Infrared Model-Based Eye-Tracking for Smartphones

by

Braiden Brousseau

A thesis submitted in conformity with the requirements
for the degree of Ph.D.
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Infrared Model-Based Eye-Tracking for Smartphones

Braiden Brousseau
Ph.D.
Graduate Department of Electrical and Computer Engineering
University of Toronto
2020

Estimating a subject's point of gaze is known as *gaze estimation* or *eye-tracking* and is used in many domains including as an indicator of cognitive, psychiatric and neurological states of individuals. The most accurate remote Point of Gaze (PoG) estimation methods that allow the highest amount of free head movements use infrared (IR) light sources and IR cameras. This type of IR hardware has already arrived on flag-ship smartphones to enable facial identification algorithms but is not yet accessible to developers. If manufactures granted access to this hardware on future commercial smartphones, it has the potential to enable low-cost, robust, and accurate infrared mobile eye-tracking. However, desktop eye-tracking methods assume, both implicitly and explicitly, that the system is stationary. This assumption is not valid for hand-held mobile-device-based eye-trackers, and this thesis aims to address the new sources of error associated with the free movement of the device for mobile eye-tracking. First, current gaze estimation models assume that the relative roll between the system and the subjects' eyes (the 'R-Roll') is roughly constant. We developed a new method to determine the PoG which compensates for the effects of R-Roll on the accuracy of the PoG. Then to accurately determine input parameters of the model (pupil center and corneal reflections) in eye images that exhibit large variability due to free device motion, we use an efficient hierarchy of Convolutional Neural Networks (CNNs) with a novel center-of-mass output layer. Finally, we combined these innovations in a complete end-to-end hand-held eye-tracking system using an industrial prototype smartphone with integrated infrared illumination and infrared camera. Experimental results using this system show that across 8 unsupervised subjects looking at 20mm square targets (with the device free to move) 96% of all gaze estimates were within the intended targets with an average bias of 0.72°. Our hybrid approach of a 3D gaze estimation model with a CNN feature extractor achieved a gaze bias that is significantly (400%) better than mobile systems in prior approaches during a higher degree of device motion than was reported in any prior work.

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisors, funding partners, colleagues, friends and family that have supported me throughout my academic career. Firstly I would like to thank Huawei Technologies for the funding support and collaboration throughout this work, without which this project would not have been realized. Next, I would like to thank my supervisors Jonathan Rose and Moshe Eizenman, whose expertise, guidance and support were valued and appreciated. Their mentorship throughout this process has facilitated my personal growth as a scientist, engineer, and educator. Finally I would like to extend a special acknowledgment to my aunt Robin whose lifetime pursuit of excellence and achievement has served as a guidepost since I was young. Without her example and guidance I may never have continued on the path that eventually leads to this degree.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The capability to estimate where a subject is looking is known as *gaze estimation* or *eye-tracking*. This capability enhances applications in a wide array of domains, including the measurement of advertising efficacy [7, 8], instrumentation of the reading process![9, 10, 11], automotive safety [12, 13, 14], pilot training [15, 16, 17], and providing accessibility interfaces [18, 19, 20, 21]. Eye tracking can also generate indicators of cognitive, psychiatric, and neurological states of individuals. In particular, four of the five most common mental health disorders (measured in disability-adjusted life years lost [22]) have seen promising eye-tracking severity assessment or diagnosis capability demonstrated in the past two decades. These include depression [23], schizophrenia [24, 25], bipolar affective disorder [26, 27], and Alzheimer's disease [28]. Eye tracking also enhances the evaluation of much less common mood, attention, perception, and learning disorders, such as autism [29, 30, 31], anxiety [32], post-traumatic stress disorder [33], dyslexia [34], and anorexia [35, 36]. Globally, in any given year, approximately 30% of the population is affected by some form of mental health issue, yet only one-third of those affected receive the care they need [37, 38]. This is in part due to the lack of sufficient access to expert human clinicians. One aspect of clinical interaction is the diagnosis and tracking of mental health states. Eye-tracking based measurement could lead to *objective* measures of mental health states and can be used by clinicians to improve clinical care of patients with psychiatric disorders. If it were possible to deploy eye-tracking technology at low cost on a widely available consumer device, then such technology may become an essential component of the future of mental health services.

The above work provided compelling data that suggest that specialized and expensive *stationary* eye-tracking systems can be used to support clinical diagnosis. These systems place the camera(s) and light sources statically in the environment and have the subject sit in front of them. These work well for research purposes, or in high-end clinical settings, but have limited applicability for at-home consumer-based measurement because they are costly and requires the purchase of specialized software and new hardware. This motivates the main goal of this research which is bringing high-quality eye-tracking technology into the modern smartphone, which is the most widely accessible and deployed personal computing platform [39] in history. The purpose of this research is to develop an eye-tracking system for hand-held mobile devices, such as smartphones, with results that approach high-performance stationary eye trackers. Currently, the most accurate video-based stationary eye-tracking produce estimation bias of under 0.5° [40]. These systems use methods that rely on artificial illumination of the eyes with infrared (IR) light sources and capture images of the eyes with IR camera sensors [40]. This hardware is not yet standard in commercial smartphones, and so all of the prior work on mobile eye-tracking has used techniques which rely only on natural light illumination and cameras [41, 42, 43, 44, 45, 46, 47, 48]. These systems have reported a gaze bias no better than 3°, even in favorable operating conditions. One reason for this large bias is the inability of use infrared light to illuminate the face and infrared cameras to capture images of the face. Smartphones may not always face this limitation as IR cameras, and IR light sources have begun to appear in some commercial smartphones [49, 50] to enhance face tracking and authentication systems. Apple, as part of it's augmented reality framework (ARKit [51]) has even now introduced calibration-free gaze estimation as an API for iPhoneX. While the vendors of these phones do not yet make the IR components available to third party software developers, the devices are otherwise almost ready to enable IR-illuminated based eye tracking.

For this reason, this thesis explores the performance and viability of more accurate IR-based eye tracking in the mobile form factor.

Even with IR illumination and cameras, there are several significant challenges associated with the development of accurate hand-held eye tracking. First, the most critical difference between stationary and hand-held eye-tracking systems is that a hand-held smartphone *moves* significantly relative to the user during operation. It is this significant relative motion between the system and the user which presents new challenges for hand-held eye-tracking systems. Secondly, smartphones are considerably more energy-constrained, and this limits the total amount of sustained IR illumination power that would be acceptable during eye-tracking sessions. Low IR illumination reduces the quality of eye-regions images and increases the difficulty of the front-end image processing that locates eye features. Finally, smartphones have much smaller displays than typical stationary eye-tracking systems, which make high accuracy eye-tracking critical to distinguish between a sufficient number of regions on display to enable useful eye-tracking applications.

1.1 Contributions

Video-based eye-tracking systems contain two primary stages; front-end feature extraction, and back-end gaze estimation. The front-end feature extractor identifies and locates specific features of a subject's face and eyes from images of the subject's head. The back-end gaze estimation stage uses the feature locations to compute a point-of-gaze (PoG). The goal and key contributions of this thesis are a) the creation of an end-to-end smartphone-based IR eye-tracking system that is as accurate as a state-of-the-art stationary system, and b) the solutions to issues related to the significant relative motion between the device and the subject's head.

1.1.1 Back-End Eye Tracking Model

The back-end eye tracking model estimates the PoG from input eye feature locations. In this thesis we extended an existing model that is insensitive to pitch and yaw head-movements [40], to account for head roll (a specific kind of motion that is likely to occur only in the mobile/hand-held context). The original model [40] assumes that the orientation and the position of the system (which is the camera, the light sources, and the display) does not change during use. This model measures some parameters in a fixed world coordinate system (WCS). In a hand-held mobile eye-tracking system, which is free to move, the model should not estimate these parameters in the WCS. One core contribution of this work was to identify and address these mobility concerns in the development of the back-end gaze estimation model.

1.1.2 Front-End Feature Extractor

The back-end gaze estimation system requires accurate locations of the input eye features to produce accurate results. These locations must be estimated with sub-pixel accuracy to achieve the gaze estimation accuracy of high-end stationary eye-tracking systems. This level of precision is more difficult to achieve with a hand-held mobile device, as the appearance of the face and eyes can change quite dramatically with significant changes in position and orientation of the device retaliative to the head. This increased variability of input images, together with the limited IR illumination capacity of the illumination sources in the smartphone and comparatively small front-facing camera sensor of a smartphone, make the process of locating eye features robustly and accurately more difficult than in stationary eye-tracking systems. In this work, we explore both traditional rule-based computer-vision methods and machine learning techniques to estimate eye feature locations. We present new efficient, and robust (in the mobile context) approaches.

1.1.3 End-to-End Complete System

We combined the previous two contributions into a complete end-to-end eye-tracking system which runs on an industrial prototype infrared smartphone from Huawei [52] and compare it with prior researchers' work [46, 47, 48]. The complete system achieves both accuracy and robustness in a mobile environment, producing a gaze estimation bias of 0.72°. Our system maintains this gaze bias during large relative movements between the device and the subject. Described another way, when subjects looked at 20 mm square grids on the display of the smartphone, 96% of all gaze estimates fell within the intended grid-square. These results occurred during 30 individual unsupervised eye-tracking sessions where the user was free to move the smartphone as she/he wished.

The remainder of the thesis is organized as follows: Chapter 2 provides background information on previous eye-tracking approaches, including front-end feature extraction techniques and the mobile eye trackers that use them. Chapter 3 gives a top-level overview of the complete mobile eye-tracking system, both the physical system and the required software. Chapter 4 describes the new back-end model that accounts for the roll of a system relative to the user. Chapter 5 describes the front-end feature extraction algorithms - both the rule-based method and the machine learning-based methods. Chapter 6 presents the end-to-end mobile eye-tracking system and a detailed comparison to existing mobile eye trackers. Finally, Chapter 7 offers the direction of future work and conclusions.

Chapter 2

Background and Related Work

Eye-tracking systems differ across several axes. One such axis, discussed in Chapter 1, separates approaches based on how the eye image is illuminated, with either natural light or artificial infrared illumination. Another axis concerns the physical attachment of the eye-tracking system to the subject: some eye-tracking systems monitor eye-movements remotely (*remote non-contact*) and some are attached to a subjects head (*head-mounted*). Head-mounted gaze estimation systems have existed in many forms for more than 50 years [53, 54, 55] and this modality is still commonly used in research [56, 57, 58, 59]. The performance of head-mounted systems benefit from proximity to the eyes (resulting in high resolution images of the eye), and minimization of relative motion between the system and the subject's eyes (captured eye images by the camera(s) do not change dramatically).

By contrast, remote non-contact systems which remain stationary with respect to the environment have to capture eye-images when subjects are free to move their heads (i.e., the captured eye-images have lower resolution) and the appearance of the eye-images shows more variability as the orientation and distance of the camera to the subjects face change.

A hand-held smartphone eye tracker is a remote non-contact gaze estimation system that has the additional issue that the system itself is not stationary with respect to the environment. Even so, remote non-contact eye-tracking methods are the closest relevant prior work, and in this chapter remote eye-tracking systems will be reviewed in depth. This chapter provides an overview of remote gaze estimation methods, the feature extraction techniques which support them, and the mobile eye-tracking systems that employ them.

2.1 Remote Non-Contact Gaze Estimation Methods

Figure 2.1 illustrates a schematic of a human eye, including the visual axis of an eye, which is the vector that connects the nodal point of the eyes optical system and the most sensitive part of the retina, the fovea. The point-of-gaze of a subject is the intersection of the visual axis vector of the subject's eye with the scene of interest [40]. Often, the scene of interest is a computer screen, or in the case of this research, a mobile phone display.

General overviews of the methods developed for point-of-gaze estimation are found in [60, 61, 62, 63]. Also, a detailed history of remote non-contact gaze estimation models can be found in [4]. Here we present a brief history of the progression of these approaches along with their theoretical limitations.

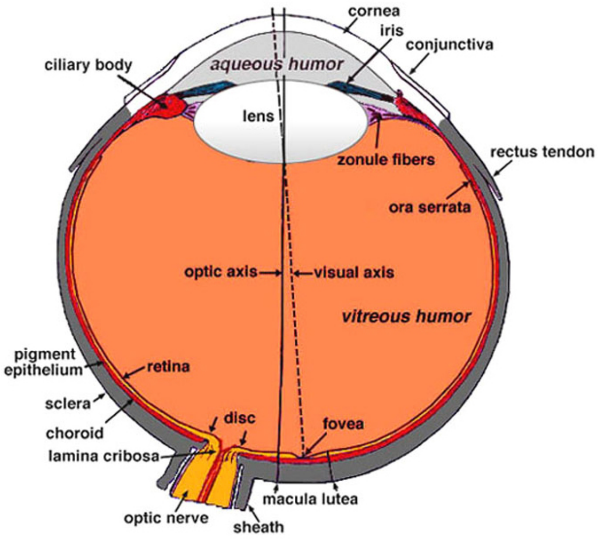


Figure 2.1: Schematic of human eye from above [1]

This review will be used to highlight the limitations of current mobile eye-tracking systems later in this chapter. As described above, we divide remote gaze-estimation methods into two categories: those methods which use natural light for face illumination and those that use artificial infrared illumination.

2.1.1 Natural Light Remote Non-Contact Gaze Estimation Techniques

Natural light remote non-contact gaze estimation methods are the most common techniques currently used in mobile eye-tracking because they require only ambient lighting and a standard RGB visible light camera. The front-facing 'selfie' camera of nearly all smartphones use this type of sensor. Three popular visible light gaze estimation methods used in mobile eye trackers are limbus back projection [64], pupil/iris centres with head pose [65], and appearance-based methods [66].

Limbus Back-Projection

The circular boundary between the iris and sclera (the boundary between the colored and white parts of the eye) is called the *limbus*. When the limbus is imaged by the camera's optics it appears as an ellipse when projected on the physical camera sensor. The eccentricity of this ellipse increases as the deviation between the optical axis of the eye and the optical axis of the camera increases. One can use the ellipse and back-projection techniques to estimate the relative orientation of the eye and the camera [67]. Some eye trackers directly map the eccentricity of this ellipse with known gaze locations on the display and then estimate the PoG with a simple interpolation process [42]. This only works when there is no relative motion between the subject's head and the eye-tracking system. The deviation between the optical axis of the eye and the camera changes during relative motion, even when the subject is gazing at the same physical location. The position of the eye in space should also be estimated to account for this relative motion. As the physical size of a subject's iris is constant, the position and size of the ellipse that describes the limbus in the image can be used to do this. Changes in the limbus ellipse major axis length are due to changes in distance between the eye and the camera. A calibration procedure in which a subject is asked to gaze at two or more targets can estimate the physical width of the iris. With both the optical axis of the eye estimated from limbus eccentricity and distance estimated from limbus size, several studies [64, 68, 18] demonstrate gaze estimation with approximately 1° RMS estimation error.

Measuring the limbus boundary accurately and robustly can be challenging, as a subject's eyelid or eye socket often cause occlusion. Improving this method requires increasing the accuracy of the detected ellipse parameters under these types of conditions. Due to the sensitivity of the method to errors associated with fitting an ellipse to points on the limbus, in practice, this technique requires high resolution, high-quality imaging of a subject's eye region. Recent efforts to implement limbus back projection on lower quality, standard webcams [69] resulted in significantly higher mean estimation errors of over 2°. Another particularly challenging scenario occurs when the subjects optical axis is close to parallel with respect to the cameras optical axis (i.e., the eccentricity of the limbus is small). In this scenario, noise in the estimation of the limbus boundary will have a significant impact on the fitted ellipse parameters.

A mobile eye-tracker will suffer from both of these issues. First, the optical axis of the front-facing camera is close to the normal to the center of the display so when subjects look directly at the display the limbus will appear with low eccentricity on the image sensor of the front-facing camera. Secondly, the physical size constraints of pixels in the front-facing camera of the smartphone result in noisier eye images on smartphones. For these reasons, we would expect poor gaze estimation performance of this technique on mobile devices.

Pupil/Iris Centre Estimation and Head Pose

In the method that uses estimation of the *pupil/iris centers and head pose* to calculate the point of gaze, the position and orientation of the eye are estimated, and then the optical axis is reconstructed. The optical axis is then intersected with the display to generate a PoG. The position of the eye is estimated by using the geometric relationship between facial features produced by a head tracker. Then the orientation of the eye is estimated based on the direction of a vector that connects the center of the pupil (or iris) with the center of eye-rotation. Before using the system, this method requires a calibration procedure [70] to measure the subject-specific radius of both the iris and eyeball. Systems that use this general gaze estimation method include a single camera [65, 71, 72], multiple cameras [73, 70], and even depth cameras [74] configurations (such as the Microsoft Kinect).

Single-camera configurations have historically had more trouble accurately locating the position of the eye due to lack of depth information (this is improving with new deep learning-based depth estimation [75]). Due to this limitation, single-camera configurations have been less accurate than other configurations. A single-camera configuration presented in [65] produced a gaze estimation accuracy of only 4-5°. A system using lower resolution eye regions of just 30x15 pixels achieved just 6° gaze estimation accuracy [71].

The work reported in [71] did not use a calibration procedure to estimated subject-specific eye parameters. Instead, it continuously estimated eye-parameters during operation purely from the eye and facial features without labeled PoG targets. The estimation error of [71] was high, but because of continuous re-calibration, it was robust to relatively large variations in operating conditions. If one assumes that the head is fixed relative to the camera of the eye-tracking system one can use the work presented in [74] to increase speed and accuracy. This Haar-based approach convolved a single small feature mask over a down-sampled version of the image. The method is quick, robust to significant lighting variations, and reasonably accurate in the context of an eye-tracking system. This work reported excellent mean estimation error of under 1° but used very high-resolution eye regions from a camera that appeared to be no more than 5-10cm from the users' heads. Also, by removing the full head pose estimation, their system did not have any tolerance to head pose variation.

Two-camera systems that can use stereo vision to measure depth (or RGBD systems which automatically produced depth information) can more accurately estimate the eye position in space [76]. A two-camera system, with a very low eye region resolution (the diameter of the iris is only 10-pixels) achieved a mean estimation error of under 3° [70]. This gaze estimation error of 3° came from averaging the PoGs computed from both eyes in both stereo frames (4 eye regions in total). Averaging of this kind is a common technique to compensate for noise in the iris center estimation in each frame. The authors did not report the mean error of a single PoG estimation. Another system used the depth information provided by the Microsoft Kinect API [74] to more easily locate the subject's eyes in space and achieve an average gaze estimation of 4°. The principal contribution here was not image processing or eye feature detection but the demonstration of the depth-based head pose model on a cheap readily accessible hardware platform.

The gaze estimation methods that use the pupil/iris centre with head pose do not report results with large relative movements between the eye-tracker and the head, and therefore it is difficult to compare their performance with the system that we are developing. Recall that our goal is to approach the performance of the best stationary remote non-contact eye-tracking systems which report a gaze estimation accuracy of as low as 0.5°. The next gaze estimation method, which also uses natural light, attempts to directly map images to a PoG, rather than using an underlying gaze estimation model.

Appearance-Based Methods

With appearance-based methods, the input eye features are portions of face images that include the eyes. These methods attempt to find a 'similar' images in a labeled (the label is the gaze direction) reference set. These techniques often rely on an initial machine learning/training process from a set of labeled training images. As with many machine learning processes, training with a dataset that reflects that variance expected during use is critical.

An early appearance-based approach used a fully connected neural network trained with 2000 images, each labeled with the known gaze locations [66]. The system reported PoG estimates with an average error of approximately 2°, but only in operating conditions without head movements or lighting variations.

A significantly more accurate appearance-based approach presented in [77] labeled images for known point-of-gaze targets on a 15 x 15 grid. Then, during operation, new eye regions are compared and matched to the closest labeled images. Interpolation between the ground truth from these labeled images is how the PoG is estimated. The gaze estimation accuracy for this method, was quite good at 0.4°. However, the approach is not robust in the presence of lighting changes and head pose variation. Any changes to the physical system (such as changing the camera, the distance to the subject, or the size of the monitor) would require generating new dataset and retraining from scratch.

Another appearance-based method introduces geometric priors [78, 79] to address head pose variation. In this approach, a method similar to [77] is trained with a set of static head poses. Then a 5-second video is captured in which the users rotate their head while gazing at a single target. An analysis of this video maps the geometric distortion of the appearance of the eyes. This information is later used to estimate what the appearance of the subject's eye *would* look like during minor head rotation for each of the original training images. It is an intelligent solution that avoids the multiplicative increase of training images required to model head rotation parameters, as is the case with previous appearance methods. This approach reduced the average gaze estimation accuracy from 10° to 1.5° when the user was allowed ± 30° of head rotation. With free head motion, the mean estimation error settles between 2-3°.

A deep neural network-based eye tracking model was introduced in [46]. Here the training set size was dramatically increased when compared to other efforts. The authors built an iPhone application that subjects could easily download and spend a few minutes looking at training targets on display. Use of this application yielded tens of thousands of labeled images from nearly 1500 people that were used to train a convolutional neural network. The unsupervised collection of training images resulted in natural variations in head pose and lighting between people. This approach became quite robust to novel faces and reported a calibration-free gaze estimator that works with visible light. The work reported gaze estimation accuracy of approximately 3-4°.

Some techniques involve the generation of a large number synthetic eye images to aid in the learning process for appearance based methods. Some of these works are designed explicitly to improve feature estimation [80] for use downstream in an eye tracking system. Others incorporate eye shape and gaze information into the image synthesis process using conditional bidirectional generative adversarial networks for the purpose of end to end gaze estimation [81]. Other uses of deep neural networks include estimating fixation or saliency maps in order to enable background calibration of an eye tracking system without needed to explicitly present targets at which the subject must gaze before using the system [82].

A common problem in eye tracking with learned appearance based approaches is the high correlation between head pose and gaze direction. By inadvertently learning head pose many systems can approach a few degrees of gaze estimation error but then have trouble improving from there. One work presented in [83] used a bayesian adversarial learning approach which deters the latent representation of the eye region from encoding information about head pose. In theory this should direct the learning process to focuses on eye features instead of head features.

Deep neural networks trained on more massive datasets have continued to improve the robustness and reliability of appearance-based eye-tracking models. However, they have not yet dramatically improved the overall accuracy as many systems hover in 3° territory.

Model-Based Methods

These visible light techniques use a model of the optical properties of the eyeball. Eye modeling includes properties such as the center of curvature of the cornea, the visual/optical axis deviation, the centre of rotation of the eye, the radius of corneal curvature and the location of the center of the pupil. Model-based methods in the visible light domain are more challenging than in the infrared domain and as such there are less examples of these systems.

One 3D gaze estimation systems that used a single camera and did not require infrared illumination was initial presented in [84] as preliminary work. This technique was advanced over time into a deformable face and eye model which produced 3° gaze estimation results in an uncalibrated single camera system [85]. This technique resulted in an exceptionally general and efficient system which operated above 30fps on a standard desktop processor without accelerators. However it, as with all current visible light techniques, have not achieved the most accurate gaze estimation results presented in the literature which use artificial infrared illumination.

2.1.2 Artificial Illumination with Infrared Light for Remote Non-Contact Gaze Estimation

All previously mentioned gaze estimation models have assumed that images were acquired with no artificial illumination. A critical branch of gaze estimation methods, which was introduced in the mid 20th century [86], utilize artificial illumination of the face (usually with infrared light to minimize interference with normal vision). The reflections of the artificial light by the subject's corneas create a virtual image of the light source(s) behind the cornea (corneal reflections) that can be used for the estimation of the point-of-gaze. Figure 2.2 show corneal reflections of two infrared light sources. The infrared light sources do not cause pupil dilation or discomfort for the subject as the retina is not sensitive to infrared light. Several methods can estimate a subjects point-of-gaze from the location of the corneal reflections and the center of the pupil once they were extracted from infrared images. Infrared eye-tracking methods have been the most accurate remote non-contact point-of-gaze estimation systems presented in the literature. They are currently in use by several commercial entities [87, 88, 89, 90]. Below we will discuss several different IR gaze estimation methods.

