited to five iterations, the systems approaches real time performance.

PC was used (3.3Ghz CPU, GTX1080 GPU). Plots of the equivalent frames-per-second (fps) throughout three different VGA Eyediap videos are shown in Figure 7.9.

If allowed ten GN iterations per frame, an average framerate of 7.40fps was achieved. If a little gaze accuracy is sacrificed, and only five GN iterations used, an average framerate of 12.63 fps is possible. This is a vast improvement over the 0.2fps achieved by the previous system.

7.3 Summary

In this chapter I presented two extensions to the analysis-by-synthesis approach from Chapter 6. First, I extended the eye region model to track both eyes at once (subsection 7.1.1). This is important for robust tracking during head motion where one eye may not be clearly visible. Second, I replaced the first-order optimization method from Chapter 6 with a GPU-assisted second-order approach (subsection 7.1.3). This was key in improving runtime performance.

Through experiments on the Columbia and Eyediap datasets, I demonstrated improved results with respect to both runtime and accuracy compared to the first-order monocular method of Chapter 6. I also demonstrated results competitive with state of the art dataset-specific deep learning systems on the Eyediap dataset (Zhang et al., 2016). Through an experiment on the MPIIGaze dataset, I showed that analysis-by-synthesis can achieve results comparable to those in Chapter 5, but there is still a wide performance gap between my cross-dataset system and other dataset-specific approaches.

8

Conclusion

My main goal was to help bring eye tracking out of the lab and into the real world. To do this, I addressed two major problems for visible light remote gaze estimation: 1) **It is hard to collect good training data**. Since it is impossible for Mechanical Turks to label eye gaze accurately, researchers must collect images of people looking in specified directions themselves – a time consuming and expensive process. I have shown that learning-by-synthesis is a promising alternative, allowing us to synthesize eye tracking training data in a fraction of the time and at a fraction of the cost. 2) Appearance-based eye trackers **struggle to generalize beyond the scenario they were trained in**: e.g. a system trained with data collected from laptop webcams will fail when deployed on a mobile phone. I have revisited model-based gaze estimation, and shown that analysis-by-synthesis can estimate gaze in a device-independent manner.

8.1 Contributions

The main contributions of my work are as follows:

Learning-by-synthesis In Chapter 3 I explored learning-by-synthesis as an alternative to traditional manual data collection. Though I am not the first to apply learning-by-synthesis to gaze estimation (Sugano et al., 2014), I am the first to demonstrate that cheap 3D head scans and modern computer graphics techniques can be used to train eye tracking systems without any manual data collection at all.

Eye region morphable model In Chapter 4 I presented a new multi-part 3D morphable model of the eye region. Previous 3DMMs grossly

simplified the eyes: they either joined the eyeballs to the face, or cut them out entirely. Instead, I correctly modeled the face and eyes as separate parts that move independently. In Chapter 5 I used this model to take learning-by-synthesis a step further, showing how a massive dataset of synthetic images could be used to estimate gaze despite extreme gaze directions and eyeball occlusions. I have made this 3D model available online and it has since been used by researchers both in academia (Sugano et al., 2016) and industry (Shrivastava et al., 2016).

Analysis-by-synthesis In Chapter 6 I proposed a new approach for gaze estimation: analysis-by-synthesis. The general idea is to fit the eye region 3DMM to an input image using energy minimization, and take the fitted 3D eyeball orientation as gaze direction. By explicitly modelling variation in eye region appearance, pose, and camera type in an underlying scene model, it is possible to estimate gaze in a person- and device- independent way. In Chapter 7 I showed how such a system can be made practical through careful engineering (GPU compute) and a well-chosen optimization strategy (Gauss-Newton).

8.2 Limitations

Have I addressed several important issues faced by visible light remote gaze estimation? Yes. Are the systems described in this dissertation a feasible replacement for traditional infrared-based eye trackers? Not yet. Limitations remain.