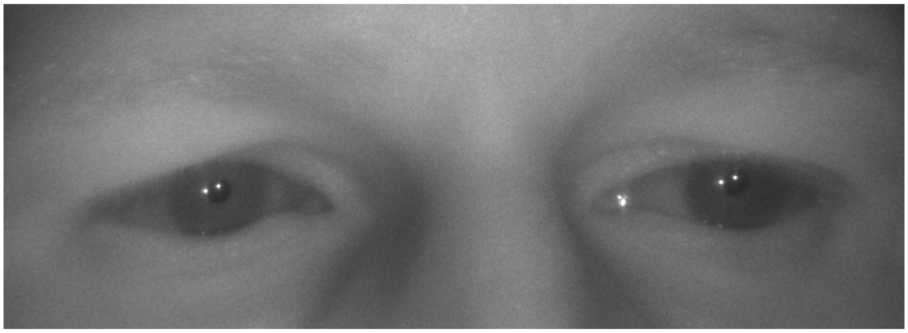


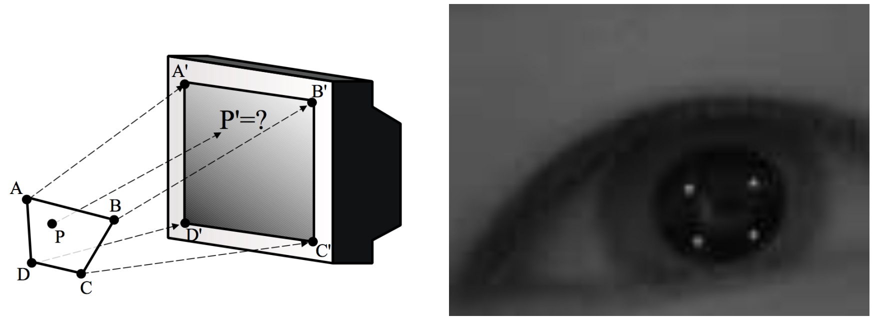
Figure 2.2: infrared eye regions with corneal reflections

Corneal Reflection Pupil Centre Vector

This method utilizes a single infrared LED light source and a single infrared camera. In this approach, the magnitude of the vector between the pupil-center and the corneal reflection can be used to determine the magnitude and direction of gaze [91]. The work done in [91] was the first gaze estimation method to result in 1° average gaze estimation accuracy. However, the method is sensitive to head movements (after the initial calibration procedure is done).

There has been continued refinement of this approach, primarily to remove approximations from the model that converts a feature vector on the image plane into the three-dimensional gaze vector [92, 93]. A detailed analysis of various mapping functions and their limitations can be found in [94]. Since this method is accurate only with precise head and eye positions, researchers explored many sophisticated multi-camera multi-light source set-ups to improve position tracking accuracy. Everything from using both wide and narrow field-of-view cameras with motorized position and zoom control [95], to ultrasonic distance measurements [96], and a myriad of other unusual hardware set-ups.

The need for precise head/eye location makes this technique less practical in a mobile environment because the noise in the estimate of the head and eye position in space directly translates into noise in the final gaze estimation. Additionally, the expected movement of both the head and the device limit the



(a) The relation between the IR LEDs, glints, pupil center, and point of gaze

(b) 80x80 pixel image of eye region with four corneal reflections

Figure 2.3: Visualization of of cross-ratios method [2]

merit of various calibration procedures which otherwise may improve the performance of this technique.

The next infrared gaze estimation method avoids some of these issues with the use of four corneal reflections rather than just one.

Cross-Ratios

The cross-ratios method uses multiple infrared LEDs fixed to a plane (typically a computer display) to estimate a subject's PoG in that plane [2]. This methodology, visualized in figure 2.3, assumes that the perspective transformation required to map the LED geometry to the geometry of the corneal reflections is the same as the transformation to convert the PoG in the display plane to the pupil center.

Some of the key benefits of this method are that it does not require any personal calibration and is not sensitive to minor head pose variation. A significant drawback is that *all* of the corneal reflections must be found to generate the perspective transformation. It is common for one or more corneal reflections to be situated beyond the iris or occluded by the eyelid and not be visible. Another issue arises because the distance between observed corneal reflections (in the image) change in proportion to changes in the distance between the eye tracker and the subject's eye. The corneal reflection pattern has to be large enough not to merge but small enough such that it fits entirely within the user's iris region. These drawbacks ultimately limit the range of distances and orientations a subject can be from the light sources.

Detailed studies outlining sources of error using this methodology can be found in [97, 98]. These studies conclude that with the addition of a personal calibration procedure and better modeling of the eye, this method can, in principle, achieve 1° gaze estimation error. In most cases, these methods simplify the modeling of the cornea by assuming it is perfectly spherical, which is a reasonable assumption near the center of the corneal but less so at the periphery. For this reason, the gaze estimation accuracy of these systems can decrease in situations where corneal reflections are found far away from the center of the pupil. Attempts have been made to model the non-spherical nature of the cornea [99, 100] enabling a more extensive range of operating conditions for the system. In doing so [99, 100] were able to reduce mean estimation error in practice to 2° during 20° head rotation. Machine learning techniques have also been introduced [101] to try and model the visual-optical axis deviation and thus improve overall performance. When doing so [101] achieved 1-2° of gaze estimation accuracy.

The gaze estimation error in cross-ratio-based eye-tracking systems could be low enough to meet the goals of our project, but some properties of this technique are troublesome. The requirement of 4 or more infrared LEDs would require bezels around more than one edge of the smartphone to house them (which is contrary to the design atheistic of virtually all modern devices). More significantly, however, is the requirement that 4 corneal reflections be visible in the image. Over the full range of positions and orientations a subject can hold a smartphone, one corneal reflection is often occluded by an eyelid or the sclera. This technique is thus not suitable for a mobile eye tracker.

Direct Model-Based 3-D Gaze Reconstruction

The final technique we describe uses a model of the optical properties of the eyeball and the entire eye-tracking system. This model considers the intrinsic and extrinsic camera parameters, infrared LED locations, and a fully 3-dimensional model of the optical properties of the eye. Eye modeling includes properties such as the center of curvature of the cornea, the visual/optical axis deviation, the centre of rotation of the eye, the radius of corneal curvature and the location of the center of the pupil.

There are multiple published gaze estimation methods which attempt to model the eye directly [102, 103]. However, the seminal theoretical work was published in 2003 [3] and expanded in 2010 [4]. Our thesis uses, and extends, the single camera configuration presented in [3, 4, 40]. The mathematical model used to produce gaze estimates is described below.

The PoG is formally defined as the intersection of the visual axes of both eyes with the 3D scene. The visual axis is the line connecting the center of the fovea with the nodal point of the eyes, and is shown in Figure 2.4. The visual axis of the human eye slightly deviates from the optical axis, and does so slightly differently for each specific subject. The model presented in [40] first considers the problem of reconstructing the optical axis and then in a second part deals with the reconstruction of the visual axis from the optic axis, and the estimation of the PoG. It is the second part of this process which was extended in this thesis for suitability on a mobile device and this extension will be discussed in detail in Chapter 4. The first part however, in which the optical axis is estimated, is used directly in our work and will be formally described here.

If the light sources are modeled as point sources and the camera is modeled as pinhole camera then Figure 2.4 consists of a ray tracing diagram of the complete optical system of the eye model. Points are represented as 3D column vectors (and presented in a bold font) in a right handed Cartesian world coordinate system (WCS). Consider a ray that comes from light source i, $\boldsymbol{l}_i$, and reflects at a point $\boldsymbol{q}_i$ on the corneal surface such that the reflected ray passes through the nodal point of camera $\boldsymbol{o}$, and intersects the camera image plane at a point $\boldsymbol{u}_i$. This condition can be expressed in parametric form as

$$\boldsymbol{q}_i = \boldsymbol{o} + k_{q,i}(\boldsymbol{o} - \boldsymbol{u}_i) \tag{2.1}$$

for some $k_{q,i}$. If a convex spherical mirror of radius R is used to model surface of the cornea then the condition that $\boldsymbol{q_i}$ is on the corneal surface can be written as

$$||\boldsymbol{q}_i - \boldsymbol{c}|| = R \tag{2.2}$$

where $\boldsymbol{c}$ is the center of corneal curvature.

The law of reflection states that: 1) the incident ray, the reflected ray and the normal at the point of reflection are coplanar; and 2) the angles of incidence and reflection are equal. The fact that vector $(\boldsymbol{q}_i - \boldsymbol{c})$ is normal to the spherical surface at point of reflection $\boldsymbol{q}_i$, implies from condition 1) that the points $\boldsymbol{l}_i$, $\boldsymbol{q}_i$, $\boldsymbol{c}$, and $\boldsymbol{o}$ are coplanar. This can be formalized as

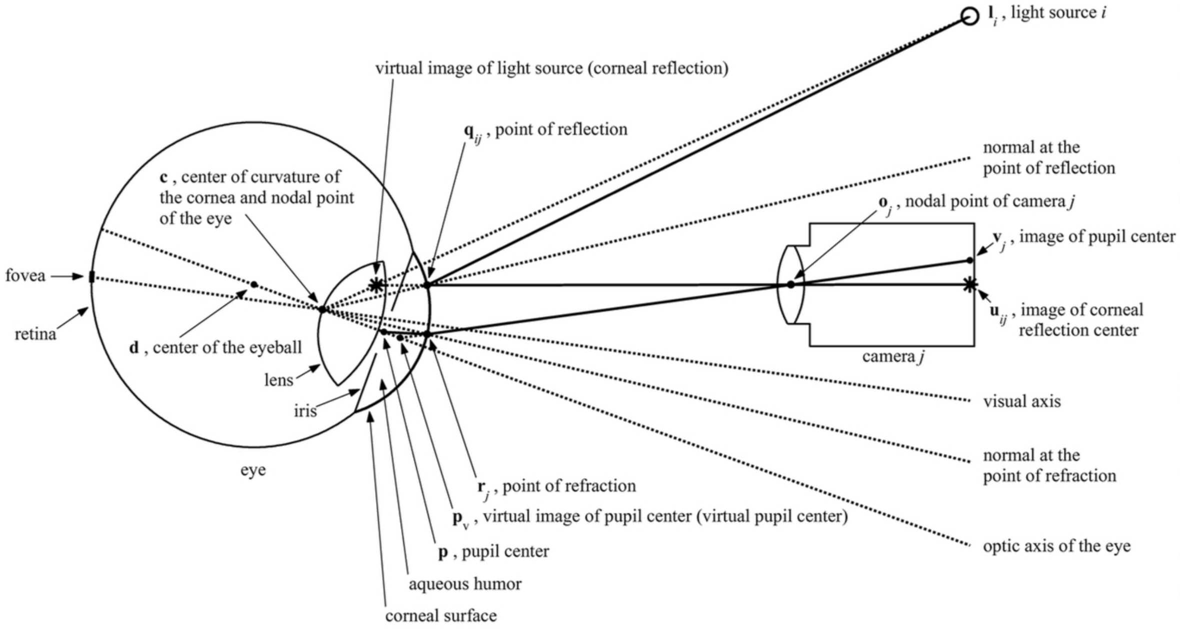

l_i , light source i
virtual image of light source (corneal reflection)
q_ij , point of reflection
normal at the point of reflection
c , center of curvature of the cornea and nodal point of the eye
o_j , nodal point of camera j
v_j , image of pupil center
fovea
u_ij , image of corneal reflection center
retina
d , center of the eyeball
camera j
lens
iris
visual axis
normal at the point of refraction
eye
r_j , point of refraction
p_v , virtual image of pupil center (virtual pupil center)
optic axis of the eye
p , pupil center
aqueous humor
corneal surface

Figure 2.4: Ray tracing diagram of system components modelled in [3, 4]

$$(\boldsymbol{l}_i - \boldsymbol{o}) \times (\boldsymbol{q}_i - \boldsymbol{o}) \bullet (\boldsymbol{c} - \boldsymbol{o}) = 0 \tag{2.3}$$

Since the angle θ between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be obtained from $\boldsymbol{ab} = ||\boldsymbol{a}||||\boldsymbol{b}||cos\theta$, condition 2) can be formalized as

$$(\boldsymbol{l}_i - \boldsymbol{q}_i) \bullet (\boldsymbol{q}_i - \boldsymbol{c})||\boldsymbol{o} - \boldsymbol{q}_i|| = (\boldsymbol{o} - \boldsymbol{q}_i) \bullet (\boldsymbol{q}_i - \boldsymbol{c})||\boldsymbol{l}_i - \boldsymbol{q}_i|| \tag{2.4}$$

Next consider a ray that comes from the pupil center $\boldsymbol{p}$, and reflects at point $\boldsymbol{r}$ on the surface of the cornea such that the refracted ray passes though $\boldsymbol{o}$, the nodal point of the camera, and intersect with the image plane at point $\boldsymbol{v}$. This can be formally expressed in parametric form once again as

$$\boldsymbol{r} = \boldsymbol{o} + k_r(\boldsymbol{o} - \boldsymbol{v}) \tag{2.5}$$

for some k_r and the condition that $\boldsymbol{r}$ is on the corneal surface is expressed as

$$||\boldsymbol{r} - \boldsymbol{c}|| = R. \tag{2.6}$$

The law of refraction states two conditions: 1) the incident ray, the refracted ray and the normal at the point of refraction are coplanar; 2) the angle of incidence, θ_1, and the angle of refraction, θ_2, satisfy Snells law (i.e., $n_1 sin(\theta_1) = n_2 sin(\theta_2)$), where n_1 and n_2 are the indices of refraction of mediums 1 and 2. Since vector $\boldsymbol{r} - \boldsymbol{c}$ is normal to the spherical surface at the point of refraction $\boldsymbol{r}$, condition 1) implies that points $\boldsymbol{p}$, $\boldsymbol{r}$, $\boldsymbol{c}$, and $\boldsymbol{o}$ are coplanar which can be formalized as

$$(\boldsymbol{r} - \boldsymbol{o}) \times (\boldsymbol{c} - \boldsymbol{o}) \bullet (\boldsymbol{p} - \boldsymbol{o}) = 0 \tag{2.7}$$

Since the angle between two vectors, θ, can be obtained from $||\boldsymbol{a} \times \boldsymbol{b}|| = ||\boldsymbol{a}||||\boldsymbol{b}||sin\theta$, condition 2) can be formalized as

$$\begin{gathered} n_1||(\boldsymbol{r}-\boldsymbol{c}) \times (\boldsymbol{p}-\boldsymbol{r})||||(\boldsymbol{o}-\boldsymbol{r})|| = \\ n_2||(\boldsymbol{r}-\boldsymbol{c}) \times (\boldsymbol{o}-\boldsymbol{r})||||(\boldsymbol{p}-\boldsymbol{r})|| \end{gathered} \tag{2.8}$$

where n_1 is index of refraction between between the aqueous humor and and cornea and n_2 is the index of refraction of air (≈1). Since the refractive index n_1 is quite small compared to the overall power of the lens system of the eye it is ignored. Finally, considering the distance K between the pupil center and the center of corneal curvature leads to

$$||\boldsymbol{p}-\boldsymbol{c}|| = K. \tag{2.9}$$

Since the optical axis of the eye passes through the pupil center $\boldsymbol{p}$ and the center of corneal curvature $\boldsymbol{c}$, if the above system of equations is solved for $\boldsymbol{c}$ and $\boldsymbol{p}$, the optical axis of the eye in space can be reconstructed as the line defined by these two points. To do so the eye parameters R and K have to be known. These are not easily measured directly and are subject specific so they are obtained though a one time calibration procedure.

For a single camera configuration, this gaze estimation model is theoretically insensitive to variation of both head orientation and position, referred to as *head pose.* Complete systems which utilize this model report 0.5° estimation accuracy on a wide array of subjects. A full analysis of the sources of errors and theoretical limits of estimating the point-of-gaze from a single camera with two corneal reflections is presented in [3].

This model represents the best-in-class remote non-contact gaze estimation model currently presented in the academic literature. It is both accurate and inherently tolerant of large head pose variations, significant advantages for a mobile eye tracker system. It is the foundation of the gaze estimation model presented in this thesis. Its only significant limitation is that it assumes a stationary eye-tracking system. In a mobile system, the device, camera, and infrared LEDs are held in a user's hand and moves relative to the user. One contribution of our work is the expansion of this model to include allow for movements of the eye-tracking system.

Most of the eye-tracking methods presented in this section, systems that use either natural light or infrared illumination, are two-stage systems. The systems require an initial computer vision stage to estimate the locations of eye/facial features, which get passed to the gaze estimation stage. The feature extraction front-end stage widely varies between eye-tracking systems, even when the same features are extracted. We discuss some of the methods and techniques that are used to localize eye features in the next section.

2.2 Image Processing and Feature Extraction

In the previous section, we discussed several methods to estimate the point-of-gaze. The accuracy of these methods, to varying degrees, is dependent on precisely estimating the location of specific eye features in the input image. In this section, we provide a brief overview of some techniques used to perform this feature extraction.

The focus of this overview is on algorithms to estimate the pupil/iris center location, as this is the most prominent facial region required for most eye-tracking techniques. The estimation of the pupil/iris center or pupil/iris boundary has traditionally been achieved with rule-based algorithms. More recently, machine learning algorithms have been used because they have an ability, with proper training data, to produce more generalizable and robust results.

2.2.1 Traditional Rule-Based Pupil/Iris Center Estimation Algorithms

There are hundreds of papers describing traditional rule-based algorithms to estimate the pupil or iris center. An extensive survey of such work can be found in [63], but we review only a few representative examples below.

Developers have optimized feature extraction techniques under many different sets of physical and operational constraints which is the reason there are many variants of these algorithms. Examples of physical constraints include the resolution of the camera, the processing speed of the computer, or the total illumination power of IR light sources, indoor vs. outdoor use, expected head pose variance, run-time processing speed requirements, or accuracy requirements. These wide variety of constraints impacted the success of any specific approach. Also, the system-specific nature of many techniques seems to dissuade many authors from publishing precise details when they might be of limited generalized value (understandably). Instead, these works opt to explain higher-level concepts about the approach and how to combine existing techniques within their systems. In this section, we outline a few relevant approaches and discuss their limitations.

A large number of gaze estimation techniques require only the detection and tracking of the iris or pupil. The pupil is often darker than its surroundings and *thresholding* approaches have been used [104, 105, 106] to detect points inside the pupil. These techniques determine an intensity threshold to segment the eye region into pupil/iris-sclera regions. These techniques have had greater adoption in infrared environments where the contrast between the pupil and iris is often better since the pupil does not reflect IR light. The contrast is worse for users who have a blue iris' since their irises are not as IR reflective as users with brown irises. These methods are very computationally efficient but are not robust to significant image variation during the use of the eye tracker and thus may not be ideal for the mobile system proposed in this work.

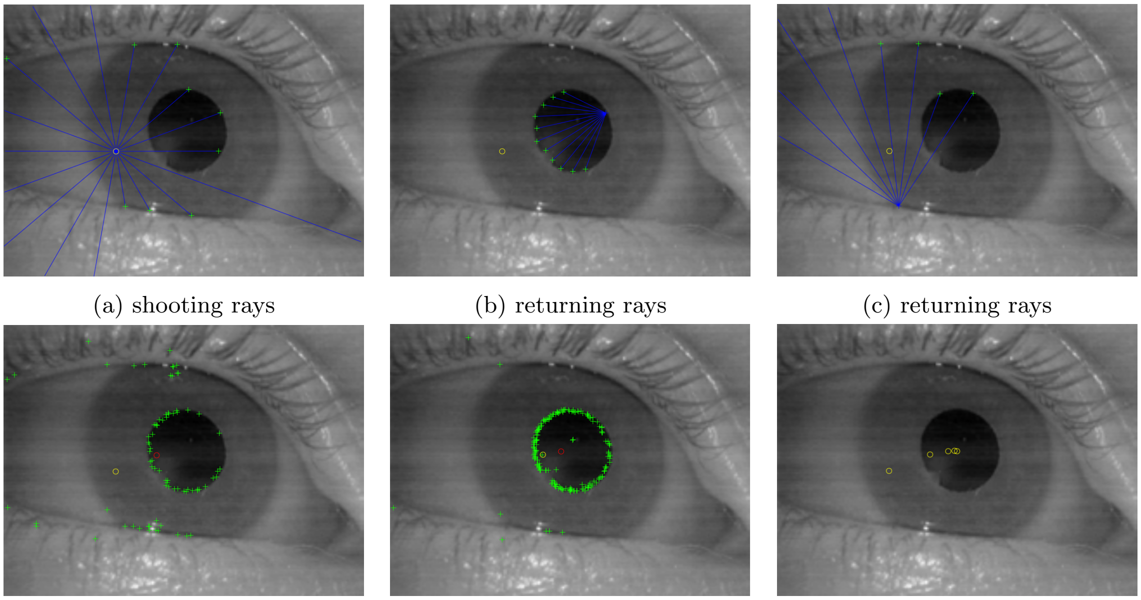
Many pupil/iris center extraction techniques model the boundary of these regions an ellipse and estimate the ellispe parameters either with voting methods [107, 108, 109, 110, 111, 112] or direct model fitting [113, 114, 5] on sets of pixels which are identified as part of the ellipse boundary.

In [107, 110] edge detection techniques are used that employ pure gradient intensity thresholds. This approach is brittle as many other regions of the images can share that same intensity gradient. Some works use the Hough transform to extract the limbus or pupil boundaries [109, 112] with limited success. The work in [108] uses the fact that the gradient field around the iris and pupil boundary *all* point outward away from the center of the iris and pupil. Heuristic rules then help filter out erroneous edge candidates. The method described in [111] extends this idea using isophote curvatures (curves on an illuminated surface that connects points of equal brightness). Rather than filtering out some potentially erroneous edges before voting [111] the weights in the voting process are based on the strength of the associated isophote curves. This method tends to mistake other curved features, such as the eye corners or eyebrows, for the pupil. This method is typically used only in a very constrained search space around the pupil (which requires that the system knows very precisely where the pupil is).

The algorithms described in [115, 113] were initially developed for pupil segmentation in iris scanning and identification applications but were then improved and used in the eye-tracking domain. This approach uses an optimization that finds the maximum of the curve integral of gradient magnitudes under an elliptical shape model. In other words, it finds an ellipse with the maximum intensity gradient orthogonal to the boundary of that ellipse. It was initially computationally expensive because the algorithm would exhaustively check every possible ellipse configuration. Non-exhaustive intelligent searching of the solution space improved performance over time [116]. The quality of its iris/pupil estimation results degrades in the presence of infrared corneal reflections in the eye region, which makes this only suitable for visible light eye images. The work presented in [114] also modeled the iris as an ellipse, but fitted the ellipse through an expectation-maximization and RANSAC [117] optimization scheme. That work introduces a likelihood model which examines a broader neighborhood around each proposed contour point and avoids using fixed feature parameters such as any single intensity level or gradient threshold. This method does very well at robustly estimating the iris ellipse (in both visible and IR light). Here, by robustness, we mean that it rarely makes significant estimation errors by falsely including, for example, an eye corner as part of the iris boundary. However, the precision, measured by the average distance between the estimate and ground truth, is as high as ±10% of the radius of the iris. This level of precision would negatively impact gaze estimation accuracy for many different gaze estimation models.

The Starburst [5] algorithm is another popular pupil estimation technique for low-cost infrared eye-tracking. The algorithm begins by removing corneal reflections via linear interpolation of pixel intensities on either side of the reflections. Then, given some initial seed location, rays are propagated in all directions, and gradients are computed only on those rays. When a computed gradient is above a user-defined threshold, the ray is is scattered backwards at many angles, as shown in Figure 2.5. Pupil boundary points are defined as the location where the reflected rays interact with a significant gradient. Then the average of the current set of boundary points is taken as a new seed point, at which point this process is repeated until the seed point stabilize. Some benefits of this approach are that it does not require a seed point inside the pupil and the ray-based approach significantly reduces the computational overhead of computing image gradients. It assumes, however, that a static threshold exists that defines the pupil/iris boundary all along the circumference. It also assumes that the corneal reflections are small compared to the size of the pupil so that removing them will not significantly distort the pupil boundary. Figure 2.6 illustrates two examples of images from a mobile infrared camera where Starburst would fail. Figure 2.6a illustrates a low contrast eye region example where there is no static gradient threshold which defines the pupil boundary. Figure 2.6b illustrates an eye region with a small pupil for which the removal of the corneal reflections would destroy the geometry of the pupil boundary. Both of the images Figure 2.6 came from the eye-tracking system used in this work.

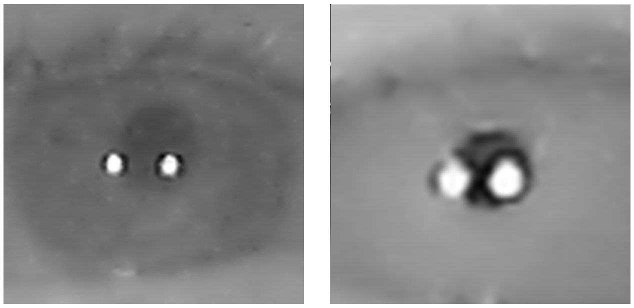
A technique that was presented in [6] analyzes all three color channels of an eye region rather than a greyscale version of the image (which is far more common). This method starts by generating an initial pupil estimate using a similar technique to [113]. Then an optimization process occurs that seeks a 3x3 matrix which transforms the RGB intensities at each pixel to a new color-mapped set of R'G'B' intensities. The matrix parameters are determined by an optimization that maximizes the sum of the gradients in each color channel across the seed pupil boundary. The algorithm then processes the new color-mapped image and produces a single edge map that has continuous contours around the edge of the pupil. Figure 2.7 illustrates the primary stages of this technique. The work was demonstrated on ultra-high-resolution eye images with significant contrast between the pupil and the iris (which is rare for images from visible light eye-tracking systems).



(a) shooting rays (b) returning rays (c) returning rays

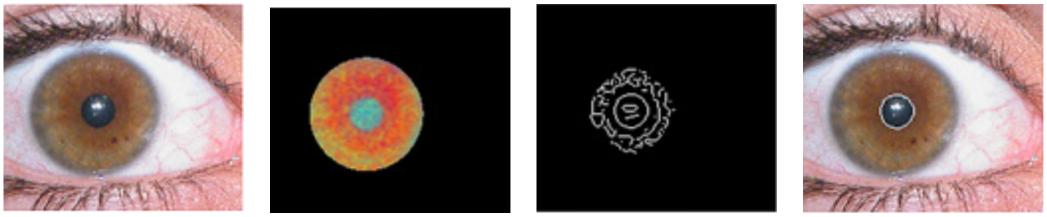
(d) candidate points (e) second iteration (f) convergence

Figure 2.5: Starbust algorithm stages [5]



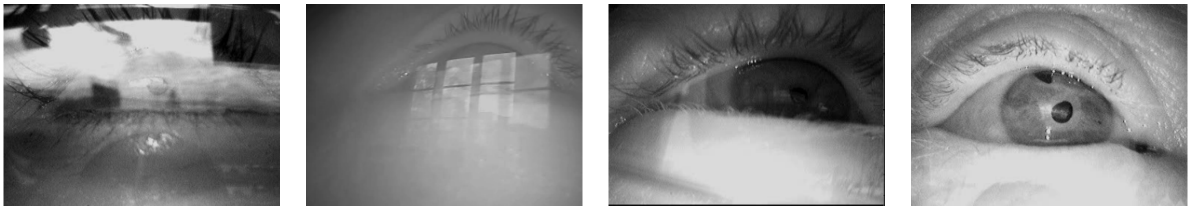
(a) low contrast eye (b) small high contrast eye

Figure 2.6: Eye region image variance



(a) original image (b) color mapped (c) edges (d) pupil segmentation

Figure 2.7: Colour mapping algorithm stages [6]



(a) reflections (b) reflections (c) dark pupil (d) philological anomaly

Figure 2.8: Example challenging outdoor real world eye regions from a head mounted camera

Among the most recent and powerful rule-based algorithms for pupil center detection are Ellipse Selection [118] (referred to as *ELSe*) and Pupil Reconstructor [119, 120] (referred to as *PuRe*). ELSe starts with simple Canny edges [121] and then uses a complex set of morphological operators to smooth and connect continuous edges before attempting to select the 'best' ellipse. The algorithm reverts to a simple convolution when it fails to find such an ellipse. This approximate alternative procedure involves convolving several down-sampled versions of the input image with two fixed convolution kernels and selecting the pupil center as the maximum location in the element-wise multiplication of two results. Similarly, PuRe [119] starts with edge detection and a series of morphological operators to clean and connect edge lines. Then each edge segment is scored based on the likelihood of it belonging to the pupil, and segments are removed if the score is below a threshold. Several local features, such as radial contrast, contribute to the scoring function. Finally, the remanding edge segment candidates are combined into every possible unique subset. The set which maximizes pupil boundary likelihood score defines the estimated pupil boundary. These two approaches estimated the pupil center to within 5 pixels with a 70% and 75% rate respectively on the same dataset.

These two approaches used datasets generated from infrared head-mounted camera systems in outdoor environments and include some very challenging cases. These specific datasets are becoming more commonly used in the pupil estimation literature precisely because of the number of challenging examples. For example Figure 2.8a and 2.8b show eye regions with strong interference from external light. Figure 2.8c and 2.8d show eye regions with very low pupil contrast and an unexpected pupil-like dark sport on the upper iris. Many eye regions in these datasets (including those shown in Figure 2.8a, 2.8b, and 2.8c) also exhibit large deviations between the optical axis of the eye and the optical axis of the camera. These types of images, especially those with strong reflections, are more difficult to process than images that we would expect in our work. Both PuRe and ELSe were built and designed to estimate pupil centers from the types of eye regions found in these datasets. They were designed and optimized for images with characteristics that are not similar to those that we expect from a smartphone.

These two approaches are not well-suited for our work for a few reasons. Both approaches were fundamentally optimized for pupil detection and not for precision (which is understandable given the difficulty of some of these examples). The number of images for which the predicted pupil center estimate was less than N pixels from the ground truth (often presented with N=5) defines success in these works. They do not attempt to minimize systematic bias from physiological features such as systematic occlusion of the pupil by the upper and lower eyelid. Secondly, by using eye regions from head-mounted cameras, these two approaches assume that the distance between the eye and the camera remains constant. Both algorithms use this assumption implicitly (combined with anatomical pupil sizes in the range of 3-8mm diameter). The algorithm parameters are tuned to evaluate the likelihood that some feature is part of the pupil boundary based on the overall curvature and size that one would expect. When these parameters are dramatically relaxed, as they must be when the camera can move freely in the user's hand , the performance of these approaches deteriorate.

The pupil estimation techniques discussed so far, model only simple features such as the iris and pupil. They have demonstrated some level of effectiveness but often are implicitly tailored for specific systems and operating conditions. These types of techniques can have difficulty capturing the variations and inter-person differences between users (e.g., other eye features such as eyelids, eye corners, and eyebrows). There is a class of more complex rule-based shape models which attempts to address these issues using deformable template-based methods or active shape models. These techniques attempt to model more components of the eye (such as the eyelid and eye corners) [122, 123, 124, 125, 126, 127, 128]. They, unfortunately, were demonstrated on eye regions with minimal eye occlusion and high contrast (especially around the boundary of the sclera). In the literature active shape models have not been used successfully on realistic real-world eye regions for eye-tracking and thus may not be a good fit for a mobile eye tracker. Rather than attempting to identify, extract, and model eye features explicitly, the alternative approach is to train a learning algorithm with labeled images and let it decide what features are essential. We discuss these approaches in the following section.

2.2.2 Appearance-Based / Machine Learning Pupil Estimation Methods

The most straightforward appearance-based techniques are template-based matching and interpolation between labeled ground truth images [129, 130]. More sophisticated approaches emerged which directly estimate facial landmarks (such as the pupil center) using general learning algorithms such as support vector machines [131], hidden Markov models [132] and Eigen analysis (principal component analysis on images) [133, 134]. These techniques significantly increased the robustness of landmark estimation in the presences of head pose variation and illumination changes (if the training dataset contained them).

The primary method for detecting facial features or estimating facial feature landmark locations in images is using convolutional neural networks (CNNs) [135, 135, 136, 137, 138, 139, 140]. Before a discussion of some of these techniques, we first give a brief description of the some of the common elements of a CNN image processing pipeline as we use one in a core part of this work.

Convolutional Neural Networks

Convolutional neural networks (CNNs) are artificial neural networks that are optimized to find spatial relationships in data. They are heavily used today in the field of computer vision [141, 142, 143, 144] for both classification and regression tasks. Our work also utilizes CNNs as part of our feature extraction process (discussed in Section 5.3). We preset the general technique here.

A *typical* CNN architecture for image processing consists of a pipeline of several types of computational layers. The computation layers consist of convolutional layers, pooling layers, batch normalization layers, dense/fully connected layers, and dropout layers. Other important elements of a CNN include weight initialization functions, optimization algorithm, loss functions, and activation functions. This list is not exhaustive, but it provides a good foundation for the discussion of CNN based pupil estimation methods that were presented in the literature. Each of these elements is discussed in more detail here:

- **Convolution Layer:** Convolution layers consist of a number of fixed-sized filter kernels which are convolved with the input image to generate output feature maps. This process is shown visually

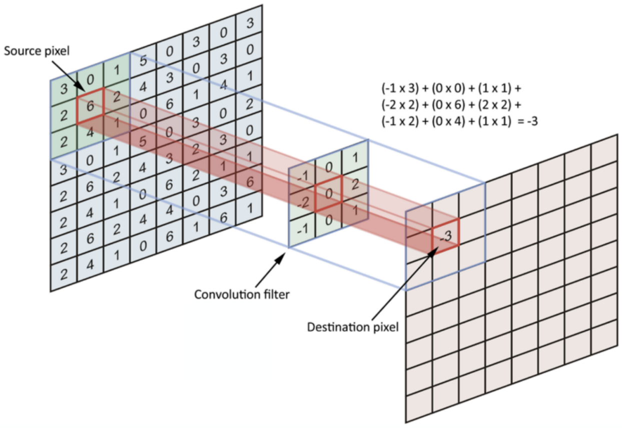

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 2.9: Convolution Layer

in Figure 2.9. The kernel weights and a single bias for each kernel are learned during the training of the network.

- **Pooling:** Pooling layers reduce the size of the associated feature maps, typically by either averaging neighboring elements together or selecting the maximum values within local windows [145]. Pooling decreases the required computation in downstream layers and can also increase network performance for some types of tasks by summarizing the presence of features.

- **Batch Normalization:** Batch normalization layers [146] are used to normalize output feature maps of hidden layers in the computation pipeline. Normalizing feature maps in between layers helps keep gradients in the objective function from being either too small which slows the training rate, or too large resulting in training failure (known as the exploding gradient problem [147]). It also has a slight regularization effect which can decrease overfitting and improves generalization.

- **Dense Layer:** Dense layers refer to fully connected traditional neural networks [148] in which there is no implied bias towards spatial locality. There is often one or more dense layers at the end of image processing CNN network pipeline. The function of these layers is to analyze the output high-level features generated by the last convolutional layer and learn how to combine those high-level features to generate the prediction.

- **Dropout Layer:** Dropout layers make the training process noisy by randomly ignoring some proportion of the previous layer's outputs [149]. Dropout aids in the generalization of the network by encouraging the network to learn a sparse representation. It also reduces over-fitting, especially when training on tasks with few training examples.

- **Activation Function:** An activation function is used to decide whether an output neuron should be activated or not as well as to introduce non-linearity into the output of a neuron. A network without any activation functions is essential just a linear regression model. Many different activation functions have been proposed for use in neural networks, some of the more popular ones include the sigmoid function [150], the rectified linear unit [151] (ReLU), noisy ReLU [152], and the tanh function [153].

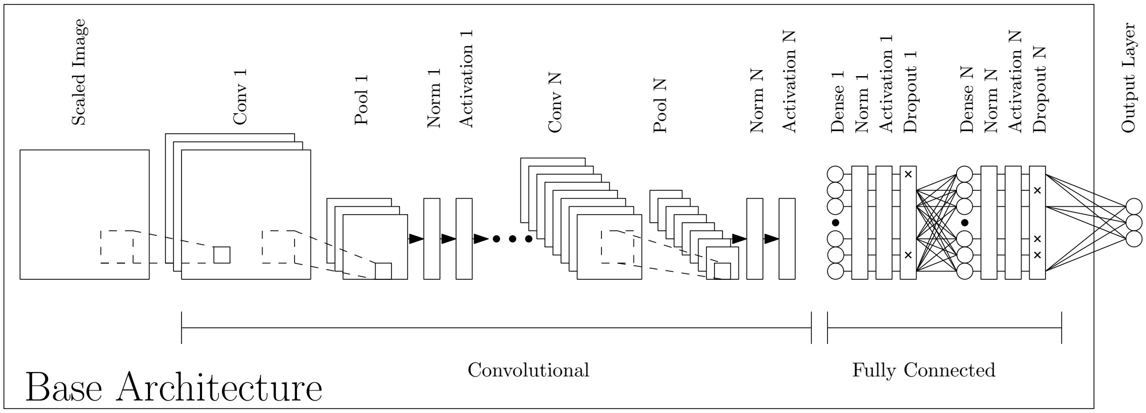


Figure 2.10: Base CNN Architecture

- **Optimization Algorithm:** The optimization algorithm defines how the training process should navigate the non-convex solution space in an attempt to minimize the loss function across a training batch. In the simplest case, one can choose to take a step in the direction of the negative gradient vector, known as *gradient decent* [154]. There are many other optimization algorithms; including Adagrad [155] or Adam [156], which automatically adjust the navigation procedure in response to the local shape of the optimization space.

- **Initialization Function:** An initialization function defines how to set the weights of a neural network before the first training step occurs. Common options include uniform random distribution [157], Gaussian [158], He [159] and Xavier [159]. Both He and Xavier introduce random distributions which depend on the specific size of each network layer. The weights are generated from distributions which vary with the square root of the number inputs weights to each layer.

The simple CNN image processing pipeline is some combination of the flow shown in Figure 2.10 and configured using the parameters described above. We refer to this as the *base architecture* in our work. This base architecture can, and has, been used successfully for both regression and classification tasks in many computer vision domains. It is used as a starting point for the discussions of CNN techniques that are used to detect the pupil center.

We outline three different CNN pupil detection approaches, Pupilnet [138], DeepEye [140], and domain-specific data augmentation [139] (DSDA). All three of these CNN systems were trained with near-IR head-mounted image data (similar to what the rule-based pupil detectors ELSe and PuRe used). Consequently, the same caveats apply in regards to the applicability of the datasets to the expected data set of images from the camera on the smartphone.

Pupilnet [138] is a two-stage positioning CNN system. Both CNN stages are quite small and uniform architecturally, consisting of a single 5x5 convolution layer, a single pooling layer, and a single fully connected output layer. The first stage is trained as a classifier to identify a sub-window that contains a pupil. It down-samples the 384x288 pixel images to 96x72 pixel and classifies each 24x24 pixel sub-window. The second CNN stage receives the sub-window with the highest probability of containing the pupil. The second stage is trained as a regression predictor to output the [x,y] location of the pupil center given a pixel sub-window in the original image. This work reports a 70% detection rate of pupil center estimates under five pixels. This result is comparable to ELSe and PuRe while being considerably less challenging/complicated from a development perspective.

Another CNN pupil detector, DeepEye [140], claims better performance than ELSe, PuRe and Pupilnet reporting a pupil estimation rate under five pixels for over 85% of the images in the same datasets. It consists of five convolutional layers and a custom Atrous Spatial Pyramid Pooling layer (ASPP). The ASPP layer contains several parallel Atrous layers, which are convolution layers in which the convolution kernels are themselves dilated. For example, consider an Atrous layer with 3x3 kernels and a dilation factor of four. In this example the nine total weights associated with each 3x3 kernel would be spread out and applied to images pixels in a 15x15 region, leaving four unused pixels between each weight. This parallel pyramid of Atrous layers is believed to help the network search more efficiently for features at different scales. Also, DeepEye uses residual shortcuts in each of the convolution blocks that append its inputs to its outputs. Residual shortcuts help mitigate deteriorated performance in networks with increased depth when there is not enough training data. Finally, DeepEye is trained for image segmentation rather than regression. This means the output is the set of pixels which belong to the pupil, not an [x,y] location. A post-processing ellipse-fitting step generates the final pupil center.

The final approach outlined here emphasizes the importance of large dataset sizes for generalization. In domain-specific data augmentation (DSDA) [139], a dataset of millions of eye regions was created with advanced data augmentation techniques. This dataset is much larger than other approaches which did hand labeling of tens of thousands of images. The authors captured high-quality head-mounted eye regions videos from 400 subjects in *ideal* conditions. The frames in these videos were at such high resolution and contrast that simple image processing techniques always produced very accurate pupil center estimates. The authors used these estimates to generate hundreds-of-thousands of training images. Then, generic data augmentation techniques were used to crop, scale, flip, blur, and over/underexpose these images. Also, domain-specific augmentation techniques were developed to add random corneal reflections and environmental reflections. A subject wearing a head-mounted system and sunglasses with an infrared mirror coating walked through various environments to generate environmental reflection videos. Frames from these reflection videos were then superimposed over the training dataset to further augment and artificially increase the overall dataset size.

The authors trained their CNN with an augmented dataset of millions of images. They used a deep CNN based on the architecture of the Inception V4 [160] object classification network. After the network was trained using an augmented dataset, they evaluated it with the exact head-mounted dataset used by DeepEye. The authors reported an 85% pupil detection rate on that dataset, which was very similar to DeepEye. The authors then evaluated both their network and the DeepEye network on a new dataset neither had seen during training. This dataset was obtained from different researchers and a different head-mounted system, in a different environment. In this scenario, the DeepEye network detection rate fell to 50% while the DSDA approach maintained an 85% detection rate. This result bolsters the claim that the larger augmented dataset produced with DSDA had aided in generalization.

Our work utilizes elements of all three of these approaches when building a feature extraction method appropriate the constraints of a genuinely mobile eye-tracking environment. Below we review the performance of the significant prior smartphone eye-tracking systems presented in the literature.

2.3 Mobile Eye Tracking Systems

The development of hand-held eye-tracking systems date back to the 2000s [161, 162] when research groups built fully-custom devices. More recently, work has begun to bring eye-tracking technology to widely available, and less expensive, mobile devices such as smartphones and tablets [42, 46, 47, 48, 43, 41, 44, 45]. The eye-tracking systems on these devices are based on the analysis of eye images from the devices' RGB cameras in which ambient visible light illuminates the face.

Earlier mobile eye-tracking systems employed simple geometric projections such as limbus back-projection [64, 18, 114] to estimate the PoG. Recall that limbus back projection suffers from poor accuracy when there are small deviations between the optical axis of the eye and the optical axis of the camera, which is nearly always the case in the use of a smartphone. The mobile eye-tracking system presented in [42], which uses limbus back projection, is negatively impacted by this effect. The work in [42] reported a gaze estimation error of 6° when the user's eye was at a relatively close distance of 20 centimeters from the camera. A similar technique presented [163] achieved only 15° gaze estimation accuracy. The high PoG estimation error of the limbus back projection systems led to projects which attempted to measure gaze gestures (i.e., the subject looked up/down/left/right) rather than PoG estimates [164]. While these methods can estimates such gestures accurately the lack gaze positions severely limits the type of eye-tracking applications which are suitable for these systems.

More advanced mobile eye-tracking systems are ScreenGlint [47] and GazeCapture [46].

2.3.1 The ScreenGlint Smartphone Eye Tracker

ScreenGlint [47] employs an eye-tracking model that uses the pupil center and the position of a single corneal reflection [91] to estimate the point-of-gaze. The system has several innovations. First, rather than using the pupil center, they used the iris center, as this feature is easier to track when there is no IR illumination. Next, the corneal reflection is not generated from an IR light source but rather from the visible light emitted from the device display. The presented system operated on a real device and achieved a PoG accuracy of 2.9° when the relative movements between the smartphone and the user's head were in the range of between 25-40cm distance from the subject.

2.3.2 The GazeCapture Smartphone Eye Tracker

GazeCapture [46] is a mobile eye-tracking system which uses an advanced CNN model. The input to the model is a picture of the subject's face, and the network produces a specific PoG estimate location. For such a system to be effective in a wide variety of environmental conditions, a significant amount of training data would need to be collected. The researchers dealt with this issue by creating an application for iPhone users. They recruited those users to spend a few minutes looking at training targets on display. This data collection methodology yielded tens of thousands of labeled images from nearly 1500 people that were used to train their CNN. Their system became quite robust to novel faces and created a calibration-free gaze estimation system that works with visible light. Variations in head pose and lighting between people were naturally incorporated into the training images which made the system robust to those variations. The system achieved a calibration-free gaze estimation error of 1.71cm and a calibrated estimation error of 1.34cm. The unsupervised nature of the data collection meant that the authors could not know the distance between the device and the subjects' heads for each sample. For this reason, the results are presented in absolute distance rather than the more traditional error unit for eye-tracking of degrees error.

To remedy this, we performed an analysis of a sample of the training images from their dataset. This analysis indicated that the users tended to position the device at a distance of 20-25cm (see Appendix B for a description of that approximation methodology). We can estimate an approximate distance using known human averages for inter-pupillary distance and the camera properties of the iPhone models in the market at the time of the data collection. Assuming a distance of 25cm, the gaze estimation errors for the GazeCapture would be about 3° and 4° with and without calibration respectively.

High quality and robust eye trackers such as ScreenGlint and GazeCapture that do not use artificial IR illumination can run on virtually all commercially available smartphones and have reported an approximate 3° gaze estimation error. If one compares this to reported to gaze estimation accuracy on the order of 0.5° for desktop infrared eye-tracking systems, it becomes a clear motivation to explore infrared-based mobile eye-tracking. The implementation and evaluation of an infrared mobile eye tracker is the purpose of our work. The next chapter describes the physical system that we used and the top-level software structure of the end-to-end system that we built.

Chapter 3

Top Level Mobile Eye Tracker

This chapter outline the main components of the mobile eye-tracking system developed in this work. These components include both the hardware platform and the top-level software, what they do, and how they communicate.

3.1 Platform

The first step in building our eye-tracking system is to select an operating-system and hardware platform. The hardware will need to include IR LEDs and IR camera, and could either be an integrated smartphone with these features built-in or an existing commodity smartphone with these features added via an external attachment. Our choice to explore both options limited our mobile operating system choice to exclusively Android [165] as Apple iPhone's neither contained the requisite infrared hardware or allowed third party external device support at the time.

We now discuss the mobile-hardware platforms that were used in this work, starting with what we call a *camera bar* that contains the IR LEDs and camera, that is attached to a commercial smartphone.

3.1.1 Infrared Camera Bar

An IR camera bar consists of one IR LED on either side of an IR camera housed in a rectangular package that connects to a computer or smartphone through a USB connection.

Developing an IR camera bar is one method to enable IR eye-tracking on commercially available smartphones. It has several advantages form a research prospective for this project. Firstly it can be built (and potentially rebuilt it) to our exact specifications with cameras, light sources and lens configurations with singular purpose of improving eye tracking. Another advantage is that a bar could be used interchangeably with many different smartphones. Thirdly, we can manufacture as many as needed for future experiments or to provide to other researchers to experiment with mobile eye tracking applications.

The disadvantages however come from the difficulty (at the time) in communicating between the smartphone and an external camera over USB on Android. The benefits of having control of the platform outweighed the difficulties and made this approach worth pursuing from a research perspective.

We built an IR camera bar and used it for several years during this project. However, an update to Google Android operating system security policy, made it infeasible to continue with this approach.

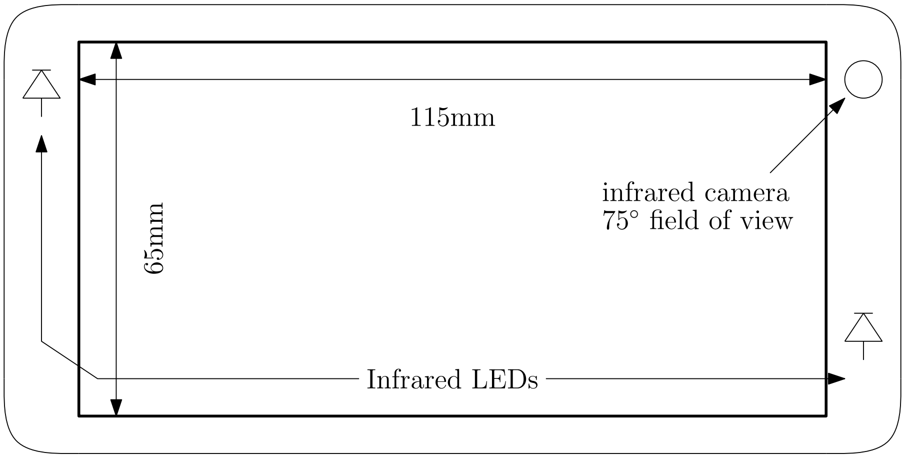


Figure 3.1: Prototype Infrared Mobile Device

As such, the IR camera bar was not functional during the core data collection phases of this work. As a result, the remainder of this thesis only presents quantitative eye-tracking data from the LittleBoy prototype discussed in the next section. For a detailed technical description of the IR camera bar, the software stack that was built to communicate with it, and how it was eventually deprecated see Appendix A.

3.1.2 LittleBoy Prototype Smartphone

Huawei [52] provided our project with an industrial-prototype smartphone which they gave the name *LittleBoy*. It has multiple IR LEDs and an IR-sensitive front-facing camera. The benefits of this prototype are the integration of the camera and light sources. The integrated camera can operate at a higher data rate than a USB2 camera. Additionally, the IR LEDs can draw more power and emit more light than a USB2 powered external camera bar, which enables higher contrast pupil regions at further distances from the user. Another advantage, in principle, is that the manufacturer would provide the camera and LED software drivers, and we could focus exclusively on the eye-tracking software. In practice, however, Huawei provided a camera and LED driver with limited functionality, which created problems that we needed to address. Another disadvantage of LittleBoy is that it is an unsupported prototype devices which means it does not receive operating systems updates from the manufacturer.