If we could generate the "perfect" synthetic dataset, systems trained using synthetic data would out-perform those trained with real data. This is not the case for gaze estimation: a within-dataset trained CNN performs better on the same dataset than one trained using synthetic data. Clearly, though synthetic training data for eye tracking is useful, it has not yet rendered traditional means of data collection unnecessary. Problems may stem from the 3D model itself. For example, the eyelid motion model may be unrealistic since it was built by hand using photo-references only. There are also issues with how synthetic data is generated. The fact that synthetic images are "perfect" is a problem – rendered images do not exhibit sensor noise or motion blur like the real world. These phenomenon could be modelled with more advanced graphics, but this would come at an additional cost to render time.

Though the morphable model in Chapter 4 encodes variation in eye region shape, texture, and pose, it fails to capture other important modes of variation. The facial texture model does not consider makeup –

eyeliner and eye shadow can have a large effect on eyelid darkness, and mascara can thicken eyelashes. Furthermore, occlusions from overhanging hair or eyeglasses are not modelled. Makeup, hair, and eyeglasses could be included as extensions to the model, though it is hard to say if the increase in model complexity would be worthwhile.

In Chapter 7 I showed that analysis-by-synthesis for gaze estimation could be performed efficiently using a typical desktop PC, However, this approach is not yet suitable for lower-powered devices like mobile phones or tablets. This is because the model fitting strategy, and the calculation of numerical gradients in particular, depends heavily on the GPU. Previous work with analysis-by-synthesis for face tracking (Thies et al., 2015) and hand tracking (de La Gorce et al., 2011) has shown that it's possible to calculate analytic derivatives for similar reconstruction energies. This could drastically reduce the computational workload required to fit the eye region model to an image.

Previous work that calculated analytic derivatives used single-part models only.

A further limitation of my analysis-by-synthesis approach is its failure under challenging illumination conditions like those found in MPIIGaze. In theory, analysis-by-synthesis should be able to recover scene illumination accurately enough to fit the eyeballs, but in practice I found it frequently failed to do so. This may be a result of the simple illumination model that does not consider cast shadows or multiple light sources. Or it might be because the energy function favours holistic image similarity over small details, so prefers to match broad illumination effects at the expense of the eyeball. Either way, there is still a large gap in gaze estimation accuracy between dataset-specific deep-learning approaches and my proposed generic analysis-by-synthesis system for challenging scenarios like MPIIGaze.

8.3 Future work

Despite the limitations described above, I believe my work has demonstrated promising results, and opened avenues for future research.

8.3.1 What level of realism is required?

In Chapter 3, I performed an experiment to measure the benefits of different parts of a synthetic data generation pipeline: variation in illumination and eyelid motion. There is much more work still to be done to determine what graphics features are important when it comes to synthesizing training data. In terms of realism, how real do you have to go (Figure 8.1)? When designing the graphics side of a learning-by-

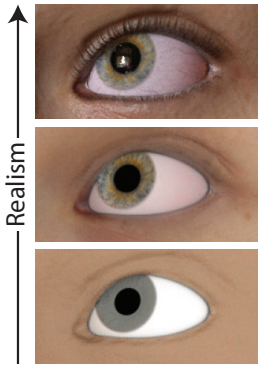


Figure 8.1: How real do you need to go for successful learning- or analysis-by-synthesis?

synthesis approach, it is very tempting to go "all the way", and try to model real life in as much detail as possible. However, different graphics techniques have different computational costs. If we could confirm that an expensive rendering technique (e.g. subsurface scattering for skin) did not help improve gaze estimation error, we could remove it from the synthesis pipeline and save a lot of computational effort. Future work could examine the benefits of different rendering techniques to find a sweet-spot between slow-but-realistic and fast-but-unrealistic when it comes to learning- and analysis-by-synthesis.