Figure 3.1 illustrates the physical geometry of LittleBoy. It consists of a five-inch display with multiple IR LEDs on opposite corners of the screen and an infrared camera. The camera resolution is 3840 x 2160 pixels, which is similar to the RGB front-facing cameras found on high-end Android devices in 2019 (even though Huawei produced it in 2015). The field of view of the lens is 75° which allows for a large range of device positions and orientations while keeping the user's head in the frame. The non-optical components of the device consist of a 1.2GHz quad-core ARM processor, 1GB of main memory, a 960x540 pixel screen, and operating system running Android version 5.0. The latter hardware specifications are of low quality in comparison to the hardware available in 2019 smartphones. The lower-performance CPU does not have any neural-network-acceleration hardware, now typical on smartphones made in 2019, and this limits the performance of the CNN front-end feature extractor described in Chapter 5.

Using LittleBoy as our platform we built a complete eye tracking system, the top-level modules of which will now be outlined.

3.2 Software

Once we obtained a suitable IR illumination and camera system, the primary eye-tracking functionality is defined in software. Here we discuss the top-level system modules of that software, describing what they do and how they interact. Figure 3.2 illustrates the flow of data between the various software modules. The system has six main components; 1) camera driver, 2) head tracker, 3) feature extractor, 4) gaze estimator, 5) calibrator, and 6) output filter. A brief description of each component is provided the sections below. A detailed overview of contributions made in the gaze estimator and feature extractor are then provided in Chapter 4 and 5 respectively.

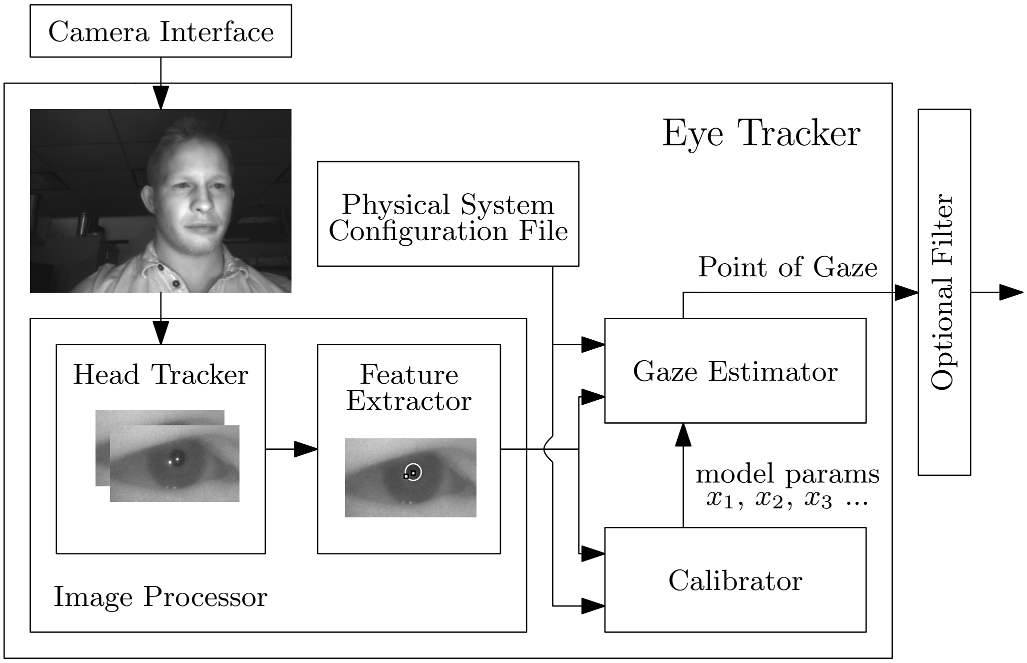
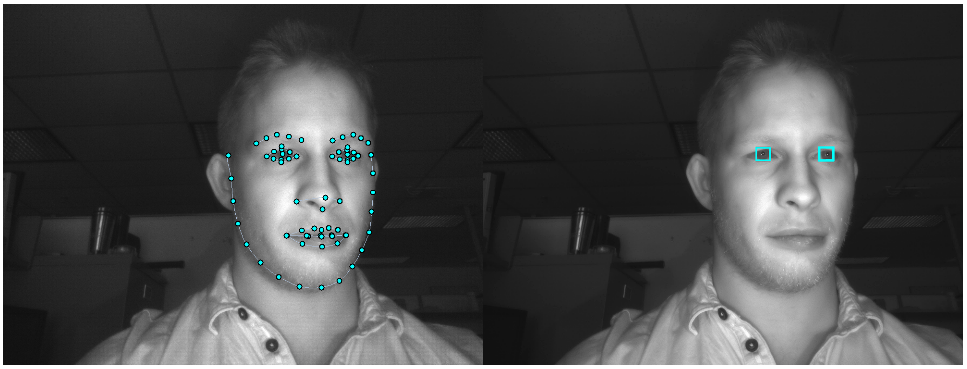


Figure 3.2: Eye Tracking System Components

3.2.1 Camera Driver

The camera driver software interacts with the camera to produce a stream of images. Huawei provided the camera driver for LittleBoy as a library precompiled within the android native development kit (NDK: as described in Appendix A.2). It leverages a robust fully-featured camera driver that exists in the Linux/Android kernel called Video for Linux 2 (V4l2 [166]). Typically user applications on commercial devices would not have the execution privileges required to call kernel drivers such as V4l2 directly. This access was possible because Huawei is the device manufacturer and can arbitrarily grant kernel-mode execution privileges on its own devices. Internally, the driver keeps a circular-FIFO queue of 10 frames. When a developer initiates a frame request, the queue transfers the oldest frame to a user-controlled frame buffer. A problem with the driver occurs when frames are requested at a slower rate than the maximum frame rate of the camera. In this case the steady-state stream of frames quickly becomes ten frame intervals delayed. This delay (300-400 ms) was very noticeable when one uses the eye-tracker. To reduce this delay we polled frames from the driver as soon as they were available on a continuous loop of a background thread. We kept both the current frame the eye tracker is processing and the newest



(a) Facial Feature Landmarks (b) Eye Sub Regions

Figure 3.3: Head Tracking Utility

frame provided by the driver in user-controlled memory. Whenever the system completed the processing of data in the current frame, it started to process the newest available frame. This procedure wasted a significant amount of memory bandwidth but allowed the system to always process the most recent frame.

There was a second issue with the driver, relating to the need for synchronization between the IR LEDs and the camera. Ideally, to save energy and reduce image smearing, the IR LEDs should be ‘turned on’ and illuminate the scene only while the camera captured a frame. Then the LEDs would turn off while the data for that frame was serially transferred into memory (for approximately 30ms). Unfortunately, there was no proper synchronization between the LEDs and frame capture, and so proper illumination would sporadically fail and cause one corneal reflection to appear either very dim or disappear altogether. Unfortunately, there was nothing we could do to address this and our image-processing pipeline needed to cope with this reality.

The next task of the eye-tracking system after it receives images from the cameras is to locate a subject’s head with a Head Tracker.

3.2.2 Head Tracker

The purpose of the head tracker is to locate the subject’s head over a wide range of head poses and then identify the rough location of the eye regions. Figure 3.3a illustrates example facial landmarks detected by a head tracker, and Figure 3.3b shows example estimates of eye regions.

Head-tracking and facial-feature estimation have become part of many standard mobile-software libraries, including Google’s vision library [167]. This library is available on most production Android devices. Using the built-in head-tracking libraries would have been the obvious choice for our eye-tracking system. However, LittleBoy does not have access to the Google mobile-services APIs, which contain these libraries.

Instead, we used a third-party head-tracking library from Visage [168]. It comes with pre-built native libraries compiled for Android with the NDK. Visage head tracking software was not trained with infrared images, which impacts facial landmark precision. It produces lower quality results on IR images when compared with typical gray-scale converted RGB images. However, because our use for it was only for determining a rough location for the eye regions it was suitable for our purposes.

Once the head tracker has located the eye regions, the next step is to determine the locations of the pupil center and corneal reflections in the image of the eye region.

3.2.3 Feature Extractor

The feature extractor is the next software stage in the eye-tracking system. It analyzes the eye windows produced by the head tracker and estimates the location of the pupil center and corneal reflections. Estimating these features accurately and with minimal variance is a vital step in the eye-tracking process.

A smartphone-based feature extractor presents some unique challenges. The appearance of eye regions can change dramatically based on how the user holds the device. A key contribution of this work is developing feature extraction algorithms that maintain high-quality in the presence of sizeable relative motion between the device and the head. This will be discussed in detail in Chapter 5. The feature extractor was implemented first as a custom rule-based algorithm (Section 5.2), and then as a hierarchy of convolutional neural networks(Section 5.3).

3.2.4 Gaze Estimator

The gaze estimator maps the eye features found by the feature extractor into a point-of-gaze location on the display of the smartphone. The method we use is a modified version of the 3D model that was developed by Guestrin and Eizenman [40]. This method uses data that describes the optical configuration of the eye-tracking system (e.g.,the position of the nodal point of the camera) and optical properties of the eye (e.g., horizontal angle between the visual and optical axis) to derive a system of equations that estimates the user's point of gaze.

The model first estimates the optical axis of the user's eye and then uses the offset between the optical and visual axes, specific to the user, to calculate the visual axis of the user's eye. A calibration procedure performed once per subject before using the system generates subject-specific parameters (e.g., the offset between the optical and visual axis) that are used in the estimation of the PoG. The model generates the location of the visual axis in a world coordinate system and its precision is not affected by head movements. However, the model is not designed under the assumption that the eye-tracking system *itself* (camera, screen, and light sources) would be moving with respect to that world coordinate system. This motion impacts the estimation of the visual axis. Extending the model to maintain accuracy during motion of the eye-tracking system is a contribution of our work and is described in Chapter 6.

3.2.5 Calibrator

The calibrator estimates the subject-specific properties of the user's eyes. Four parameters are determined during the calibration process. These are described in Table 3.1 and illustrated in Figure 2.4. These four parameters all represent physical geometric properties of a user's eye. They can be saved and reused later with future uses of the eye tracker by the same person once a calibration procedure is completed. Before the calibration procedure, it is possible to run the gaze estimator using 'typical' values that represent approximate human averages. Operating without calibration parameters significantly reduces gaze-estimation accuracy, but the system would remain functional.

The calibration routine consists of three stages: sample collection, outlier removal, and parameter optimization, which are all described below.

Sample Collection: In this stage, the calibration routine displays several gaze targets, one at a time, at known locations on the device display. The subject is instructed to gaze at each target while the

Name	Description	Units
R	Radius of corneal curvature	mm
K	Distance between the center of corneal curvature and the center of pupil	mm
Alpha	Horizontal angle between the optical axis and the visual axis	degrees
Beta	Vertical angle between the optical axis and the visual axis	degrees

Table 3.1: Calibration Model Parameters

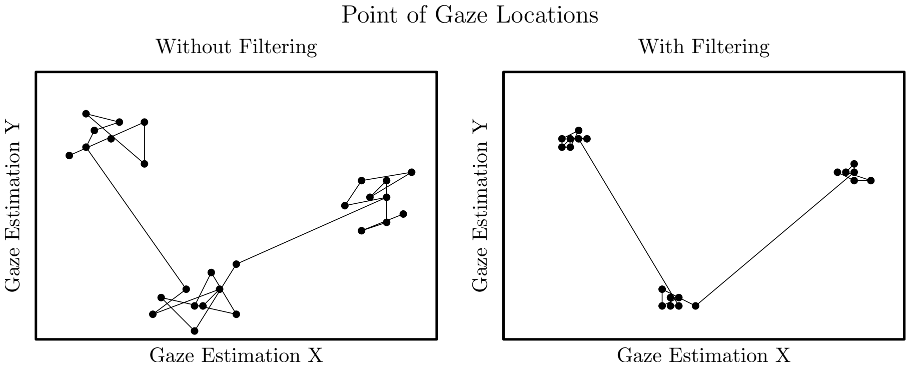


Figure 3.4: Illustration of Output Filtering

feature extractor estimates the pupil center and corneal reflection locations in at least 50 frames. Once completed, the calibration target moves to the next location. Our routine uses five targets, although only 2 are strictly necessary.

Outlier Removal: This stage identifies which eye features, collected in the previous stage, are inaccurate and should be discarded. The first step is to set the four subject-specific eye parameters to average human values. Then for each of the eye features collected in the previous stage, the gaze estimator generates a point-of-gaze. The distribution of the gaze estimates associated with each unique calibration target is analyzed. We classify as outliers as any gaze estimates which are located two or more standard deviations from the mean location of each target. Finally, we remove all eye features associated with outlier gaze estimates from consideration for the remainder of the calibration procedure.

Parameter Optimization: The final stage estimates the four eye parameters. A non-convex optimization process attempts to minimize the sum of the squared Euclidean distances between calibration targets and the PoG locations. Thus, the calibration computation involves nested non-convex numerical optimizations, since the gaze estimation procedure itself is a non-convex optimization. On the Littleboy prototype system, with a relatively old ARM processor, the computational component of the calibration procedure requires 250ms to complete.

3.2.6 Output Filter

The filtering module reduces PoG estimation noise when a user is fixating at the same location for many frames. Figure 3.4 illustrates an example of this effect.

We use a varying-length running-average filter for some experiments in this work. Increasing the length of the running average reduces the variance of successive gaze estimates. Filtering of this type comes at the cost of additional latency between fixation changes and the eye tracker updated the PoG to the new location. However, to analyze the quantitative performance of an eye-tracking system, this

Param	Description	Units
display size	physical dimensions of the device display	mm
display resolution	Operating resolution of the display	pixels
camera focal length	distance between the nodal point of the lens and image sensor when subject is in focus	mm
camera resolution	resolution of the camera sensor	pixels
camera principal point	The intersection of the optical axis of the camera lens and the image sensor, typically half of the camera resolution in both the horizontal and vertical axis	pixels
camera location	X,Y,Z location of the principal point	mm
camera field of view	total angle of light directed by the lens to the image sensor	degrees
camera pixel pitch	physical size of each pixel on the image sensor, which are assumed to be square	mm
camera orientation	rotation about the X,Y,Z axis, for a typical mobile device with a front facing camera mounted in the display plane there will only be rotation around the axis orthoginal to the display of this will be 0°, 90°, 180°, or 270°	degrees
infrared LED locations	X, Y and Z locations of the two infrared light sources	mm

Table 3.2: Required physical system Parameters

type of filtering would not be applied unless explicitly stated in the experimental procedure.

3.2.7 Physical-System Configuration

The final component of our eye tracker is a description of the properties of the physical system. For completeness this list is provided in Table 3.2. It includes properties such as the geometry of the display, light sources, and camera as well as the camera's intrinsic parameters. Each smartphone model (or IR camera bar) would have a different set of physical properties, and a would require separate configuration file containing the parameters found in Table 3.2. Each item in Table 3.2 represents a way in which our mobile eye-tracking system could physically change while the gaze estimation software would remain unchanged.

The next chapter discusses the details of the back-end gaze-estimation model improvements that we developed, and these improvements are one of the main contributions this work.

Chapter 4

Relative Roll Compensation for 3D Direct Model Gaze Estimation

This chapter presents a novel model-based method for the estimation of the PoG on displays of mobile devices that takes into account the motion of the device. Since a mobile device can be moved freely in the hands of subjects, several issues arise in PoG estimation on mobile devices that are not as present in stationary desktop systems. The first issue is associated with the need to use more than one coordinate system to describe all the parameters of the gaze estimation model. Desktop systems measure all model parameters (camera and light sources locations, eye-parameters, and display parameters) in one fixed world coordinate system. In an eye-tracking system that is free to move, some model parameters, such as the camera and light source locations, continuously change with the movements of the device. Other model parameters, such as the subject's eye parameters, remain fixed. Therefore, for PoG estimation on mobile devices, model parameters that are naturally in different coordinate systems must be transformed into a single coordinate system. The second issue that arises relates to the common assumption in gaze estimation models is that the relative roll angle (R-Roll) between the eye-tracking system (which comprises the camera, the light sources, and the display) and the subjects eyes is approximately 0° [40]. This assumption is reasonable for desktop eye-tracking systems where the head is approximately erect, and the system is stationary on a table. It does not hold for hand-held eye-tracking systems as the handheld device can easily be rotated by 45 degrees and if unaccounted for will result in significant systematic increase in gaze estimation error.

We extend the model-based gaze estimation approach in [40] to address the issues outlined above. The theory provided here applies to any hand-held system that may experience R-Roll and computes the PoG using a reconstruction of the optical and visual axis of the eye. The objectives of this chapter are two-fold. First, it is to describe a novel method that enables the use of accurate model-based estimation of the PoG on mobile devices. Second, to provide experimental data that demonstrates the importance of the modified model for mobile devices.

4.1 Mathematical Model

We refer the original PoG estimation model, designed for desktop systems, used in this work [40] as the *prior* model. The *prior* model estimates the PoG by determining the intersection of the visual axis (a

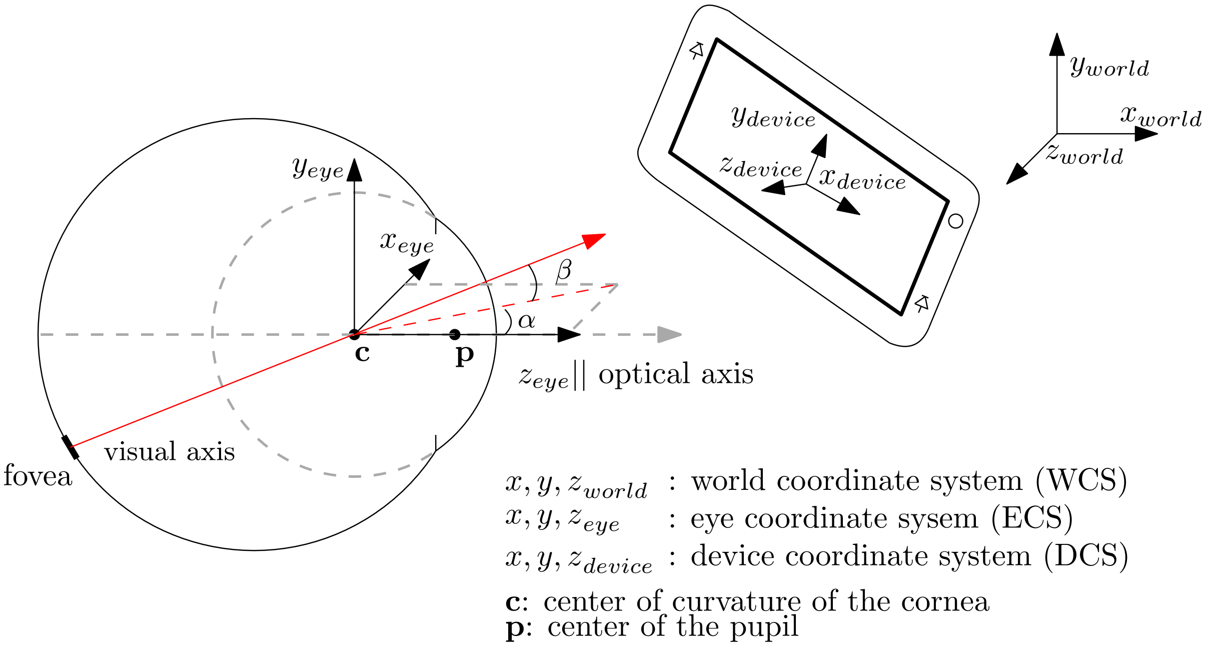

y_{eye}
x_{eye}
β
α
c
p
$z_{eye}||$ optical axis
fovea
visual axis
y_{device}
z_{device}
x_{device}
y_{world}
x_{world}
z_{world}
x, y, z_{world} : world coordinate system (WCS)
x, y, z_{eye} : eye coordinate sysem (ECS)
x, y, z_{device} : device coordinate system (DCS)
c: center of curvature of the cornea
p: center of the pupil

Figure 4.1: Mobile Eye Tacking System

line that passes through the pupil center and highest acuity region of the retina) with the stationary display. It estimates the visual axis, and all intermediate parameters, in a right-handed 3D-Cartesian world coordinate system (WCS) that is rigidly attached to the world. The x-axis and y-axis of the WCS are parallel to the rows and the columns (respectively) of the computer display. The z-axis is perpendicular to the display, pointing towards the subject. The *prior* model first estimates the direction of the optical axis of the eye in the WCS. The optical axis is defined as a line connecting the center of curvature of the cornea, **c**, and the center of the pupil, **p**. Where the locations of **c** and **p** are estimated using the formulation described in Section 2.1.2. Figure 4.1 presents a visualization of these features. As derived in the *prior* model, the equation for the direction of the optical axis, $\boldsymbol{\omega}$, is given by:

$$\boldsymbol{\omega} \equiv \frac{\mathbf{p}-\mathbf{c}}{\|\mathbf{p}-\mathbf{c}\|} = \begin{bmatrix} sin(\theta_{eye})cos(\phi_{eye}) \\ sin(\phi_{eye}) \\ -cos(\theta_{eye})cos(\phi_{eye}) \end{bmatrix} \tag{4.1}$$

where **p** and **c** are measured in the WCS and θ_{eye} and ϕ_{eye} are the horizontal (yaw) and vertical (pitch) angles of the eye (not shown in Figure 4.1) with respect to the WCS. The pupil center, **p**, and the center of curvature of the cornea, **c**, are estimated by the prior model using the coordinates of the pupil center and the virtual images of the light sources that illuminate the eye (corneal reflections) in images from the camera.

Next, the visual axis of the eye is estimated using the fixed angular separation from the optical axis. The angular separation is subject-specific but in the eye-coordinate system it does not change with eye movements. The visual axis of the eye, ν, also illustrated in Figure 4.1, is defined by a line connecting the center of curvature of the cornea, **c**, and the fovea, the highest acuity region of the retina. In the eye-coordinate system the visual axis the visual axis has fixed horizontal and vertical angular offsets (α_{eye}, β_{eye}) from the optical axis. These offsets are subject-specific which the prior model estimates with respect to the horizontal and vertical axes of the WCS during a calibration procedure. The equation for the direction of the visual axis, $\boldsymbol{\nu}$, in the prior model is given by:

$$\boldsymbol{\nu} = \begin{bmatrix} sin(\theta_{eye} + \alpha_{eye})cos(\phi_{eye} + \beta_{eye}) \\ sin(\phi_{eye} + \beta_{eye}) \\ -cos(\theta_{eye} + \alpha_{eye})cos(\phi_{eye} + \beta_{eye}) \end{bmatrix} \tag{4.2}$$

For a mobile eye-tracking system, the free movement of the device prevents the estimation of parameters in Equation 4.1 in the WCS. Since we cannot estimate the position of the device in the WCS, we instead estimate parameters with respect to the device itself. For this reason, we introduce a device coordinate system (DCS) that is a right-handed 3D-Cartesian coordinate system rigidly attached to the mobile device. The x_{device} and y_{device} unit vectors are parallel to the rows and columns of the mobile display, the xy-plane is coincident with the display plane, and the z_{device} axis is normal to the display pointing towards the subject. In the notation used going forward in this chapter, a superscript is used to denote the coordinate system of a parameter. For example, the location of the pupil with respect to the DCS is denoted $\mathbf{p}^{DCS}$.

If we estimate both the pupil center ($\mathbf{p}^{DCS}$) and the center of curvature of the cornea with ($\mathbf{c}^{DCS}$), with respect to the DCS then Equation 4.1 describes the direction of the optical axis in that coordinate system. The prior model can estimate $\mathbf{p}^{DCS}$ and $\mathbf{c}^{DCS}$ if the position and orientation of the camera and IR LEDs are also provided in the DCS. To determine the PoG on a hand-held device (i.e., the intersection of the visual axis of the eye with the screen), a unit vector in the direction of the visual axis must also be calculated in the DCS, $\boldsymbol{\nu}^{DCS}$. This requires that the model estimates the subject-specific eye parameters (α_{eye}^{DCS} and β_{eye}^{DCS}) in the DCS. However, α_{eye}^{DCS} and β_{eye}^{DCS} are estimated in the WCS and have to be transformed to the DCS coordinate system before they can be used by the model (one cannot use Equation 4.2 from the prior model to estimate the direction of the visual axis).