For example, (Shrivastava et al., 2016), claim that images rendered with UnityEyes lack realism in certain ways so are sub-optimal for learning-by-synthesis. To address this, they upgrade UnityEyes images with neural networks to improve their realism. This lets them avoid having to discover what types of realism are missing from UnityEyes. However, another approach would be to explicitly model the parts of the image formation process that are missing, thus resulting in this lack of realism. If there is a difference in image noise between UnityEyes images and real images, this could be efficiently applied in post-processing. If we can discover that UnityEyes models lack certain types of eye wrinkles around the eyelids, this could be included as part of the eye region model. Though this may not always be possible, if we can make informed decisions about what parts of the image formation process are really important for learning-by-synthesis, it should be possible to build a "perfect" synthetic dataset for training.

8.3.2 Hybrid approaches

In this dissertation I presented learning-by-synthesis and analysis-by-synthesis separately. It is actually important to combine them. Model-fitting methods that use analysis-by-synthesis must start from some initial configuration. Of course, the closer the initialization is to the optimum, the better. Therefore, a common tactic is to combine the benefits of a bottom-up generative approach with a top-down discriminative one: use a machine-learning *initializer* to choose initial parameters, and then fit your model from there (Taylor et al., 2016). While I used facial landmarks to initialize model pose, future work could use a more sophisticated learning-by-synthesis system to initialize facial shape, texture, environmental illumination, and indeed eye gaze too. Such a system could be learned from synthetic training data. This could help us avoid poor model fits, and might save time by cutting down on the number of required fitting iterations.

8.3.3 Tracking the eyes and face together

Previous work in facial capture has used specialized subsystems to track the eyes separately from the rest of the face (Wang et al., 2016). In my opinion, this is an admission that the face model is not good enough. I have shown that, given a good model, it is possible to fit both the face and the eyes to an image *simultaneously*. If you go down the "track each part of the face with a different system" route, where do you stop? You start with an eye-specific sub-system for tracking eyes (Wang et al., 2016; Bermano et al., 2015). You then might add a teeth-specific sub-system (Wu et al., 2016). What next? An ear tracker? A tongue tracker? Rather than fitting a collection of models in turn, future work should explore *jointly optimizing a single comprehensive model*. Though each facial part may be parameterized separately, they can still be fit together jointly. Though building such a model would be challenging, I think the benefits would be worthwhile, particularly for tracking facial details at the boundaries of different parts.

Indeed, work towards tracking the face with a unified model has already begun. Li et al. (2017) used my eye region model to augment their FLAME face model (Figure 8.2) so they could track the eyes and face together. They found that tracking the eyeballs jointly with the face improves alignment, in particular for the eye lids.

Figure 8.2: The FLAME face model includes my eyeball model for improved face tracking.

8.4 Final remarks

The field of gaze estimation has advanced rapidly over the last few years. This is a result of recent technological breakthroughs like deep learning that have shaken computer vision as a whole, and a wave of recently collected gaze datasets on which we can develop and test our algorithms. My work has been a part of this. I set out to address the problem of training data collection for gaze estimation, and showed that learning-by-synthesis was a viable alternative. I also set out to build a generic gaze estimator, and showed how a person- and device-independent analysis-by-synthesis system could estimate gaze almost as accurately as a data-set specific deep learning system.

While we are still some way from being able to track gaze reliably and accurately in the wild, I believe my work in this dissertation has taken several important steps towards this goal.

Bibliography

Aldrian, O. and W. A. Smith
2013. Inverse rendering of faces with a 3d morphable model. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1080–1093.

Asthana, A., S. Zafeiriou, S. Cheng, and M. Pantic
2013. Robust discriminative response map fitting with constrained local models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 3444–3451.

Bäck, D.
2006. Neural network gaze tracking using web camera. *Technical Report, Institutionen för medicinsk teknik*.

Baltrušaitis, T., P. Robinson, and L.-P. Morency
2012. 3d constrained local model for rigid and non-rigid facial tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, Pp. 2610–2617. IEEE.

Baltrusaitis, T., P. Robinson, and L.-P. Morency
2013. Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Pp. 354–361.

Baltrušaitis, T., P. Robinson, and L.-P. Morency
2016. Openface: an open source facial behavior analysis toolkit. In *IEEE Winter Conference on Applications of Computer Vision*.

Belhumeur, P. N., D. W. Jacobs, D. J. Kriegman, and N. Kumar
2013. Localizing parts of faces using a consensus of exemplars. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2930–2940.

Bérard, P., D. Bradley, M. Nitti, T. Beeler, and M. Gross
2014. Highquality capture of eyes. *ACM Transactions on Graphics*.

Bermano, A., T. Beeler, Y. Kozlov, D. Bradley, B. Bickel, and M. Gross
2015. Detailed spatio-temporal reconstruction of eyelids. *ACM Transactions on Graphics (TOG)*, 34(4):44.

Blanz, V. and T. Vetter
1999. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, Pp. 187–194. ACM Press/Addison-Wesley Publishing Co.

Booth, J., A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway
2016. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 5543–5552.

Brown, M., M. Marmor, E. Zrenner, M. Brigell, M. Bach, et al.
2006. Iscev standard for clinical electro-oculography (eog) 2006. *Documenta ophthalmologica*, 113(3):205–212.

Bulling, A., J. A. Ward, H. Gellersen, and G. Tröster
2009. Eye movement analysis for activity recognition. In *Proceedings of the 11th international conference on Ubiquitous computing*, Pp. 41–50. ACM.

Buswell, G. T.
1935. *How people look at pictures*. University of Chicago Press Chicago.

Canny, J.
1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Cao, C., Q. Hou, and K. Zhou
2014a. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on Graphics (TOG)*, 33(4):43.

Cao, C., Y. Weng, S. Zhou, Y. Tong, and K. Zhou
2014b. Facewarehouse: A 3d facial expression database for visual computing. *Visualization and Computer Graphics, IEEE Transactions on*, 20(3):413–425.

Catmull, E. and J. Clark
1978. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-aided design*, 10(6):350–355.

Cootes, T. F., C. J. Taylor, et al.
2001. Statistical models of appearance for medical image analysis and computer vision. In *Proc. SPIE medical imaging*, volume 4322, P. 5.

Cornsweet, T. N. and H. D. Crane
1973. Accurate two-dimensional eye tracker using first and fourth purkinje images. *JOSA*, 63(8):921–928.

de La Gorce, M., D. J. Fleet, and N. Paragios
2011. Model-based 3d hand pose estimation from monocular video. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1793–1805.

Debeljak, M., J. Ocepek, and A. Zupan
2012. Eye controlled human computer interaction for severely motor disabled children. In *International Conference on Computers for Handicapped Persons*, Pp. 153–156. Springer.

Debevec, P.
2002. Image-based lighting. *IEEE Computer Graphics and Applications*, 22(2):26–34.

Delabarre, E. B.
1898. A method of recording eye-movements. *The American Journal of Psychology*, 9(4):572–574.

Dodge, R. and T. S. Cline
1901. The angle velocity of eye movements. volume 8, P. 145. The Macmillan Company.

Duchowski, A. T., D. H. House, J. Gestring, R. Congdon, L. Świrski, N. A. Dodgson, K. Krejtz, and I. Krejtz
2014. Comparing estimated gaze depth in virtual and physical environments. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, Pp. 103–110. ACM.

Egger, B., S. Schönborn, A. Forster, and T. Vetter
2014. Pose normalization for eye gaze estimation and facial attribute description from still images. In *German Conference on Pattern Recognition*, Pp. 317–327. Springer.

Fanelli, G., M. Dantone, J. Gall, A. Fossati, and L. Van Gool
2013. Random forests for real time 3d face analysis. *International Journal of Computer Vision*.