To address this issue we introduce an eye coordinate system (ECS) that is rigidly attached to the eye of the subject as can be seen Figure 4.1. The ECS is a right-handed 3D-Cartesian coordinate system whose axes are labeled x_{eye}, y_{eye}, z_{eye}. The z_{eye}-axis is coincident with the optical axis of the eye, pointing forwards out from the front of the head. In the ECS, the unit vector in the direction of the visual axis, $\boldsymbol{\nu}^{ECS}$, does not change with rotations or translations of the eye, head or device. Therefore, α_{eye}^{ECS} and β_{eye}^{ECS}, are constant. Using this coordinate system, $\boldsymbol{\nu}^{ECS}$ is written as:

$$\boldsymbol{\nu}^{\boldsymbol{ECS}} = \begin{bmatrix} sin(\alpha_{eye}^{ECS})cos(\beta_{eye}^{ECS}) \\ sin(\beta_{eye}^{ECS}) \\ cos(\alpha_{eye}^{ECS})cos(\beta_{eye}^{ECS}) \end{bmatrix} \tag{4.3}$$

where α_{eye}^{ECS} and β_{eye}^{ECS} are the horizontal and vertical angular offsets between the optical and visual axes with respect to the eye itself. Using equation 4.3 we can formally write:

$$\boldsymbol{\nu} \equiv \boldsymbol{\nu}^{DCS} = \mathbf{R}_{ECS}^{DCS} * \boldsymbol{\nu}^{ECS} \tag{4.4}$$

where $\mathbf{R}_{ECS}^{DCS}$ is the rotation matrix of the ECS with respect to the DCS and can be expressed as

$$\mathbf{R}_{ECS}^{DCS} = \begin{bmatrix} \mathbf{x}_{eye}^{DCS} & \mathbf{y}_{eye}^{DCS} & \mathbf{z}_{eye}^{DCS} \end{bmatrix}. \tag{4.5}$$

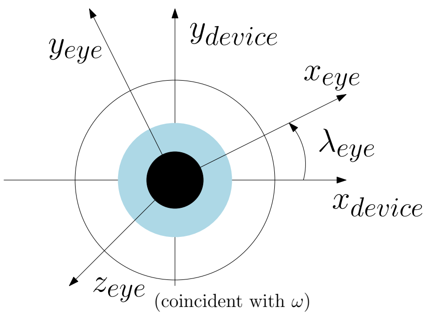


Figure 4.2: Illustration of λ_{eye} translation from DCS to ECS

The notation $\mathbf{x}_{eye}^{DCS}$ corresponds to a unit vector in the direction of the x_{eye}-axis but described in the device coordinate system. Note that by definition the unit vector in the direction of z_{eye}-axis of the ECS with respect to the DCS is the optical axis:

$$\mathbf{z}_{eye}^{DCS} = \boldsymbol{\omega}^{DCS}. \tag{4.6}$$

Recall that the R-Roll is the rotation angle of the x_{eye}-axis and y_{eye}-axis with respect to the DCS. We denote this parameter λ_{eye}, and illustrate it in Figure 4.2. If λ_{eye} is assumed to be 0° (this assumption will be relaxed later on in the derivation) then a unit vector in the direction of the $\mathbf{x}_{eye}^{DCS}$-axis is:

$$\mathbf{x}_{eye,\lambda=0}^{DCS} = \frac{\mathbf{y}_{device}^{DCS} \times \boldsymbol{\omega}^{DCS}}{\|\mathbf{y}_{device}^{DCS} \times \boldsymbol{\omega}^{DCS}\|} \tag{4.7}$$

where

$$\mathbf{y}_{device}^{DCS} \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.8}$$

is the unit vector in the direction of the y-axis of the DCS. Next, a unit vector in the direction of the y-axis of the ECS with respect to the DCS for $\lambda_{eye} = 0$ is defined as:

$$\mathbf{y}_{eye,\lambda=0}^{DCS} = \mathbf{z}_{eye}^{DCS} \times \mathbf{x}_{eye,\lambda=0}^{DCS}. \tag{4.9}$$

If we now relax the assumption that λ_{eye} is 0° and set it to the actual value of the R-Roll angle between the DCS and the ECS, the direction of the unit vectors in the rotation matrix of the ECS with respect to the DCS can be calculated by:

$$\mathbf{x}_{eye}^{DCS} = cos(\lambda_{eye})\mathbf{x}_{eye,\lambda=0}^{DCS} + sin(\lambda_{eye})\mathbf{y}_{eye,\lambda=0}^{DCS} \tag{4.10}$$

and,

$$\mathbf{y}_{eye}^{DCS} = -sin(\lambda_{eye})\mathbf{x}_{eye,\lambda=0}^{DCS} + cos(\lambda_{eye})\mathbf{y}_{eye,\lambda=0}^{DCS} \tag{4.11}$$

In summary, the method to estimate the PoG on a mobile device first estimates the direction of the optical axis of the eye in the device coordinate system using (Equation 4.1). Then, using values for α_{eye}^{ECS}, β_{eye}^{ECS}, λ_{eye} and eqs. (4.3) to (4.11) an estimate of the direction of the visual axis in the DCS is determined. Finally, the intersection of a vector aligned with the visual axis and the display, $z^{DCS} = 0$, is calculated, providing the PoG estimate on the display of the mobile device.

4.2 Quantifying the effects of relative roll on PoG estimation

In this section we derive an expression for the difference between the PoG estimates when $\lambda_{eye} = 0°$ (using eqs. (4.1) and (4.3) to (4.9)) and when λ_{eye} is set to the R-Roll (using eqs. (4.1) and (4.3) to (4.11)). The purpose of this derivation is to determine effects of the R-Roll angle on the estimation of the PoG (if unaccounted for) and to determine the expected magnitude of these effects.

Let $\boldsymbol{\nu}_0^{DCS}$ be the direction of the visual axis when λ_{eye} is artificially assumed to be 0° and $\boldsymbol{\nu}^{DCS}$ be the direction of visual axis when λ_{eye} = R-Roll. From Section 4.1 we have

$$\boldsymbol{\nu}_0^{DCS} = \mathbf{R}_{ECS}^{DCS}(\lambda_{eye} = 0°)\boldsymbol{\nu}^{ECS} \tag{4.12}$$

and,

$$\boldsymbol{\nu}^{DCS} = \mathbf{R}_{ECS}^{DCS}(\lambda_{eye})\boldsymbol{\nu}^{ECS} \tag{4.13}$$

where

$$\mathbf{R}_{ECS}^{DCS}(\lambda_{eye} = 0°) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.14}$$

and,

$$\mathbf{R}_{ECS}^{DCS}(\lambda_{eye}) = \begin{bmatrix} cos(\lambda_{eye}) & -sin(\lambda_{eye}) & 0 \\ sin(\lambda_{eye}) & cos(\lambda_{eye}) & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.15}$$

A key metric that indicates the effect of the new model is the angular difference between $\boldsymbol{\nu}^{DCS}$ and $\boldsymbol{\nu}_0^{DCS}$. The larger this angular separation the greater gaze estimation error we would expected when estimating the PoG without compensating for R-Roll. We will denote this angular separation as δ, and it can be obtained from:

$$\boldsymbol{\nu}^{DCS} \cdot \boldsymbol{\nu}_0^{DCS} = |\boldsymbol{\nu}^{DCS}||\boldsymbol{\nu}_0^{DCS}|cos(\delta). \tag{4.16}$$

Substituting eqs. (4.12) and (4.13), into 4.16 and using eq. (4.3) (noting that both $\boldsymbol{\nu}^{DCS}$ and $\boldsymbol{\nu}_0^{DCS}$ are unit vectors), yields δ as a function of λ_{eye}, α_{eye}^{ECS}, β_{eye}^{ECS}.

$$\begin{aligned} \delta = cos^{-1}\big[&cos(\lambda_{eye})\big[sin^2(\alpha_{eye}^{ECS})cos^2(\beta_{eye}^{ECS}) + sin^2(\beta_{eye}^{ECS})\big] \\ &+ cos^2(\alpha_{eye}^{ECS})cos^2(\beta_{eye}^{ECS})\big]. \end{aligned} \tag{4.17}$$

Figure 4.3 shows the expected difference between the directions of the visual axes, δ, when λ_{eye} is set to 0° (the assumption of the prior model) and when λ_{eye} is properly set to the actual R-Roll. In Figure 4.3, λ_{eye} changes from 0° to 180°. The four curves in Figure 4.3 are for four values of α_{eye}^{ECS} and β_{eye}^{ECS}. These values were selected to span the full range of expected angular offsets between the optical and visual axes in adults. The curve for $\alpha_{eye}^{ECS} = 3.0°$ and $\beta_{eye}^{ECS} = 1.5°$ shows the expected effects of the R-Roll angle on the estimated direction of the visual axis for an average adult human eye. For an average eye, during an orientation change of the mobile device from landscape to portrait ($\lambda_{eye} = 90°$) the value of δ is 4.7°.

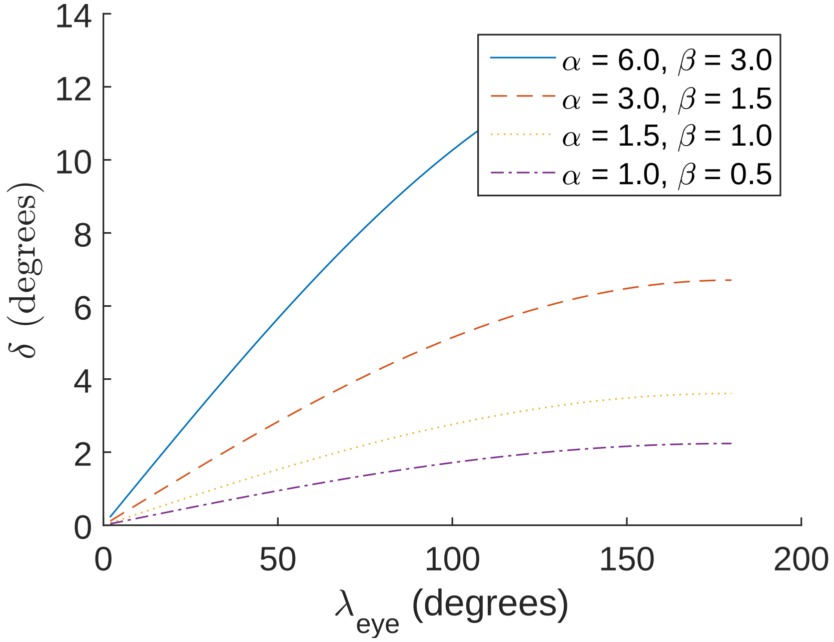


Figure 4.3: Expected estimation error (δ) as a function of the R-Roll angle (λ_{eye}) for four different angular offsets (α and β) between the optical and visual axes.

If the eye is at a distance of 30cm from the display this orientation change would result in a PoG estimation error of approximately 2.5cm, or more than one third the display-width of a typical smart phone. Figure 4.3 also shows that individuals with larger offsets between the optical and visual axes are expected to have larger differences between the directions of their estimated visual axis (δ) due to R-Roll. Moreover, the change in δ is approximately linear for a large range of R-Roll angle values, λ_{eye}. The relatively modest slopes of the lines in Figure 4.3 (1/50 to 1/11), indicate that the sensitivity of δ

to errors in the estimation of λ_{eye} is small.

Estimates generated by the head tracker are used to a) determine the rough position of regions in the face that include the eyes and b) in estimating the compensation for the role angle. For (a) the regions are so large that the performance of the head-tracker will not affect the accuracy of the gaze estimation. For the compensation of errors due to roll. For an average eye, a 1° error in the estimation of head roll (this will be translated to 1° error in λ_{eye}) will affect the estimation of the direction of the visual axis by only 0.04°. Therefore small errors in the estimation of λ_{eye}, through the use of a head tracker, can slightly increase the average gaze estimation error when the true R-Roll is zero. However if the magnitude of the relative-roll increases to, for example, to 45° the overall gaze error will be improved by nearly 2° (as shown in Figure 4.3).

The theoretical analysis presented in this section suggests that not accounting for R-Roll in mobile eye trackers can significantly increase PoG estimation errors. In addition, it suggests that PoG estimates are insensitive to noise in the estimation o the R-Roll value itself. This property is essential because the practical methods for estimating R-Roll (discussed in Section 4.5) are noisy. Before discussing those methods, the following sections discuss experiments that were used to validate the model predictions.

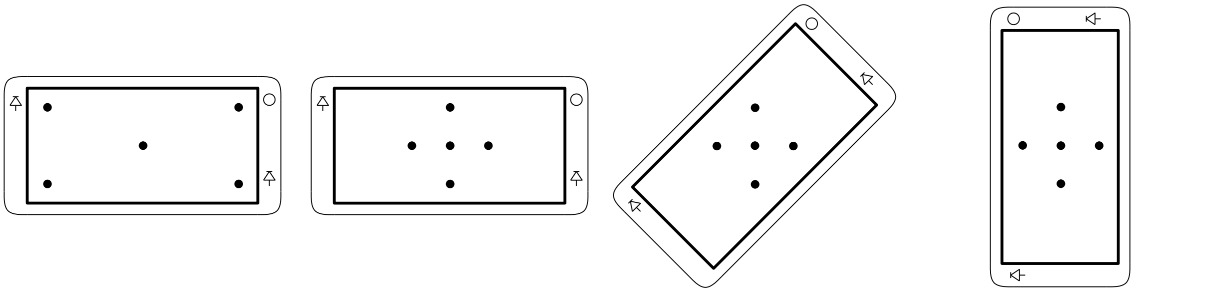
4.3 Experimental Procedure

We conducted a study with four subjects to validate the predictions of the model-based gaze estimation method in the presence of R-Roll that was developed in Section 4.1. The four subjects looked at targets on the display of a mobile device while the R-Roll angle was set to one of three angles, 0°, 45°, and 90°. The mobile device that we used in the experiments was LittleBoy, the prototype that was provided by Huawei (Figure 3.1), and is described in Section 3.1.2.

Recall that our eye-tracking system has the following components: a head and facial feature tracker, a feature extractor, and a gaze estimation model. The feature extractor that we use for this experiment is the rule-based version described later in Section 5.2. The details of the feature extractor are not yet relevant as the experiment is designed to explicitly minimize the impact of the feature extractor on the results. Our goal is not to evaluate the performance of the eye-tracker but rather to validate the effect of R-Roll on accuracy of gaze estimation. We present a complete evaluation of the eye-tracker in realistic scenarios in Chapter 6. During this experiment, we placed the subject's head on a stationary chinrest and the smartphone in a firm rotating stand 30 centimetres away. The chinrest and device stand act to minimize sources of errors that are associated with head or device movements. Using a chinrest in this way allows us to isolate the change in gaze estimates due only to R-Roll between the subject and the device.

We collected four videos from each of the four subjects (16 videos in total) while setting the R-Roll between the device and subject to a known angle – by rotating the device while leaving the subject's head upright. During the recording of each video, subjects were instructed to look at five targets on display for 50 frames each. The first video was used to determine subject-specific eye parameters that included α_{eye}^{ECS} and β_{eye}^{ECS}. These parameters, along with the amount of R-Roll (fixed for each experiment), and the method described in Section 4.1 were used to estimate the PoGs in the three following videos. Recall that the R-Roll angles for these three videos are 0°, 45° and 90° from the orientation during the calibration video as seen in Figure 4.4.

We estimate the PoG for each frame using one of two methods: Method 1 sets λ_{eye} to 0°, for all test



Calibration Targets Estimation Targets 0° Estimation Targets 45° Estimation Targets 90°

Figure 4.4: Calibration and Estimation Targets

		Average Gaze Error (mm)			Average Gaze Error (degrees)		
Subject	Method	R-Roll = 0°	R-Roll = 45°	R-Roll = 90°	R-Roll = 0°	R-Roll = 45°	R-Roll = 90°
01	2	5.07	4.7	5.05	0.96	0.89	0.96
	1	4.91	11.11	15.96	0.93	2.12	2.75
02	2	3.73	4.55	5.18	0.71	0.86	0.98
	1	3.52	9.08	14.39	0.67	1.73	2.74
03	2	3.15	4.77	4.12	0.60	0.91	0.78
	1	3.14	11.91	18.42	0.59	2.27	3.51
04	2	7.23	5.94	10.05	1.38	1.13	1.91
	1	7.31	15.44	24.66	1.39	2.94	4.69
Average	2	4.80	4.99	6.10	0.92	0.95	1.16
	1	4.72	11.88	18.36	0.90	2.26	3.50

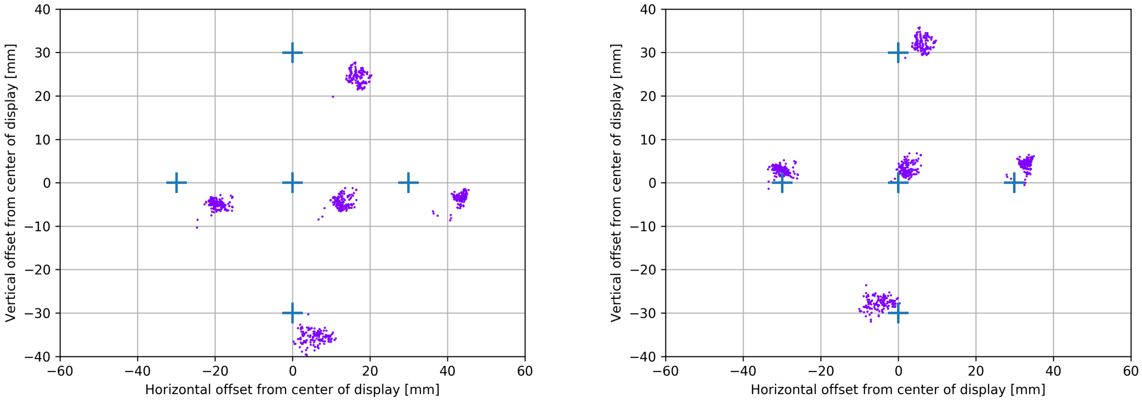
Table 4.1: Gaze estimation errors for R-Roll angles of 0°, 45° and 90°. In Method 1 λ_{eye} set to 0° when estimating the PoG while in Method 2 λ_{eye} set to the measured R-Roll from the head tracker.

conditions and Method 2 sets λ_{eye} to the static R-Roll defined by the conditions of each specific video. Comparing the results of the two methods provides a direct estimate of the improvement in the accuracy of the estimates by the model when the actual R-Roll angle is used in the calculations. Both methods use the identical estimates of pupil center and corneal reflection locations because we generated those results through off-line processing of the same recorded videos. Therefore, we can ensure that the only difference between the PoG estimates of the two methods is associated with the use of the R-Roll angle in the computation. The PoG estimation results are quantified by generating the average estimation error for each sample video.

First, for each fixation target, we calculate the average absolute distance between each PoG estimate and the position of the associated target. Then we average the values generated for each of the five fixation targets.

4.4 Model Validation

Figure 4.5a and 4.5b show the PoG estimates of one subject (subject 2) when the R-Roll angle is 90°. Figure 4.5a shows the PoG estimates using Method 1 (λ_{eye} set to 0°) and Figure 4.5b shows estimates using Method 2 (λ_{eye} is set to the correct value of 90°). The crosses in each figure shows the location of the targets, and the scatter plot of dots are the estimated PoGs. When Figure 4.5a and 4.5b are compared it is clear that when the estimated R-Roll angle is used in the calculations of the PoG the accuracy of the PoG estimation improves.



(a) PoG estimates computed with Method 1 (no-compensation for R-Roll=90°)

(b) PoG estimates computed with Method 2 (with compensation for the R-Roll=90°)

Figure 4.5: Gaze estimates for subject 02 at 30cm device distance and $\lambda_{eye} = 90°$. The 5 crosses represent the fixation points. The point [0,0] is the center of the display.

Subject	01		02		03		04	
$\|\alpha_{eye}\|$	1.73		1.21		1.78		2.67	
$\|\beta_{eye}\|$	0.5		0.28		0.92		0.25	
λ_{eye}	45°	90°	45°	90°	45°	90°	45°	90°
δ_{theory}	1.37°	2.54°	0.94°	1.75°	1.53°	2.83°	2.06°	3.81 °
$\delta_{measured}$	1.22°	2.08°	0.86°	1.76°	1.36°	2.72°	1.81°	2.78°
$\|\delta_{theory} - \delta_{measured}\|$	0.15°	0.46°	0.08°	0.01°	0.17°	0.11°	0.25°	1.03°

Table 4.2: Predicted and measured differences between PoG estimations when the R-Roll angle is used in the computation of the PoG (Method 2) and when it is assumed to be 0 (Method 1)

Table 4.1 presents the average PoG estimation errors for each of the four subjects at three R-Roll angles (0°, 45° and 90°). The column labeled Method indicates if Method 1 or Method 2 were used in the estimation of the PoG. The results show that the average error for Method 2 is approximately 1 degree and that the magnitude of the errors for the three roll angles is similar. The average error increased by 3.2% (0.92° to 0.95°) when the R-Roll angle changed from 0° to 45° and by 26% (0.92° to 1.16°) when the R-Roll angle changed from 0° to 90°.

When Method 1 (i.e. $\lambda_{eye} = 0$) was used for the computation of the PoG, the average error increased significantly. The average PoG error increased by 251% (0.90° to 2.26°) when the R-Roll angle was changed from 0° to 45° and by 389% (0.90° to 3.50°) when the R-Roll angle was changed from 0° to 90°.

To highlight the importance of using the R-Roll angle in the estimation of the PoG in mobile-device-based eye tracking systems (i.e., limited screen sizes) consider a metric that describes the radius of a circle that includes 95% of the PoG estimates around a fixation target. When Method 2 is used for a 90° relative roll, 95% of the estimates are within a radius of 12mm from the fixation target. When Method 1 is used for 90° relative roll, the 95% enclosing radius is much larger, at 28mm. If one assumes that for applications that use gaze-selection, reliable operation requires that 95% of the estimates are associated with the intended fixation target, the use of Method 2 will allow 45 distinct fixation targets on a 5 inch display while Method 1 would allow for only 8 distinct fixation targets.

Table 4.2 shows, for each of the four subjects, differences in PoG estimation when λ_{eye} is set to 0° in the computation of the PoG and when λ_{eye} is set to the R-Roll angle. The Table also provides the differences computed by our error estimation model, δ_{theory}, (derived as δ in Section 4.2, equation 4.17) and the experimentally measured differences, $\delta_{measured}$. The last row of the Table gives the absolute difference between $\delta_{measured}$ and δ_{theory} for each R-Roll angle. The average error between the measured and predicted differences is 0.28° which is approximately 15% of the average measured differences (1.82°). This suggests that the model in Section 4.1 can be used to accurately determine the PoG when the R-Roll angle changes.

The data in Table 4.2 gives a sense of errors as a function of α_{eye} and βeye of the subject. The first two lines of the table gives each subject's α_{eye} and βeye. As predicted by equation 4.17, the magnitude of the measured differences is correlated with the magnitude of the offset between the optical and visual axes (α_{eye}^{ECS} and β_{eye}^{ECS}), with a correlation factor of 0.89 at a R-Roll angle of 90°.

We hypothesize that the small differences we do observe between δ_{theory} and $\delta_{measured}$ are due to inaccurate estimates the subject's specific eye parameters α_{eye}^{ECS} and β_{eye}^{ECS} during the calibration procedure. Errors in estimating eye features in the calibration videos combined with the proximity of calibration targets on a small mobile display increase the estimation error of the calibration parameters. It follows that if the feature extractor did a poor job estimating eye features, calibration parameters would be inaccurate and PoG estimation errors are likely also large. Therefore we can use outlier PoG estimation errors to indicate inaccurate calibration parameter estimation.

For subjects 01 and 02, the average gaze estimation error is small, which indicates that estimated subject-specific eye parameters were relatively accurate. For these subjects, the expected and observed changes in gaze estimations due to R-Roll are relatively small (less than 0.25°). For subject four, the average gaze estimation error is double the amount of the others, which indicates that estimated subject-specific eye-parameters were relatively inaccurate. The expected and observed changes due to R-Roll are much larger (1° when R-Roll was 90°) for this subject. These results suggest that the calibration procedure with small screen mobile eye trackers is vulnerable to errors.

In the above experiments the R-Roll of the devices and therefore λ_{eye} were known precisely and did not need to be estimated. In a more realistic scenario λ_{eye} must be estimated in real time, which can be done a few different ways.

4.5 Estimation of R-Roll, λ_{eye}

A parameter that also affects the accuracy of the estimated PoG in smart-phone-based eye-tracking systems is the accuracy of the measured R-Roll angle, λ_{eye}. Recall that R-Roll is the relative roll of the device and the eye. This is needed to transform some parameters from the ECS to the DCS. In general, to obtain an accurate measurements of the R-Roll angle in all circumstances, one could measure the rotation of iris structures in images from the camera [169] directly. This technique can provide accurate estimates of the R-Roll regardless of the root cause for the relative movements, whether it is device rotation, head rotation, ocular counter roll [170] or torsional eye movements that are governed by Listing's law [171] (these are rotations in the plane orthogonal to the visual axis as a function of gaze direction). This method, however, is impractical for current-generation mobile devices because of insufficient resolution to detect iris robust patterns in images from the front-facing cameras.