Ferhat, O. and F. Vilarino
2015. Low cost eye tracking: The current panorama. *Journal of Computational Intelligence and Neuroscience*, 22(23):24.

Fu, L. and L. B. Kara
2011. Neural network-based symbol recognition using a few labeled samples. *Computers & Graphics*, 35(5):955–966.

Funes Mora, K. A., F. Monay, and J.-M. Odobez
2014. EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras. *Proceedings of the ACM Symposium on Eye Tracking Research and Applications*.

Furukawa, Y. and J. Ponce
2010. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376.

Gross, R., I. Matthews, J. Cohn, T. Kanade, and S. Baker
2007. The cmu multi-pose, illumination, and expression (multi-pie) face database. *CMU Robotics Institute. TR-07-08, Tech. Rep*.

Hansen, D. W., J. P. Hansen, M. Nielsen, A. S. Johansen, and M. B. Stegmann
2002. Eye typing using markov and active appearance models. In *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, Pp. 132–136. IEEE.

Hansen, D. W. and Q. Ji
2010. In the eye of the beholder: A survey of models for eyes and gaze. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):478–500.

Hansen, D. W. and A. E. Pece
2005. Eye tracking in the wild. *Computer Vision and Image Understanding*, 98(1):155–181.

Hinton, G., O. Vinyals, and J. Dean
2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Huang, J., B. Heisele, and V. Blanz
2003. Component-based face recognition with 3d morphable models. In *Audio-and Video-Based Biometric Person Authentication*, Pp. 27–34. Springer.

Huang, Q., A. Veeraraghavan, and A. Sabharwal
2015. Tabletgaze: A dataset and baseline algorithms for unconstrained appearance-based gaze estimation in mobile tablets. *arXiv preprint arXiv:1508.01244*.

Huang, S.-C., Y.-L. Wu, W.-C. Hung, and C.-Y. Tang
2010. Point-of-regard measurement via iris contour with one eye from single image. In *Multimedia (ISM), 2010 IEEE International Symposium on*, Pp. 336–341. IEEE.

Huber, P., G. Hu, R. Tena, P. Mortazavian, W. P. Koppen, W. Christmas, M. Rätsch, and J. Kittler
2016. A multiresolution 3d morphable face model and fitting framework. In *Proc. VISAPP*.

Huey, E. B.
1898. Preliminary experiments in the physiology and psychology of reading. *The American Journal of Psychology*, 9(4):575–586.

Huey, E. B.
1900. On the psychology and physiology of reading. i. *The American Journal of Psychology*, 11(3):283–302.

Ishikawa, T.
2004. Passive driver gaze tracking with active appearance models. *Technical Report*.

Javal, E.
1879. Essai sur la physiologie de la lecture. In *Annales D'Oculistique*.

Jeni, L. A. and J. F. Cohn
2016. Person-independent 3d gaze estimation using face frontalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Pp. 87–95.

Jeni, L. A., J. F. Cohn, and T. Kanade
2015. Dense 3D Face Alignment from 2D Videos in Real-Time. *Face and Gesture*.

Jimenez, J., E. Danvoye, and J. von der Pahlen
2012. Separable subsurface scattering & photorealistic eyes rendering. In *SIGGRAPH Talks, Advances in Real-Time Rendering in Games*. ACM.

Joseph Tan, D., T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton
2016. Fits like a glove: Rapid and reliable hand shape personalization.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 5610–5619.

Kabsch, W.
1976. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923.

Kaneva, B., A. Torralba, and W. T. Freeman
2011. Evaluation of image features using a photorealistic virtual world. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 2282–2289. IEEE.

Karis, B., T. Antoniades, S. Caulkin, and V. Mastilovic
2016. Digital humans: Crossing the uncanny valley in unreal engine 4. In *Games Developer Conference (GDC) Talks*. EPIC.

Kassner, M., W. Patera, and A. Bulling
2014. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction.

Kleinke, C. L.
1986. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78–100.