Without the ability to directly estimate the retaliative roll of the eye with respect to the DCS, less optimal solutions can be used. The R-Roll angle can be decomposed approximately as a combination of the head tilt angle with respect to the WCS and the eye's ocular counter-roll (OCR) [170]. Activation of the gravity sensors in the inner ear generates reflexes that act to maintain posture and gaze, and OCR is an example of such a reflex. OCR is a non-linear function of head-tilt, and it varies between subjects, but its gain is at most 0.1 [170]. This means that a head tilt of 10° with respect to gravity would never result in more than 1° of OCR (and for most subjects much less).

We first consider the estimation of R-Roll with a simplifying assumption of a seated upright subject. This condition minimizes the tilt of the head with respect to gravity which minimizes the effect of OCR. Later we relax this assumption.

4.5.1 Upright Head Tilt

The previous experiment minimized OCR impact on R-Roll angle with a chinrest. The chinrest keeps the head upright and prevents head tilt with respect to gravity during the experiment. During real-world use of the eye tracker however, it is not unreasonable to assume a subject is sitting or standing with their head approximately upright with respect to gravity, even if the hand-held device can rotate significantly. In this scenario, R-Roll is mostly captured by roll of the device with respect to the head. This can be estimated from a head-tracker that provides estimates of the head orientation in the DCS (accurate to 1-2° [168]).

Only a minor component of the R-Roll would be from the OCR associated with unintentionally head tilts of 5-10°. If the value of the R-Roll angle for this situation is estimated based on the measurement from the head tracker alone the overall error would be no higher than 3°. For the range of possible offsets between the optical and visual axis (Figure 4.3), a 3° error in the estimation of λ_{eye} would change the PoG estimation by only 0.06° to 0.27°. Estimating λ_{eye} using only the head orientation therefore seems reasonable for a subject with an expected upright head.

4.5.2 Any Head Tilt

When the subject's head is not assumed to be upright, and head-tilt can change significantly during the experiment, the R-Roll angle between the eye-tracker and the eyes could be significantly affected by OCR if only using head tracker to estimate R-Roll. A subject who changes posture from standing or sitting to lying horizontally (which is a head tilt change of 90°), could exhibit an OCR of as much as 9°. If the value of the R-Roll angle for this situation was estimated based on the measurement of head-tilt from the head tracker alone, the overall error in λ_{eye} could be as high as 10°. For the range of possible offsets between the optical and visual axes (Figure 4.3), a 10° error in the estimation of λ_{eye} would change the PoG estimation by 0.2° to 0.9°, a significant source of error.

An approach for estimating the R-Roll when the head is not presumed to be upright would be to create a model that describes the typical OCR as a function of head tilt with respect to gravity. In this approach, the orientation of the mobile device relative to gravity would be estimated using the accelerometer. Then the orientation of the head relative to the device would be measured by the head tracker. When combined, they can estimate the tilt of the head relative to gravity and the expected OCR generated from the model. λ_{eye} is generated by subtracting the estimated OCR from the head tracker measured head-tilt.

Unfortunately, the primary mobile device used in this work, the infrared prototype provided by Huawei, does not include a built-in accelerometer. As a result, we estimate the R-Roll between the device and the eye with the less accurate head tracker only method. On a commercial device, this would not be a limitation. As a result, we conducted no experiments with the subject lying down or otherwise exhibiting large intentional head tilt with respect to gravity.

The next chapter describes methods to accurately locate the input eye features (pupil center and corneal reflections) from the camera images so that we can provide robust input to our updated mobile-friendly gaze estimation model.

Chapter 5

Pupil Center and Corneal Reflection Feature Extraction

In this chapter, we discuss the feature extraction front end of our eye-tracking system. The feature extractor refers to the algorithms that estimate the location of the pupil center and corneal reflection eye features, given localized eye region windows as input. These are used in the model back-end, as discussed in the previous chapter.

The feature extraction process implicitly contains both regression and classification tasks. The classification task identifies which eye regions contain all required features. If the pupil is significantly occluded or a corneal reflection is missing, it indicates that the eye-tracking procedure should not proceed for the current eye region. The regression task produces estimates of the two corneal reflection and pupil center locations estimates. Robust and accurate estimation of eye features is a critical requirement for an accurate eye tracker.

The feature extraction methods developed in this work evolved from being a rule-based custom algorithm into a hierarchy of CNNs; we present both approaches in this Chapter. First, however, we begin with an overview of the dataset we generated to evaluate our feature extractor because understanding its properties illuminates the type of image variability we expect to encounter in a mobile eye tracker.

5.1 Dataset

In Section 2.2 we described principal causes of variability in eye images used for testing rule-based featured extraction techniques on head-mounted systems. The key issues for those systems are sun-light and environmental reflections that interfere with the eye-trackers eye images when the system is exposed to day-light illumination and estimation errors for large gaze angles. These issues are not as prominent in the mobile context as gaze-angles are small due to the size and the position of the screen of the cell-phone relative to the user's eyes. Also, the environmental reflections are minimal for our intended use case of a willing participant indoors. The image variability we expect in a smartphone environment comes mainly from the relative position and orientation changes between the device and the subjects face. There can also be higher variability because of the (possibly) weaker contrast between the pupil and iris, which is caused by limited illumination power and a noisier camera sensor.

Position	Range	Orientation	Range
x_{HCS}	-10cm to +10cm	ω (pitch)	-15° to +30°
y_{HCS}	-25cm to +5cm	ϕ (yaw)	-30° to +30°
z_{HCS}	+15cm to +40cm	κ (roll)	-45° to +45°

Table 5.1: Range of Device Positions and Orientations

5.1.1 Smartphone Position and Orientation Ranges

The motivation of this work allows us to place some constraints on acceptable device positions with respect to the face and eyes. We assume that the subject is sitting on a coach, or on a chair at a desk or table, in an indoor environment and is a willing participant. Within these constraints, there is still a wide range of distances and angles one could naturally hold the device in which we would want our system to maintain its accuracy.

We reference the device coordinate system (described in Chapter 4) to describe this range of device positions and orientations and introduce a new head coordinate system (HCS). The HCS is a right-handed 3D-Cartesian coordinate system whose x_{head} is co-linear with the line connecting the center of the eyes, y_{head} bisects the face vertically, and the xy-plane is coincident with the plane of the face. The z_{head} axis is perpendicular to the face pointing towards the mobile device, and the origin is the bridge of the nose between the eyes.

The smartphone 'position' is the location of the origin of the DCS with respect to the HCS. The smartphone orientation is the yaw (ω), pitch (ϕ) and roll (κ) angle of device relative to a cardinal orientation in which z^{HCS}_{device} intersects with the origin of the HCS and y^{HCS}_{device} is collinear with y^{HCS}_{head}. Table 5.1 presents the acceptable operating range of positions that we define in this work. Through instruction to the subjects we kept the device position and orientation to approximately within the ranges given in Table 5.1 during the collection of the dataset. Most of these ranges are set to keep the subject's head within the field of view of the camera.

Here we describe the more specific reasons for the choices in Table 5.1. When interacting with a smartphone, it is natural for the device and face to end up approximately coplanar with the device and below the eye line. More specifically, the device is typically centered in x^{HCS}, slightly negative in y^{HCS}, has a minimal yaw angle ω, and a slightly positive pitch angle (ϕ) to point back up towards the eyes. A minimal roll angle κ is imposed during data set collection to keep the device in landscape mode. Finally, the distance between the device and the head (z^{HCS}) can vary widely. We limited the distance to a range from 15 to 40 centimeters based on personal experience. When collecting data, we used the ranges in Table 5.1 as soft constraints. The experimenter would intervene to correct only if the subject's held the device significantly outside these ranges such that is was apparent by visual inspection. In practice, when subjects were instructed to hold the device anywhere they felt is comfortable, the intervention happened only 4-5 times during the collection of thousands of samples.

In the remainder of this thesis, when we speak of the *robustness* of the feature extractor or the eye tracker, what we mean is that it maintains a given level of performance when the relative position between the device and the subject's face changes, but stays within the ranges given in Table 5.1. An eye tracker that works well when κ is 0° but not 20°, or an eye tracker that works well when z_{DCS} is 20cm but not 30cm, is not a *robust* mobile eye-tracker.

The feature extractor design of a mobile eye tracker should account for large changes in the appearance of the eye regions when the position and orientation changes of the hand-held device changes. This large position variability creates a wide range of possible feature sizes captured by the front-facing camera during an eye-tracking session. Assuming a typical human pupil diameter (2-8mm [172]) moving within the typical distances one may hold the phone (15-40cm) implies that the diameter of the pupil can vary an order of magnitude. High variability in the pupil size makes it less desirable to use the datasets collected from head-mounted systems, or to use the algorithms found in [118, 119]. These datasets (while very challenging) do not represent the distribution of images that we expect from mobile devices. The associated algorithms implicitly make decisions based on the smaller expected pupil size variability and higher resolution images because the camera system is a fixed distance from the subjects' eyes. Thus it was necessary to collect a new infrared mobile eye region dataset and to build a new feature extractor for it.

5.1.2 Collection Methodology

As described above, there is no public dataset that adequately matches our use case and the desired eye region image distributions, and so it was necessary to collect one. The dataset consists of infrared images of eyes captured by our prototype smartphone (which was described in Section 3.1.2) using a sample collection app we developed. For each eye region the eye features are hand labeled. We used the following procedure to generate the dataset in which 2000 face images (4000 eye regions) were collected and labelled from 100 participants. Each participant was given the device and asked to position it anywhere that they felt comfortable holding a phone and look at a single target on the display while a single image of their face was captured. The target was positioned at a random location on the screen before each image was taken. The sole purpose of this target was to encourage the subject to look at the screen and not at the experimenter or elsewhere in the environment during data capture. Its location is not used for gaze estimation in any capacity (although it was recorded).

Then the subject was given five seconds to re-position the device and repeat the same procedure before the next image was captured. The positioning and re-positioning of the device resulted in a variety of relative distance and orientations of the device and the participants' heads. This process repeated twenty times for each subject, resulting in 20 infrared images of their face captured by the front facing camera. During the 20 image collection process a random set of 6 frame indexes are generated for the purpose of artificially increasing the number of invalid eye regions in the dataset. When capturing these specific frames either one or both of the infrared LEDs (chosen at random) are disabled resulting in eye regions unsuitable for the gaze estimation model used in this work. For the other 14 frames both LEDs were enabled. The orchestration of the LEDs in this fashion was done automatically by the samples collections app.

Both features that our model requires, the pupil center and corneal reflections, are low dimensionality simple geometric shapes viewed from a distance under 50cm and relatively on axis. As a result the number of examples required to adequately model and estimate their positions will be considerably less than other approaches which require modeling all of the variance in the whole eye region. We will demonstrate in the end-to-end eye tracking results in Chapter 6 that the front-end feature estimation performance achieved with only 4000 labeled eye regions samples was enough to produce state of the art mobile eye tracking results. The collection and labeling of many more data (e.g., 10 times more) might improve the robustness and accuracy of the system.

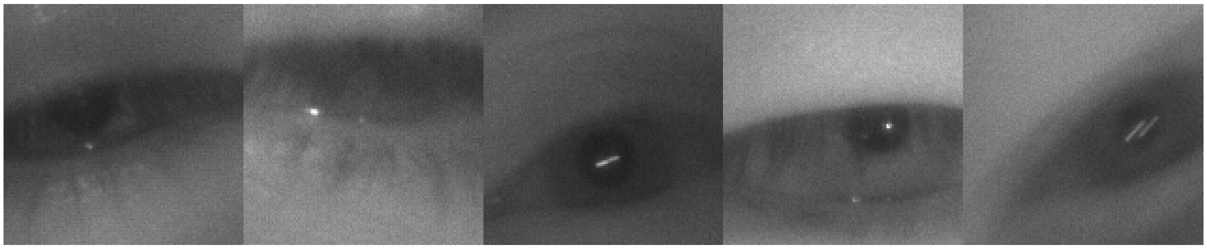


Figure 5.1: Example Invalid Dataset Images (128x128 pixel set)

5.1.3 Labeling

A human annotator was responsible for labeling each of the 2000 face images captured in the previous step. Labelling occurred in two stages; a classification stage and then a feature location stage.

Class Labeling

In the first stage each eye region is labeled with either a valid eye or an invalid eye class label. As discussed earlier in this chapter an eye region is considered ‘invalid’ if the pupil is significantly occluded, a corneal reflection is missing, or the image is exhibiting significant motion blur. Invalid eye regions can naturally occur during a blink or when images are captured during large device motion. Examples of eye regions labeled as valid and invalid are shown in Figure 5.2 and 5.1 respectively. The class distribution between valid and invalid eye regions in the 2000 captured images is approximately balanced, with 55% valid and 45% invalid. In many cases eye regions are either clearly valid or clearly invalid by the criteria mentioned above. Qualitatively approximately 1-2% of eye regions were difficult for the human annotator to classify. These cases require a judgement as to what is *too* much occlusion or *too* much motion blur. There is not a right or wrong answer here but it is important to highlight the existence of such cases and their approximate frequency. The upper accuracy limit of any classification algorithm applied to this dataset will be limited by this frequency.

Feature Labeling

In the second stage, the location of the pupil center and both corneal reflections were labeled for each of the valid eye region. To label the pupil center the annotator first clicks on 10 points along the pupil-iris boundary. These 10 boundary points are used to fit a pupil ellipse with a least squared optimization in a OpenCV [173] computer vision and image processing library. The center of the fitted ellipse is recorded as the ground truth pupil center location for our dataset.

After the pupil is labeled the two corneal reflection locations are labeled by the annotator with a single click. As most corneal reflections are very small, with a radius of 1-2 pixels, a single click is all that is feasible for estimating their locations. Single click estimation introduces significant quantization errors associated with mapping a continuously varying *real* center location with an integer [x,y] pixel location produced by a click. In ideal circumstances this quantization error results in ab average labeling error of ±0.25 pixels in both the x and y direction (or an average error magnitude of 0.35 pixels).

Figure 5.2 illustrates valid eye regions for which the features were annotated. As expected there is significant feature size variability in the 2000 face images that were captured. The pupil radius range from 4 to 28 pixels, and the corneal reflection radius range is 1 to 12 pixels.

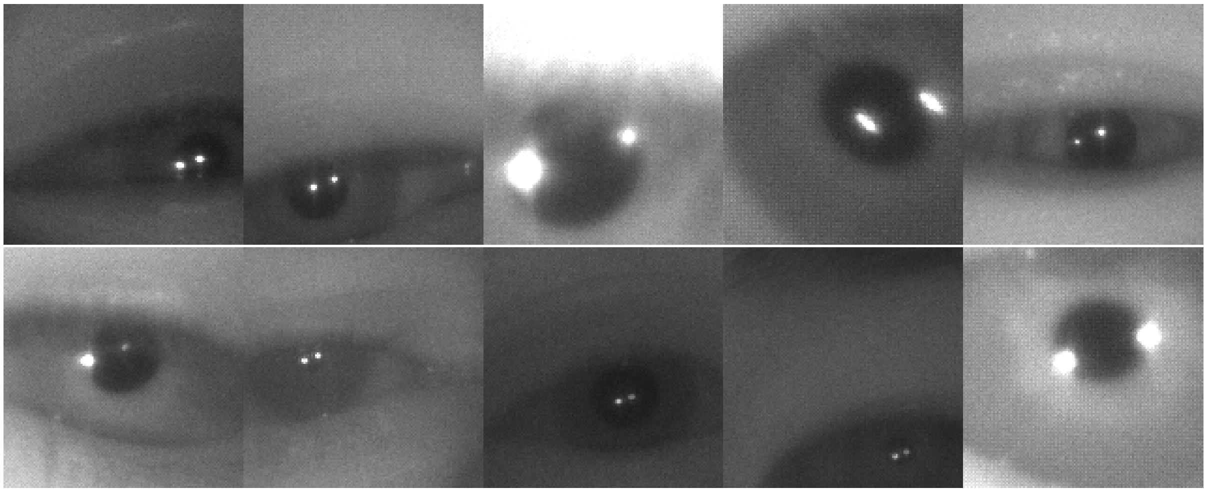
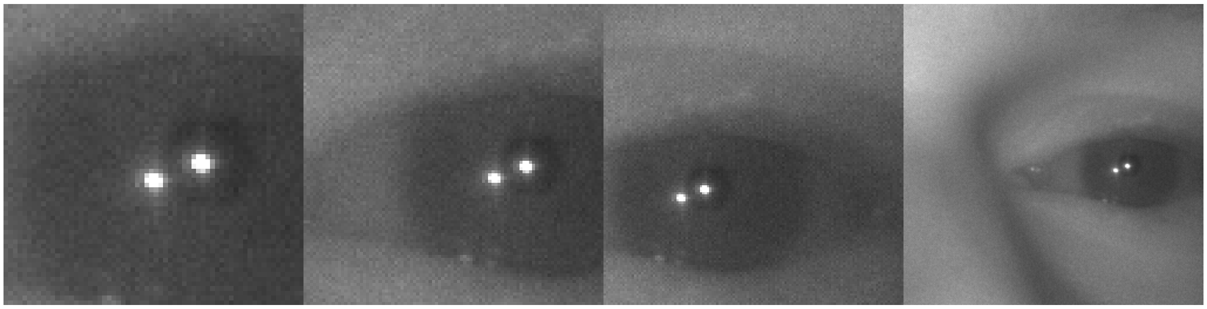


Figure 5.2: Example Valid Dataset Images (128x128 pixel set)



(a) 64x64 (b) 96x96 (c) 128x128 (d) 256x256

Figure 5.3: Example Eye Region Crops (in pixels)

5.1.4 Eye Region Dataset Generation

The final step in the dataset generation process is to create collections of eye regions from the face images with labeled eye features described previously. This is because the input to the feature extractors discussed in this Chapter are localized eye region windows like those shown in Figure 5.3 and not the full face images directly captured by the camera.

The eye region only datasets were constructed by taking 10 fixed size crops around each of labeled eye locations in the full face images. Each crop around the same eye location would vary the relative placement of the pupil center within that window under the constraint that both corneal reflections and the pupil must be contained in the windows for valid regions. The 10 specific crop locations were generated randomly from the set of all possible crop locations for a specific eye region that meets these constraints. By taking 10 fixed size crops per eye location instead of one each dataset was artificially increased from 4000 to 40000 samples.

Using this procedure a total of four eye region datasets were generated which differ only in the fixed crop size that was used for the eye windows; 64x64, 96x96, 128x128, and 256x256 pixels (as seen in Figure 5.3). By having datasets with different crop sizes, we can evaluate some tradeoffs between computational cost and feature estimation or classification accuracy.

Each of these four eye region datasets were then split into training and validation subsets containing

80% and 20% of the total number of samples respectively. We ensured that all the eye regions of any specific subject were in one of these two subsets and not split between them. There are more challenging eye region datasets presented in other works but non that are from infrared smartphones with the same level of feature size variance, illumination variance, and motion blur as presented here.

The end goal of this work is to approach the gaze estimation performance of desktop eye-tracking systems such as those presented in [4], which will require sub-pixel feature estimation accuracy. Using these datasets, we first attempted to make rule-based feature extractor with sub-pixel accuracy and then switched to a CNN-based feature extractor.

The final metric we are concerned with however is not the feature extractor performance but rather than end-to-end eye-tracking gaze estimation performance, which we discuss in Chapter 6.

5.2 Rule-Based Feature Extractor

The large size and contrast variability in our dataset constrain the types of techniques used for estimating the eye features. The general goals and constraints we tried to obey, to enhance the robustness and applicability of the results are:

1. minimize parameters that encode the expected size of the pupil.
2. minimize dependence on a consistent or stable pupil or iris intensity.
3. computationally efficient enough to run in real-time on a smartphone.

This section describes the algorithm developed to meet these goals.

In exploring different algorithms, we were unable to find a single feature extraction algorithm that provided robust estimates of the pupil center, in terms of the fraction of frames that the pupil center estimates the required achieved sub-pixel accuracy. Instead, our solution was to employ two different simple feature extraction approaches and to accept only estimates when both approaches agree. Each algorithm is prone to occasional gross estimation failures, but it is unlikely for these errors to be consistent for both approaches. Conversely, if both techniques produce similar estimates for the pupil center and radius, we can be relatively confident that we found an authentic feature. This dual-mode approach lowers the likelihood of a significant gaze estimation error but comes at the cost of increasing the likelihood of outright failure in processing a specific eye region (and therefore producing no downstream gaze estimation output).

We begin by describing the pre-processing steps we employed before the dual-mode consensus-based approach. Then we discuss the two approaches for the estimation of the pupil center, which we call *Edgle Voting* and *Flat Fill*. Finally, we discuss a relatively simple corneal reflection location estimation procedure.

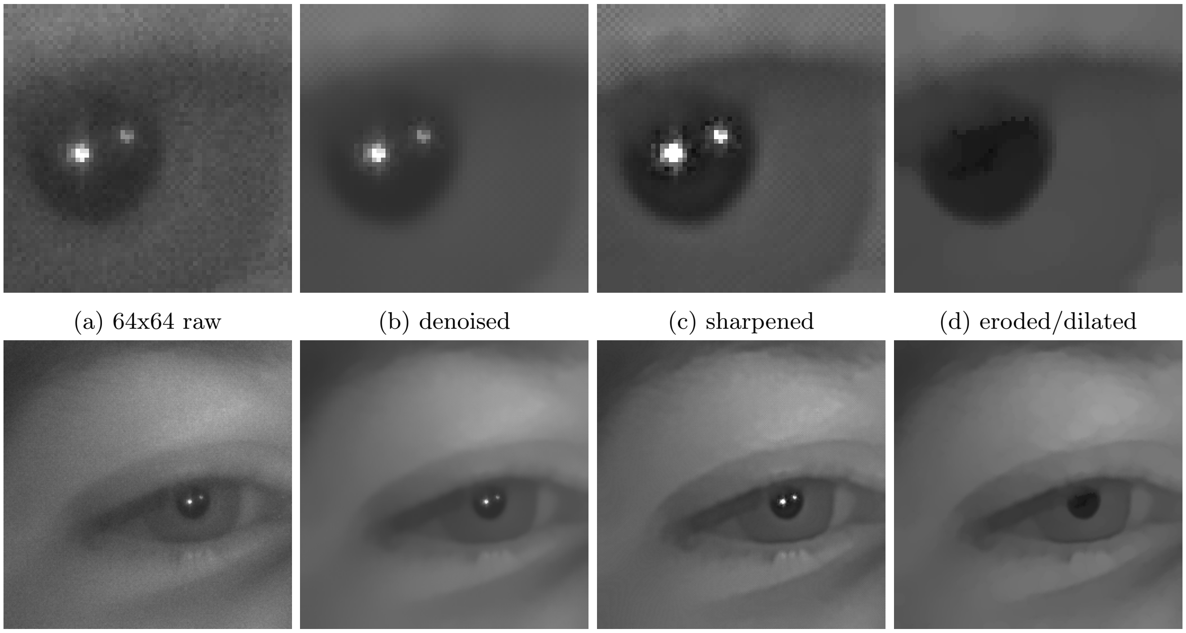
5.2.1 Eye Region Image Pre-Processing

Before estimating the pupil, a pre-processing step is performed to reduce noise, sharpen edges, and remove corneal reflections. Figure 5.4 illustrates these pre-processing steps on a candidate image at two eye region crop sizes, 64x64 and 256x256 pixels.

The first stage, (seen in Figure 5.4b and 5.4f), uses a 3x3 Gaussian smoothing operator [174] to reduce noise. Gaussian smoothing refers to convolving the image with a normalized 2D Gaussian kernel generated using Equation 5.1.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.1}$$

Applying this operator has the effect of reducing the images high frequency content because the Fourier transform of a Gaussian is another Gaussian. We explored more complicated denoising techniques such as anisotropic diffusion [175] and non-local means [176]. Subjectively these did produce more visually appealing images but had little impact on final pupil estimation accuracy. The sharpening stage (seen in Figure 5.4c and 5.4g) uses a standard Laplacian sharping algorithm [177]. The final stage (seen in Figure 5.4d and 5.4h) applies large kernel (11x11) erosion and dilation morphological operators. The erosion operator computes each pixel in the new image as the minimum intensity value found in an NxN window around that location in the input image. This has the effect of removing corneal reflections as they are brighter than their local surroundings. As a side effect, the pupil diameter increases, and so the opposite dilation operation is then performed to rectify this.