Krafka, K., A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba
2016. Eye tracking for everyone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 2176–2184.

Krupinski, R. and P. Mazurek
2010. Electrooculography signal estimation by using evolution–based technique for computer animation applications. In *International Conference on Computer Vision and Graphics*, Pp. 139–146. Springer.

Le, V., J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang
2012. Interactive facial feature localization. In *European Conference on Computer Vision*, Pp. 679–692. Springer.

LeCun, Y., Y. Bengio, and G. Hinton
2015. Deep learning. *Nature*, 521(7553):436–444.

Lee, A., H. Moreton, and H. Hoppe
2000. Displaced subdivision surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, Pp. 85–94. ACM Press/Addison-Wesley Publishing Co.

Lefohn, A., B. Budge, P. Shirley, R. Caruso, and E. Reinhard
2003. An ocularist's approach to human iris synthesis. *IEEE Computer Graphics and Applications*, 23(6):70–75.

Levine, J. L.
1981. *An eye-controlled computer*. IBM Research Division, TJ Watson Research Center.

Li, D. and D. J. Parkhurst
2005. Starburst: A robust algorithm for video-based eye tracking. *Elselvier Science*, 6.

Li, T., T. Bolkart, M. J. Black, H. Li, and J. Romero
2017. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(4).

Liebelt, J. and C. Schmid
2010. Multi-view object class detection with a 3d geometric model. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, Pp. 1688–1695. IEEE.

Liu, D. C. and J. Nocedal
1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Loop, C.
1987. *Smooth Subdivision Surfaces Based on Triangles*. Department of Mathematics, University of Utah.

Loper, M. M. and M. J. Black
2014. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, Pp. 154–169. Springer.

Lu, F., T. Okabe, Y. Sugano, and Y. Sato
2011. A head pose-free approach for appearance-based gaze estimation. In *BMVC*, Pp. 1–11.

Lu, F., Y. Sugano, T. Okabe, and Y. Sato
2014. Adaptive linear regression for appearance-based gaze estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(10):2033–2046.

Malbouisson, J. M., A. A. V. e Cruz, A. Messias, L. V. Leite, and G. D. Rios
2005. Upper and lower eyelid saccades describe a harmonic oscillator function. volume 46, Pp. 857–862. The Association for Research in Vision and Ophthalmology.

Mora, K. and J.-M. Odobez
2014. Geometric generative gaze estimation (g3e) for remote rgb-d cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 1773–1780.

Nguyen, K., C. Wagner, D. Koons, and M. Flickner
2002. Differences in the infrared bright pupil response of human eyes. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, Pp. 133–138. ACM.

Okada, R. and S. Soatto
2008. Relevant feature selection for human pose estimation and localization in cluttered images. In *European Conference on Computer Vision*, Pp. 434–445. Springer.

Orvalho, V., P. Bastos, F. Parke, B. Oliveira, and X. Alvarez
2012. A facial rigging survey. In *Proc. of the 33rd Annual Conference of the European Association for Computer Graphics-Eurographics*, Pp. 10–32.

Paysan, P., R. Knothe, B. Amberg, S. Romdhani, and T. Vetter
2009. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, Pp. 296–301. Ieee.

Peng, X., B. Sun, K. Ali, and K. Saenko
2014. Exploring invariances in deep convolutional neural networks using synthetic images. *CoRR, abs/1412.7122*, 2(4).

Phillips, P. J., P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek
2005. Overview of the face recognition grand challenge. In *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on*, volume 1, Pp. 947–954. IEEE.

Priamikov, A. and J. Triesch
2014. Openeyesim - a platform for biomechanical modeling of oculomotor control. In *ICDL-Epirob*, Pp. 394–395.

Ramamoorthi, R. and P. Hanrahan
2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, Pp. 497–500. ACM.

Rätsch, M., P. Quick, P. Huber, T. Frank, and T. Vetter
2012. Wavelet reduced support vector regression for efficient and

robust head pose estimation. In *Computer and Robot Vision (CRV), 2012 Ninth Conference on*, Pp. 260–267. IEEE.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi
2015. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*.

Riedmiller, M. and H. Braun
1992. Rprop - a fast adaptive learning algorithm. In *Proc. of ISCIS VII, Universitat*. Citeseer.

Romdhani, S. and T. Vetter
2005. Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, Pp. 986–993. IEEE.

Ruhland, K., S. Andrist, J. Badler, C. Peters, N. Badler, M. Gleicher, B. Mutlu, and R. Mcdonnell
2014. Look me in the eyes: A survey of eye and gaze animation for virtual agents and artificial systems. In *Eurographics*, Pp. 69–91.

Sagar, M. A., D. Bullivant, G. D. Mallinson, and P. J. Hunter
1994. A virtual environment and model of the eye for surgical simulation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, Pp. 205–212. ACM.

Sagonas, C., E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic
2016. 300 faces in-the-wild challenge: Database and results. *Image and Vision Computing*, 47:3–18.

Sagonas, C., G. Tzimiropoulos, S. Zafeiriou, and M. Pantic
2013. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, Pp. 397–403.

Scherbaum, K., J. Petterson, R. S. Feris, V. Blanz, and H.-P. Seidel
2013. Fast face detector training using tailored views. In *Proceedings of the IEEE International Conference on Computer Vision*, Pp. 2848–2855.

Schneider, T., B. Schauerte, and R. Stiefelhagen
2014. Manifold alignment for person independent appearance-based gaze estimation. In *Pattern Recognition, International Conference on*, Pp. 1167–1172.

Sesma, L., A. Villanueva, and R. Cabeza
2012. Evaluation of pupil center-eye corner vector for gaze estimation using a web cam. In *Proceedings of the symposium on eye tracking research and applications*, Pp. 217–220. ACM.

Shotton, J., T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore
2013. Real-time human pose recognition in parts from a single depth image. volume 56, Pp. 116–124. ACM.

Shrivastava, A., T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb
2016. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*.

Skodras, E., V. G. Kanas, and N. Fakotakis
2015. On visual gaze tracking based on a single low cost camera. *Signal Processing: Image Communication*, 36:29–42.

Smith, B., Q. Yin, S. Feiner, and S. Nayar
2013. Gaze Locking: Passive Eye Contact Detection for Human Object Interaction. In *ACM Symposium on User Interface Software and Technology (UIST)*, Pp. 271–280.

Stratton, G. M.
1902. Eye-movements and the aesthetics of visual form. *Philosophische Studien*, 20:336–359.

Sugano, Y., Y. Matsushita, and Y. Sato
2013. Appearance-based gaze estimation using visual saliency. *IEEE transactions on pattern analysis and machine intelligence*, 35(2):329–341.

Sugano, Y., Y. Matsushita, and Y. Sato
2014. Learning-by-synthesis for appearance-based 3d gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 1821–1828.

Sugano, Y., X. Zhang, and A. Bulling
2016. Aggregaze: Collective estimation of audience attention on public displays. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, Pp. 821–831. ACM.

Swirski, L., A. Bulling, and N. Dodgson
2012. Robust real-time pupil tracking in highly off-axis images. In

Proceedings of the Symposium on Eye Tracking Research and Applications, Pp. 173–176. ACM.

Swirski, L. and N. Dodgson
2013. A fully-automatic, temporal approach to single camera, glint-free 3d eye model fitting. *Proceedings of Pervasive Eye Tracking and Mobile Eye-Based Interaction (PETMEI)*.

Świrski, L. and N. Dodgson
2014. Rendering synthetic ground truth images for eye tracker evaluation. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, Pp. 219–222. ACM.

Tan, K.-H., D. J. Kriegman, and N. Ahuja
2002. Appearance-based eye gaze estimation. In *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, Pp. 191–195. IEEE.