(a) 64x64 raw (b) denoised (c) sharpened (d) eroded/dilated

(e) 256x256 raw (f) denoised (g) sharpened (h) eroded/dilated

Figure 5.4: Pre-processing steps

All three pre-processing stages illustrated in Figure 5.4 are stored during the feature extraction processes and used independently as inputs to various downstream algorithms.

5.2.2 Edgel Voting

The first step in this approach is to differentiate the pre-processed image in both the horizontal and vertical directions, using standard 3x3 Sobel operators [178], to produces edgels at every pixel. An edgel consists of the direction and magnitude of the local gradient at a specific pixel in an input image. Edgels with large gradient magnitudes are potential candidate locations for points on the pupil-iris boundary (the gradient of the iris-sclera boundary has a lower magnitude in infrared images).

The edgels produced are then used with a decomposed Hough transform [179] to find the largest and highest contrast circular boundary of the eye region. This approach estimates a circle boundary with the following steps:

1. Create an accumulator space with one cell for each pixel in image.
2. For each edgel, identify all cells which could be the center of a circle with a boundary point at that edgel.
3. Increment cells identified in the previous step by the magnitude of the associated edgel gradient.
4. Estimate the center of the circle as the local maximum in the accumulator space.
5. Estimate the radius of the circle as the average distance between the center and each edgel which voted for it.

This approach differs from a traditional circle Hough transform in a few ways. First we operate on every edgel, not just edgels with gradient magnitudes above a certain threshold. Since the contrast between the pupil and iris for our mobile images can, in practice, vary widely and this removes the need to pick a static (or generate a dynamic) edgel selection threshold.

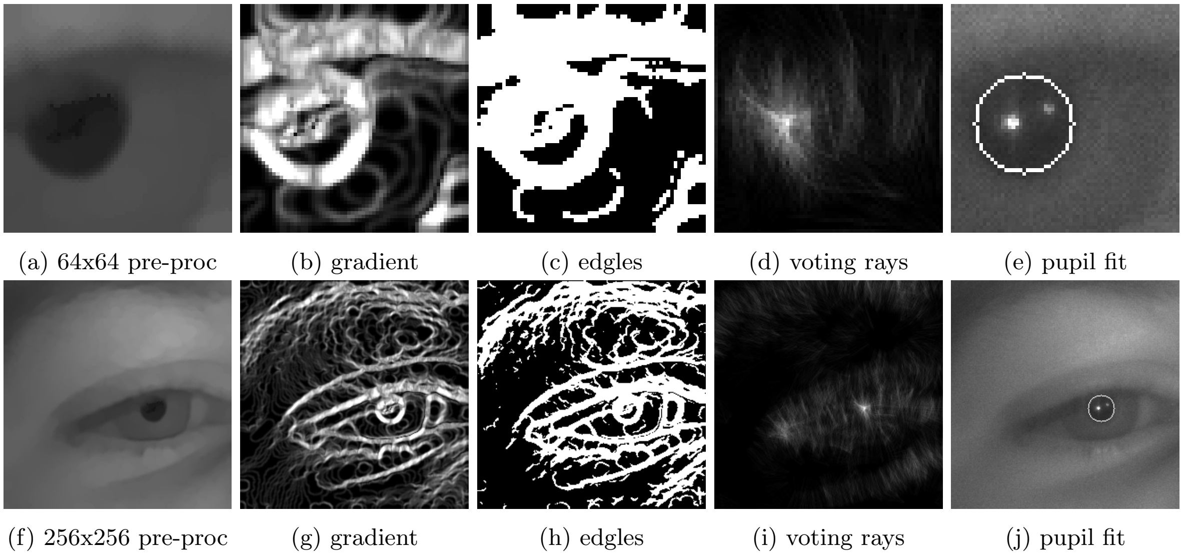
Next, the accumulator space is incremented by the magnitude of the edgel gradient instead of by one. This allows edgels at stronger edges in the image (such as the pupil-iris boundary) to automatically have a larger influence on the selection of the center of the estimated circle.

Finally, we use the direction of the edgel gradient to reduces the dimensionality of populating the accumulator space. Points along the circumference of a circle have gradient directions which point towards center of that circle. Therefore all candidate pupil centers for an edgel will only exist on the line that extends from the edgel in the direction of the gradient. When populating the accumulator space for each edgel our algorithm only needs to iterate over a one dimensional line in the direction of the edgel gradient. a visual representation of each of the stages discussed here can be seen in Figure 5.5.

The length of this line is a parameter in our algorithm (in pixels), which should be somewhere in between the radius of the pupil and the radius of the iris. The length of this line represents the maximum possible circle radius which the algorithm can, in principle, output. If the accumulator uses a length that is less than the radius of the pupil, the algorithm will not succeed in finding the center of the pupil. If the accumulator uses a length that is much larger than the pupil radius, it becomes possible to find other larger circular features such as the iris. However, since the 'true' pupil-iris gradient has high gradient magnitudes, edgels along that boundary vote with higher weight making the true pupil center the most likely output of this algorithm.

We set the length of the lines casts by each edgel in the accumulator space to be equal the radius of the largest pupil we expect to see for images captured from LittleBoy (our infrared prototype smartphone). Using the field of view of the camera, a minimum reasonable subject to device distance of 15cm, the largest expected human pupil size of 8mm radius, and the resolution of the camera, we can compute the radius in pixels of the largest pupil we would expect to see. Such a computation yields a 38-pixel radius, which is the value we used in this work. Practically speaking, this turned out to be a reasonable choice as the dataset we collected from 100 subjects, detailed in Section 5.1, which was used to evaluate both of our feature extractors, had pupils which ranged from 4-pixel radius to a maximum of a 34-pixel radius.

Using our edgel voting algorithm we generate one estimate for the pupil center in a given eye region. Then using a separate flat fill method we attempt to confirm this result.


(a) 64x64 pre-proc (b) gradient (c) edgles (d) voting rays (e) pupil fit

(f) 256x256 pre-proc (g) gradient (h) edgles (i) voting rays (j) pupil fit

Figure 5.5: Edgle voting steps

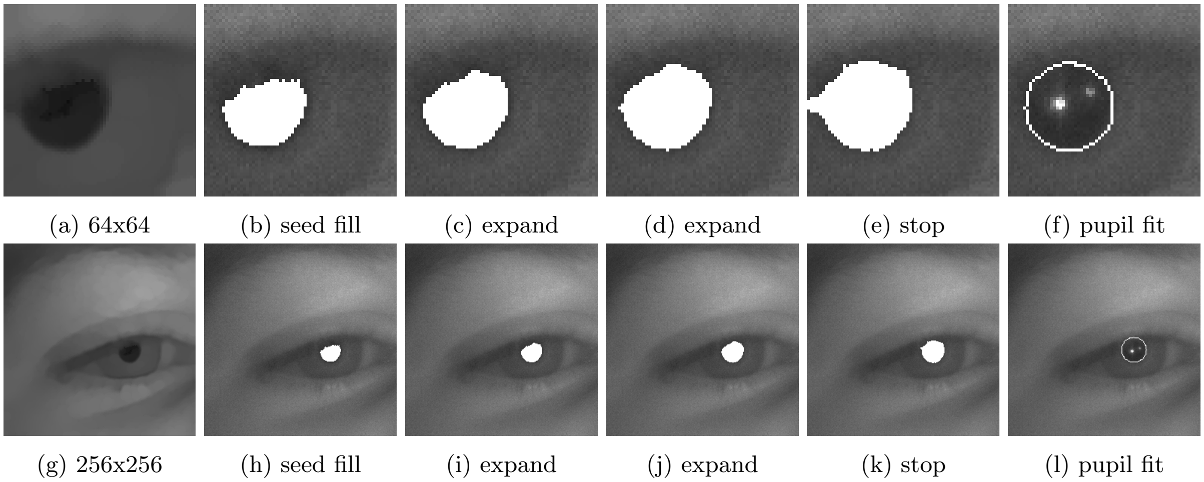
5.2.3 Flat Fill

The second simple pupil estimation technique that we use is a threshold-based flat fill algorithm that finds connected clusters of pixels that are at or below a given intensity threshold. As the pupil does not reflect infrared light, it should be the darkest region of eye.

In this stage, we generate a seed location from the pupil center using the output of the previous stage, and a seed threshold is selected as a small fixed offset above the intensity value at that location. A flat fill with that seed and threshold generates the first candidate connected component pupil, as shown in Figure 5.6. That connected component region is then successively expanded by increasing the flat fill threshold by small fixed intensity steps and appending new surrounding pixels. At each step, we compute the ratio of the area of the connected component to the area of a circle whose circumference is equal to the perimeter of the connected component (this is known as the Polsby-Popper compactness measure [180]). A connected component Polsby-Popper score falls with the range of [0,1], and a score closer to 1 indicates a more compact and more circular region. The termination criteria for this expanding flat fill process is when the Polsby-Popper score decreases. This will happen when the threshold increases above the iris intensity, and the connected component region begins to bleed out from the pupil into the iris.

This pupil estimation method requires a seed location inside the pupil, which, in our case, is generated from the edgle voting method. Otherwise, it makes no assumptions about the size of the pupil or the strength of the pupil-iris boundary gradient (both of which can vary significantly as a function of the environment and the subject, among other factors).

The two pupil methods for determining the location of the centre of the pupil, generated by the edgle voting method and flat fill method respectively, are then compared for similarity. If both candidate


(a) 64x64 (b) seed fill (c) expand (d) expand (e) stop (f) pupil fit

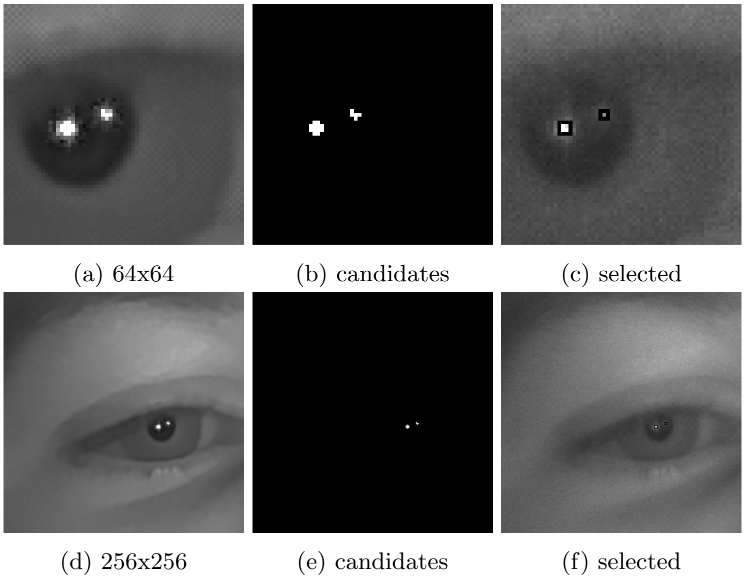
(g) 256x256 (h) seed fill (i) expand (j) expand (k) stop (l) pupil fit

Figure 5.6: Flat fill steps

results have a radius and center location within ±2 pixel of each other, we consider them to have located the same eye feature. A final pupil estimate is then constructed by averaging the two candidate locations. If the results of the two approaches are not similar, the feature extractor fails, and a gaze estimate is not produced downstream for this eye region. If the pupil is located successfully, the final step is the relatively simple process of locating the two corneal reflections.

5.2.4 Corneal Reflections

If the pupil was found using the approaches described above, the final step involves finding the two required corneal reflections in the denoised and sharpened version of the input image produced during the pre-processing phase.


(a) 64x64 (b) candidates (c) selected

(d) 256x256 (e) candidates (f) selected

Figure 5.7: Flat fill steps

The location process relies on the fact that the maximum pixel intensities inside the corneal reflections saturate the image sensor over a wide range of distances. Given this fact, we find corneal reflection candidates by locating all of the connected component regions with an intensity above a fixed intensity threshold. Doing so often produces only two connected components (if the corneal reflections are, in fact, visible). If it finds fewer than two connected components, the overall feature extraction is deemed to fail. If it finds two or more, the slope of a line which connects the centers of each pair is determined. The expected slope between the two corneal reflections in the image is defined by the geometry of the physical locations of the infrared LEDs in relation to the camera (as defined in the physical system configuration file) on the device. Therefore we can filter candidate corneal reflection pairs by selecting the pair with a slope closet to the expected slope. If no candidate pairs define a slope within ±7.5° of the expected slope the process fails and the corneal reflection were not found.

Section 5.5 will provide results from the rule-based feature extractor described in this section, and its use in a mobile eye-tracking system. However, as we will demonstrate later in this Chapter, machine learning techniques can produce more accurate and more reliable feature estimation results and, by extension, better gaze estimation results.

5.3 Machine Learning-Based Feature Extractor

Our machine learning based feature extractor consists of a hierarchy of multiple independently trained Convolutional Neural Networks (CNNs, outlined in Section 2.2.2). One of the CNNs operates explicitly as a classifier, and the others estimate feature locations. The classifier network determines if both corneal reflections are visible and that the pupil is not significantly occluded (such as it would be when a subject blinks). If this network determines that the eye region is 'valid', several independent position estimation networks determine the location of each of the required features.

The classifier and position estimation networks use similar CNN based architectures, and are illustrated in Figure 5.8. They contain convolutional layers (with batch normalization and optional pooling) followed by fully-connected dense layers. Within this base architecture, there are many configurable hyper-parameters (see Table 5.2), but the main difference between the networks is the output layer, the loss function, and the encoding of the ground truth. The discussion below, for both the classifier and position estimation networks, focuses on these differences.

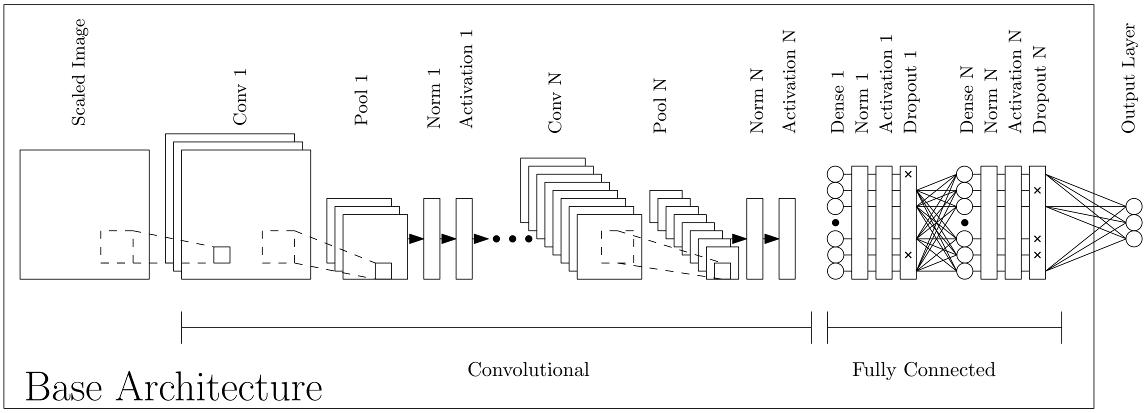


Figure 5.8: Base CNN Architecture

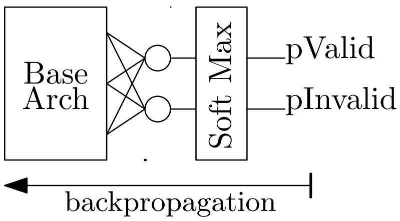


Figure 5.9: Eye Region Classifier Architecture

5.3.1 Eye Complex Classifier

The Eye Complex classifier is responsible for determining if an image of an eye region contains two corneal reflections and a pupil. Figure 5.1 and Figure 5.2 illustrate examples of valid and invalid eye region images. The output layer for the classifier, shown in Figure 5.9, extends the base architecture with a fully connected layer containing two neurons, one for each of the two classes: *invalid* or *valid*. This is followed by a softmax function to generate the probability that the given image belongs to either class. The ground truth for each training sample is a one-hot encoded vector indicating the class of the sample as either a valid eye region or an invalid region. A cross-entropy function that sums the log of the difference between each predicted class probability and the ground truth is used as the loss function. This architecture is a standard approach in image classification [181] and is not novel to our work. If the network classifies an eye region as invalid, no additional processing is performed, and the subsequent frame is processed. Otherwise, the next step is to determine the locations of the pupil center and the two corneal reflections.

5.3.2 Feature Estimator

The position estimation networks are crucial for achieving high accuracy and robust mobile eye-tracking. A standard approach to position estimation is the construction of a regression network, as illustrated in Figure 5.10a. In this approach, we extend the base architecture with a two neuron fully connected layer representing a predicted [x, y] relative location of the feature in the input image. For example, an inference output of [0.5, 0.5] would indicate the feature was in the center of the input image. The ground truth for each training sample is the human-labeled relative [x, y] location of the feature. The objective function for training a single network minimizes the Euclidean distance between the prediction location and the ground truth. At a higher level, we care about selecting a network that optimizes the accuracy of the position determination while being constrained by the computational and memory footprint limitations imposed by the desire to operate in real-time on a smartphone (without hardware neural network accelerators). Two factors that play an important role are the design of the output layer and the choice of the input image crop size.

Input Image Crop and Regression Hierarchy

The input image size plays a vital role in attaining both high accuracy and computationally efficient position estimation networks. It is important to avoid down-sampling the input image to achieve the highest accuracy, because of the reduction in spatial resolution. A direct, unscaled, crop of the original image near the eye region should be used to maximize accuracy . Figure 5.3 shows four examples of crop sizes that we explored when training our pupil center estimator.

We explored several crop-sizes and determined that a smaller input image size (a tighter crop around the eye region) results in less overall computation and produces (as shown later in Section 5.4) a more accurate estimate of the pupil center. However, the tighter crop requires that the system must already know approximately where the pupil center is. To solve this problem, we used a sequence of small CNNs to focus in on the rough pupil-center position successively.

These rough pupil locator networks do not need maximal accuracy and so, we trade accuracy for speed/computational effort by using down sampled inputs and accepting the loss in spatial resolution. For example, a 256x256 pixel eye region crop from the original face image could be scaled down to 16x16 pixels and a CNN could be trained with those scaled inputs. Such a network might only estimate the pupil center to within 10 pixels in the original image but would be very computationally efficient. A fast pupil estimate that is within 10 pixels of the true center is a good enough seed location to produce a 64x64 pixel crop from the original unscaled image. That tightly cropped eye region could then be sent to a more computationally expensive and precise locator network described above. We refer to this sequence of CNNs as a network hierarchy, the selection of the networks in this hierarchy is discussed in Section 5.4.

The use of a CNN hierarchy to produce tighter image crops is used as a way to optimize the performance to cost ratio of the base CNN architecture. The next step to explore changes in network architecture itself which will produce positive trade-offs between performance and computational costs.

Center of Mass Regression

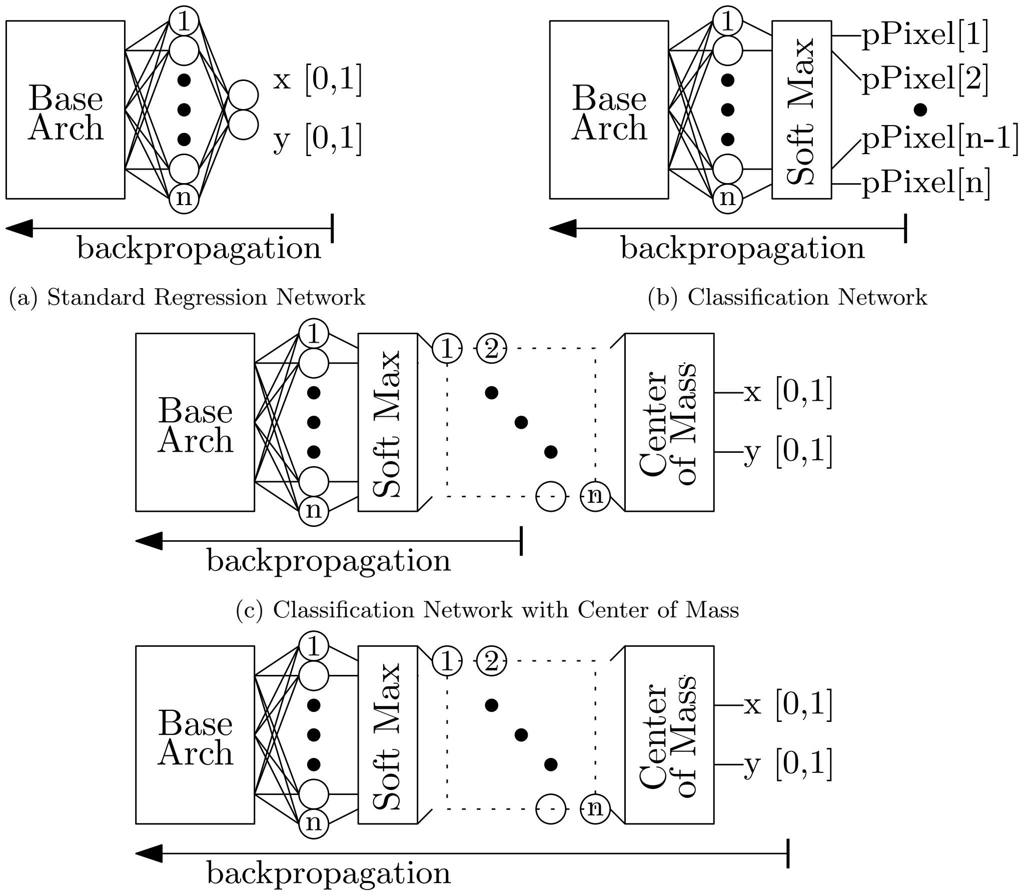
Next, we will consider how changes to the output layer of our base architecture may impact the ability of the network to accurately estimate a feature location (either pupil center or corneal reflections).

Naively, these locators are constructed as a standard regression network, as illustrated in Figure 5.10a, extending our base architecture with a fully connected hidden layer followed by a two neuron fully connected output layer representing the inferred [x,y] location of the feature. The objective function to minimize in this case is the Euclidean distance between the prediction location and the ground truth.

It is possible however, to recast the problem as a classification task rather than regression. This is a technique that has been demonstrated useful in the biomedical image segmentation domain [182]. To do this the base architecture is first extended with a fully connected layer with one neuron per input image pixel and then passed into a softmax function to generate a probability distribution. For an input image of n pixels this defines one output class for each pixel. Each class represents the probability that the feature (i.e. pupil center) location is located at that pixel. The structure of this network is illustrated in Figure 5.10b. The ground truth for each training sample would be a one hot encoded 2D mask indicating the location of the feature. The objective function to be minimized is the cross entropy function, as is standard with classification tasks. When performing a forward inference with a trained network the feature location is defined as the pixel corresponding to the maximum probability output by the softmax layer.

An issue with this classification approach is that real feature locations are between discrete pixels and our application requires sub-pixel accuracy. Thus rather than simply selecting the pixel with the largest predicted probability some averaging should be performed on the output probability distribution. Figure 5.10c illustrates how the final feature location can be obtained by interpreting the output neurons as a 2D grid and computing the center of mass (CoM) of the probability distribution.

A center of mass calculation over the whole probability distribution will result in poor feature esti-



(a) Standard Regression Network

(b) Classification Network

(c) Classification Network with Center of Mass

(d) Regression Network with Center of Mass

Figure 5.10: Eye Feature Locator Networks

mation accuracy if the distribution is multimodal. In these cases the output location will be computed as somewhere in between the two candidate location peaks. To force the network to produce unimodal probability distributions we convert the network back into a regression network as shown in Figure 5.10d. Extending the backpropagation during training to include the center of mass calculation. Now the output of the network is returned to being a continuous [x,y] value and the objective function to minimize is back to the Euclidean distance between the prediction location and the ground truth.

We call this architecture regression with center of mass and we will show in Section 5.4 that our center of mass architecture improves the ratio of the feature estimation accuracy to computational cost relative to the direct regression approach.