Taylor, J., L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, et al.
2016. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143.

Thies, J., M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt
2015. Real-time expression transfer for facial reenactment. ACM.

Thies, J., M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner
2016. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Timm, F. and E. Barth
2011. Accurate eye centre localisation by means of gradients. *Proceedings of the Sixth International Conference on Computer Vision Theory and Applications*, 11:125–130.

Tonsen, M., X. Zhang, Y. Sugano, and A. Bulling
2016. Labelled pupils in the wild: a dataset for studying pupil detection in unconstrained environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, Pp. 139–142. ACM.

Valenti, R. and T. Gevers
2008. Accurate eye center location and tracking using isophote

curvature. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Pp. 1–8. IEEE.

Valenti, R., J. Staiano, N. Sebe, and T. Gevers
2009. Webcam-based visual gaze estimation. In *International Conference on Image Analysis and Processing*, Pp. 662–671. Springer.

Vertegaal, R. et al.
2003. Attentive user interfaces. *Communications of the ACM*, 46(3):30–33.

Wang, C., F. Shi, S. Xia, and J. Chai
2016. Realtime 3d eye gaze animation using a single rgb camera. *ACM Transactions on Graphics (TOG)*, 35(4):118.

Wang, J., E. Sung, and R. Venkateswarlu
2003. Eye gaze estimation from a single image of one eye. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Pp. 136–143. IEEE.

Wei, S.-E., V. Ramakrishna, T. Kanade, and Y. Sheikh
2016. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 4724–4732.

Williams, O., A. Blake, and R. Cipolla
2006. Sparse and semi-supervised visual mapping with the s3gp. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, Pp. 230–237. IEEE.

Wollaston, W. H.
1824. On the apparent direction of eyes in a portrait. *Philosophical Transactions of the Royal Society of London*, 114:247–256.

Wood, E., T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling
2016a. A 3d morphable eye region model for gaze estimation. In *European Conference on Computer Vision*, Pp. 297–313. Springer.

Wood, E., T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling
2015. Rendering of eyes for eye-shape registration and gaze estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, Pp. 3756–3764.

Wood, E., T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling
2016b. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Symposium on Eye Tracking Research and Applications*.

Wood, E. and A. Bulling
2014. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, Pp. 207–210. ACM.

Wu, C., D. Bradley, P. Garrido, M. Zollhöfer, C. Theobalt, M. Gross, and T. Beeler
2016. Model-based teeth reconstruction. *ACM Transactions on Graphics (TOG)*, 35(6):220.

Wu, H., Y. Kitagawa, T. Wada, T. Kato, and Q. Chen
2007. Tracking iris contour with a 3d eye-model for gaze estimation. In *Asian Conference on Computer Vision*, Pp. 688–697. Springer.

Xiong, X. and F. De la Torre
2013. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pp. 532–539.

Yarbus, A. L.
1967. *Eye movements during perception of complex objects*. Springer.

Young, D., H. Tunley, and R. Samuels
1995. *Specialised hough transform and active contour methods for real-time eye tracking*. University of Sussex, Cognitive & Computing Science.

Yu, J., D. Farin, C. Krüger, and B. Schiele
2010. Improving person detection using synthetic training data. In *International Conference on Image Processing*.

Zhang, W., T.-N. Zhang, and S.-J. Chang
2010. Gazing estimation and correction from elliptical features of one iris. In *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 4, Pp. 1647–1652. IEEE.

Zhang, X., Y. Sugano, M. Fritz, and A. Bulling
2015. Appearance-based gaze estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 4511–4520.

Zhang, X., Y. Sugano, M. Fritz, and A. Bulling
2016. It's written all over your face: Full-face appearance-based gaze estimation. *arXiv preprint arXiv:1611.08860*.

Zhou, B., H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba
2016. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*.

Zhu, X. and D. Ramanan
2012. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, Pp. 2879–2886. IEEE.