Figure 5.11 shows the complete network hierarchy for our feature extractor. The input to this hierarchy is a very coarse eye region which is produced periodically from a head-tracking system, or based on the final pupil center estimate produced by the eye-tracker in a previous frame. The hierarchy consists of six total networks: a) two very fast networks that localize successively tighter eye regions; b) one network to determine if the generated tight region is a valid eye region; c) three accurate position estimation networks to determine the exact position of the pupil center and both corneal reflections. These positions are then given to the gaze estimation model to compute the PoG. The next section discusses how the feature extractor networks were trained and selected before being integrated into the

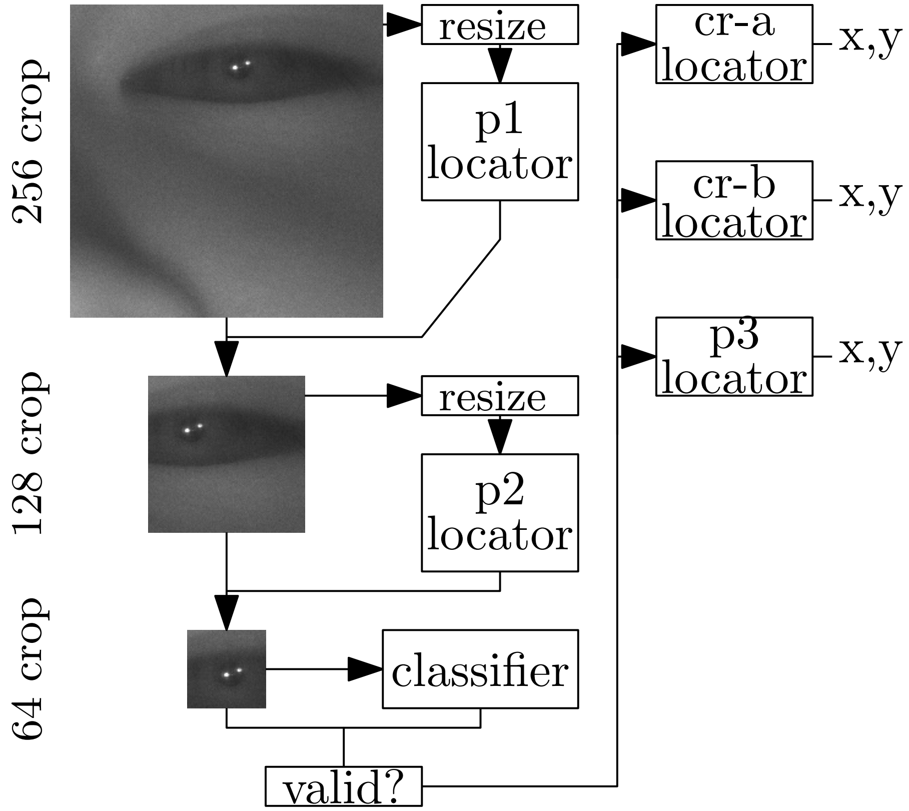


Figure 5.11: Multiple Network Feature Estimator Hierarchy

end-to-end eye-tracking system.

5.4 Networks Training & Selection

5.4.1 Eye Complex Classifier Network Selection

To evaluate and select the hyper-parameters for the eye complex classifier network, we performed the following experiment: first, we generated 256 different hyper-parameter configurations for the base architecture (illustrated in Figure 5.8) from the set shown in Table 5.2. Then we trained each architecture independently with the four datasets each with different initial image crop sizes as discussed in the previous section. Each network was evaluated based on the classification accuracy of its corresponding validation set and the computational cost of one forward inference through the network. The computational cost is measured as the number of required floating-point operations, a metric that is provided by the tf.profiler() method included in Tensorflow version 1.12, the neural network library that we used [183].

Figure 5.12 shows the Pareto-optimal networks for this experiment, where each point represents the accuracy and computational requirement of a specific instance of one of the configurations of the trained network. The X-axis of the figure is the true positive classification accuracy using a threshold of 0.5, and the Y-axis is the number of floating-point operations required to compute a single classification. The four curves correspond to the size of the input image crop that was used to train and validate data for each curve. Observe that the bottom curve, corresponding to a 64x64 input image crop, is superior to all the other curves. This implies that training on a smaller eye region crop produces similar or better

Hyper Parameter:	Choice Set
Input Down Scaling Factor:	[1x, 2x]
Number of CNN Layers:	[1, 2, 3, 4, 5]
Number of Kernels in CNN Layer:	[4, 8, 16, 32, 64, 128]
Square Size of Kernels in CNN Layer:	[3, 4, 5, 6, 7, 8, 9]
Max Pooling at Each CNN Layer (2x2):	[enabled, disabled]
Number of Fully-Connected Hidden Layers:	[0,1,2]
Number of Neurons in a Fully-Connected Hidden Layer:	[64, 128, 256, 1024, 2048]
Dropout % of Fully-Connected Layers:	[0.3, 0.4, 0.5, 0.6, 0.7]

Table 5.2: Base Architecture Hyperparameter Ranges

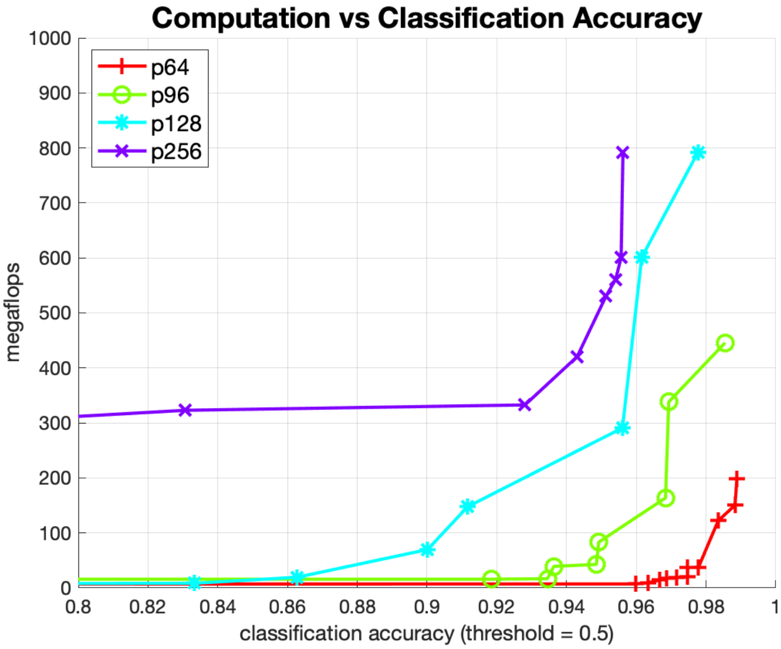


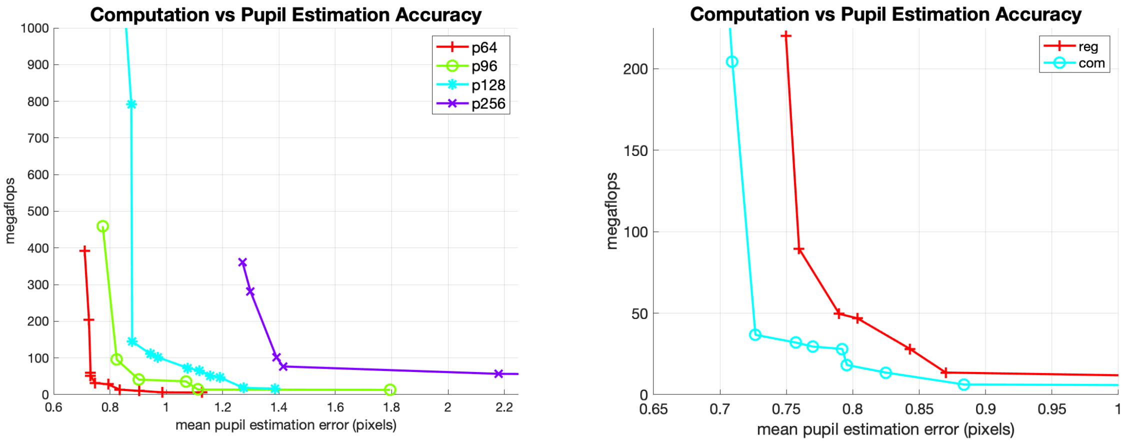
Figure 5.12: Pareto Classification Network Architectures: Performance vs Cost

classification accuracy with much less computational effort. As an example, to achieve a classification accuracy of 98% on the 128x128 pixel dataset required a network that used roughly thirty times more computation than to achieve the same accuracy on the 64x64 dataset.

The final chosen configuration achieved an classification accuracy of 98%. Recall that during the labelling discussion in Section 5.1.3 that we estimate that approximately 1-2% of the eye regions in our dataset could have been reasonably classified as either valid or invalid. For this reason our classification networks would have logical upper bounds on accuracy of 98-99%. The basic classification networks used in this work are therefore approaching upper bound performance and no additional optimization is needed.

5.4.2 Feature Location Estimator Network Selection

Once an eye region has been classified as valid (i.e., containing a pupil and two corneal reflections) the next step involves locating the position of those features. The pupil center and corneal reflections



(a) pupil networks

(b) pupil networks w/wo center of mass

Figure 5.13: Pareto Pupil Estimator Network Architectures: Performance vs Cost

position estimation networks were evaluated in a similar way to the eye complex classifier above. We selected 256 hyper-parameter configurations for the base architecture from the set of parameter ranges given in Table 5.2. Each network was evaluated based on the mean distance between the network output and the ground truth locations, as labeled by the human annotator, measured in pixels of the original image.

Figure 5.13a shows the Pareto optimal curves for the suite of pupil estimation networks using the network with the center of mass output layer as described in Section 5.3. The X-axis is the mean estimation error (in pixels), and the Y-axis is the number of floating-point operations required to compute a single estimate.

The average pupil estimation errors reported in this figure were computed after eliminating estimates that had large errors (greater than 7.5 pixels). The removal of these large errors is required to normalize the comparison between networks trained on different image crop sizes. Otherwise, the mean estimation error favors smaller crop sizes. For example, a gross feature estimation failure on a 64x64 image crop cannot result in an error of more than 64 pixels (in either direction) but on a 265x256 image crop the error could be as large as 256 pixels. In both cases however, the smallest error which can be produced is zero pixels.

The effect of image crop input size on the estimation error for the corneal reflections was explored in an analogous way to the pupil. Figure 5.14 shows the Pareto optimal curves for a suite of networks with the center of mass output layer as described in Section 5.3 but estimating the location of one of the corneal reflections.

Figure 5.13a and 5.14 shows that there is a significant advantage in training either the pupil or corneal reflection networks location networks using images with smaller crop sizes. These advantages include both performance to cost ratio and absolute estimation accuracy. This leads us to choose a 64x64 crop size for our accurate pupil center and corneal reflection locator networks.

The next experiment evaluated the performance of the pupil location estimator with and without the center-of-mass output layer described in the Section 5.3.2. Figure 5.13b shows the Pareto optimal curves for networks using simple regression and network using the center-of-mass output layer. Across all 256 hyper-parameter configurations, the average estimation error was improved by 14% by using the

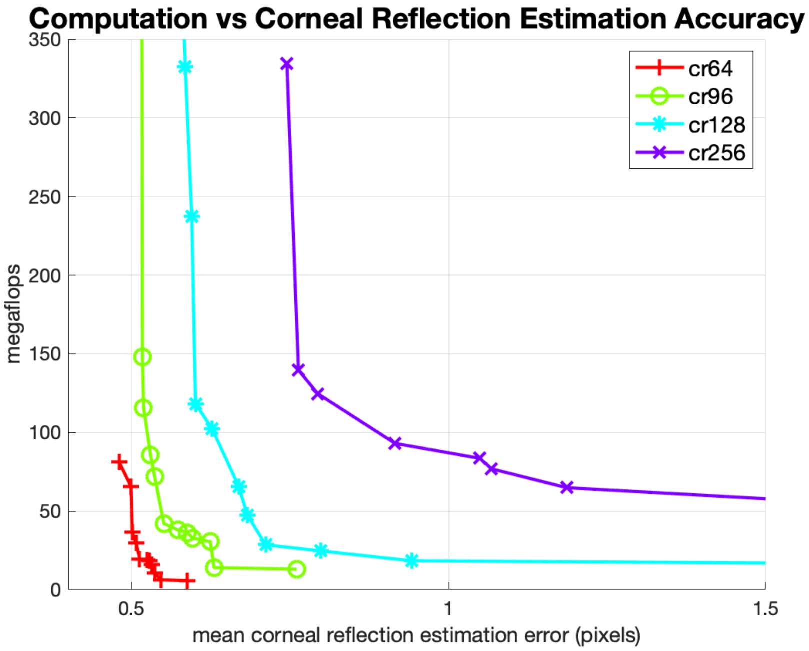


Figure 5.14: Pareto Corneal Reflection Estimator Network Architectures: Performance vs Cost

center-of-mass calculation. Near the optimal trade-off points of the Pareto curves, this improvement was roughly 10%. For any specific computational cost, the net reduction in pupil estimation error due to both tighter crop sizes and the center-of-mass output layer is about 50% when compared with a naive single CNN regression network trained on a 256x256 image crop.

Figure 5.15 shows the Pareto optimal curves when training networks to estimate the location of each of the two corneal reflections. The 256 hyper-parameter configurations for architecture exploration were generated again from the set defined in Table 5.2. All networks in Figure 5.15 were trained on the smallest 64x64 eye region window crop and used the center-of-mass output layer, as described above with the pupil estimation networks. In our end-to-end eye tracking system the corneal reflection locations are only estimated after the pupil is found, so there is not need to create a coarse to fine hierarchy of networks as was done previously for the pupil.

Figure 5.15 indicates that these networks cannot estimate the location of the corneal reflections with an error of less than 0.5 pixels. A large part of this 0.5 pixel estimation error limit is explained by the quantization error involved in the labeling process of the ground truth that was discussed in Section 5.1.3.

Using the Pareto optimal curves for the eye feature regression and classifications networks, it is necessary is to select the specific networks to be used in the end-to-end eye tracker. Here specific choices of accuracy performance vs. computational cost must be made. Our final feature extraction network hierarchy is composed of 6 networks; two small locator networks that quickly localize the pupil, an eye complex classifier to determine if the cropped image is valid, and three location estimators for the pupil center and corneal reflection locations.

Four of these networks, the classifier and the three precise feature locator networks, are selected to optimize the trade-off between performance (accuracy) and computational cost. These four networks are trained on 64x64 pixel image crops with possible hyper parameter configures defined in Table 5.2.

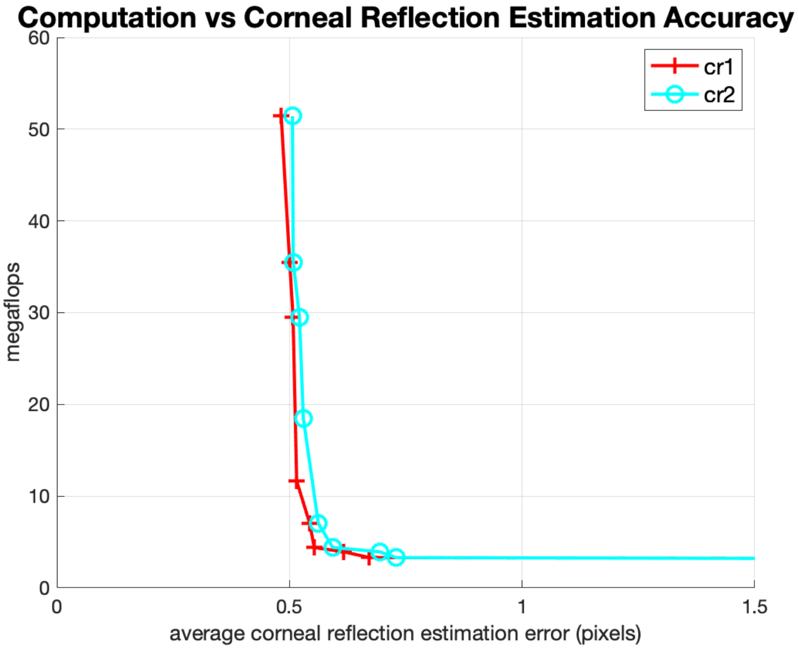


Figure 5.15: Pareto Optimal Corneal Reflection Network Architectures: Performance vs Cost

network	crop size (px)	downscale	average error (px)	cost (mflops)
pupil coarse	256	8x	3.8	2.5
pupil medium	128	4x	1.6	2.5
pupil fine	64	1x	0.75	32.0
corneal ref a	64	2x	0.52	11.6
corneal ref b	64	2x	0.53	12.6
			accuracy	
classifier	64	1x	97.9%	20.1

Table 5.3: Selected Networks: Size, Performance, and Cost

The coarse and medium pupil locator networks were trained with the 256x256 pixel and 128x128 pixel image crop datasets respectively. The hyper parameter sweep used to test the architectures for these networks was the same a Table 5.2 with one notable exception. The input down scaling factor was chosen from the set [4x, 8x, 16x] rather than from [1x, 2x]. These coarse and medium pupil locator networks are intended only to generate a rough seed location of the pupil that will be used to create the 64x64 pixel crop used by the other networks. As such we can accept large reduction in spatial resolution in order to drastically reduces the number of parameters (and thus speed) of the networks.

The performance and cost of each of our six network are presented in Table 5.3 which shows the image crop sizes, amount of down-scaling where applicable, accuracy, and computational cost of these networks. While these networks represent a good trade-off between compute costs and accuracy we could have selected either more expensive and higher performing networks or lower cost and worse performing networks. Examples of which are outlined in Appendix C.

There was significant flexibility in the selection of the hyper-parameters that generated the best results. In most cases, we could have selected a different network near the same location on the Pareto

network	cnn l1		cnn l2		cnn l3		cnn l4		dense		cost (mflops)
	k	w	k	w	k	w	k	w	neurons	dropout	
pupil coarse	24	4x4	2	3x3	16	5x5	n/a	n/a	512	0.7	2.50
pupil medium	24	4x4	2	3x3	16	5x5	n/a	n/a	1024	0.7	2.5
pupil fine	8	4x4	32	6x6	8	3x3	32	6x6	1024	0.7	32.0
corneal ref a	64	5x5	8	3x3	64	5x5	n/a	n/a	1024	0.5	11.6
corneal ref b	16	4x4	48	3x3	48	6x6	n/a	n/a	1024	0.4	12.6
classifier	8	7x7	n/a	n/a	n/a	n/a	n/a	n/a	1024	0.4	20.1

Table 5.4: Selected Networks: Architectures

curves that had different hyper-parameters but performed very similarly. For example, a network might have had more CNN layers but fewer kernels per layer, resulting in approximately the same overall computational requirements and accuracy. However, we can draw some conclusions from the networks which produced the best trade-off between accuracy and computational cost: all nearby networks (in the Pareto-optimal sense) have max pooling enabled after each CNN layer. The loss in spatial resolution of the intermediate feature maps that results from pooling was less critical than the improvement which stems from adding more kernels when normalized for total compute requirements. Another commonality among all networks that appears near the optimal trade-off between accuracy and computational cost is the inclusion of one, and only one, hidden fully connected layer. For completeness Table 5.4 presents the number of CNN layers, number of kernels per layer (k), the width of the kernels in each layer (w), the size of the single hidden fully connected layer and the dropout percentage of that layer.

5.4.3 Feature Extraction Network Hierarchy Compute Costs

The architectures and performance of the hierarchy of networks used in our feature extractor is shown in Table 5.4. The most computationally 'expensive' network in Table 5.4 is the pupil fine location network which determines the final pupil location, at 32 megaflops. In total, if all six networks performed one forward inference, it would cost 81 megaflops for a single eye region or 162 megaflops for both eye regions in a single frame.

The use of megaflops as a unit of computation has allowed us to compare and contrast networks explored in this work. It is important however, to understand what these values represent in terms of absolute real world performance, measured in milliseconds, when these networks are run on a smartphone. Ideally a mobile eye tracking system would be able process incoming frames as fast as they can be provided by the devices camera. On our infrared prototype smartphone (and most modern devices) the front facing camera produces images at a rate of thirty frames per second (fps). As a result the target processing times per frame should be under 33ms.

Our infrared prototype smartphone averages 257ms of processing time when running the proposed feature extractor on both eye regions in a single frame. This does not meet the performance target of 33ms and translates to four frames per second. This is due to the devices lack of the hardware neural network acceleration present in modern smartphones. The size of networks and amount of computation required, outlined in Table 5.4 were selected based on thir expected performance on a modern device with this type of acceleration.

We estimate the run time performance of our networks by interpolating between the performance of similar CNN networks which have already been benchmarked on modern devices. Google has released

feature extractor	accuracy	precision	recall	f1 score
rule-based	94.9%	96.9%	94.6%	95.7
machine learning	97.9%	98.0%	98.5%	98.3

Table 5.5: Feature Extractor Classification Performance

multiple suites of small, low-latency, low-power image processing networks for classification, detection, embeddings, and segmentation tasks they call MobileNetsV1 [184] and MobileNetsV2 [185]. The smallest of these networks have a computational cost of 28 megaflops and runs at 5ms (200fps) on the Google-branded Pixel 3 smartphone, released in October 2018. The largest MobileNets require 1100 megaflops and ran at 200ms (5fps) on Pixel 3 smartphone. Using this data we can estimate that the 162 megaflops of image processing CNN inference would take approximately 29.5ms (33fps) on a Pixel 3 smartphone.

With both the rule-based and machine learning-based feature extractors presented, the next Section compares and contrasts the performance of each.

5.5 Performance Comparison of Rule-Based vs. Machine Learning Based Feature Extraction

Both the rule-based feature extractor described in Section 5.2 and the machine learning-based feature extractor described in Section 5.3 were evaluated using the validation set of the 256x256 cropped eye region dataset. Both feature extractors will quickly narrow in on the pupil and then sub crop to a smaller size before performing the most expensive operations.

The validation set contains both eye regions labeled as valid and invalid. Valid eye region indicates that the pupil and both corneal reflections are visible (and their locations have been hand-labeled), and the feature extractor is expected to produce a result. Invalid eye regions indicate the eye region is mid-blink, or missing a corneal reflection, or is overly blurry from device motion. In this case, the features extractor is expected to indicate the eye features were not found successfully. This segments our evaluation into both classification and regression statistics.

5.5.1 Classification

The classification statistics in Table 5.5 show accuracy as well as the precision, recall, and f1 score of our rule-based and machine learning feature extractor for a classification threshold of 0.5. The machine learning method does outperform the rule-based method with an accuracy of 97.9% vs 94.9% and an f1 score of 98.3 vs 95.7.

Unlike other classification tasks, there is some subjectivity as to when an eye region is *too* dim, a corneal reflection is *too* weak, or a pupil is *too* occluded to continue. The rule-based methods recall of 94.6% vs 98.0% for the machine learning method indicates that the rule-based method exhibits a higher propensity to discard a 'valid' eye region, which may be of low quality. This is acceptable behavior as a falsely rejected eye region results in nothing more than a frame that doesn't produce a gaze estimate for that eye. A falsely accepted eye region, however, is likely to produce inaccurate feature location estimates and result in significant gaze estimation errors for the eye tracker for that particular frame. The precision of the comparison between the rule-base and machine learning feature extractors were closer at 96.9% and 98.0% respectively, which is indicative that both systems can identify eye regions well.

The more substantial difference between our rule-based and machine leaning-based methods is seen in the feature location estimation accuracy.

5.5.2 Pupil and Corneal Reflection Location Accuracy

If an in input eye region sent to either feature extractor produces a valid result, it returns the precise location of both the pupil center and two corneal reflections. Here we evaluate the accuracy of these estimates with three main regression statistics; average estimation error, estimation bias, and root mean squared estimation error (RMS). The average estimation error refers to the average length of the estimation error vectors. It is computed with equation 5.2, where N is the number of eye region samples, x_i and y_i are the estimated x and y location for a particular sample and gtx_i and gty_i are the ground truth labeled x and y location for a sample.

$$\frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - gtx_i)^2 + (y_i - gty_i)^2} \tag{5.2}$$

.

Estimation bias refers to the center of mass of the estimation error vectors and is computed with equation 5.3.

$$\sqrt{(\sum_{i=1}^{N} x_i - gtx_i)^2 + (\sum_{i=1}^{N} y_i - gty_i)^2} \tag{5.3}$$

.

Finally, we present RMS which the square root of mean squared error and is computed with equation 5.4

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - gtx_i)^2 + (y_i - gty_i)^2} \tag{5.4}$$

A lower value for any of these statistics is beneficial for the gaze estimation error produced by the downstream eye tracker. Of the three, however, bias is particularly important for accurate handheld mobile eye-tracking results because the effects of feature estimation variance can be mitigated in the end-to-end eye-tracker by averaging successive gaze estimates. Large feature estimation bias reduces the accuracy of subject-specific eye parameter estimation during calibration which reduces the gaze estimation accuracy during large relative motions between the device and the head (sub-optimal compensation for relative movements after calibration). Increased bias in the features estimates will therefore results in increased degradation of the eye trackers performance as the device move further away from the position in which it was calibrated. This effect is demonstrated in the end-to-end eye-tracking result presented in Chapter 6.

Table 5.6 shows statistics for estimating the location of the pupil center and both corneal reflections. Only valid eye regions had these locations labeled, so the following statistics indicate how well the feature extractors performed on eye regions that were valid but did not incorporate how poorly they perform on invalid regions, which might be incorrectly classified as valid.

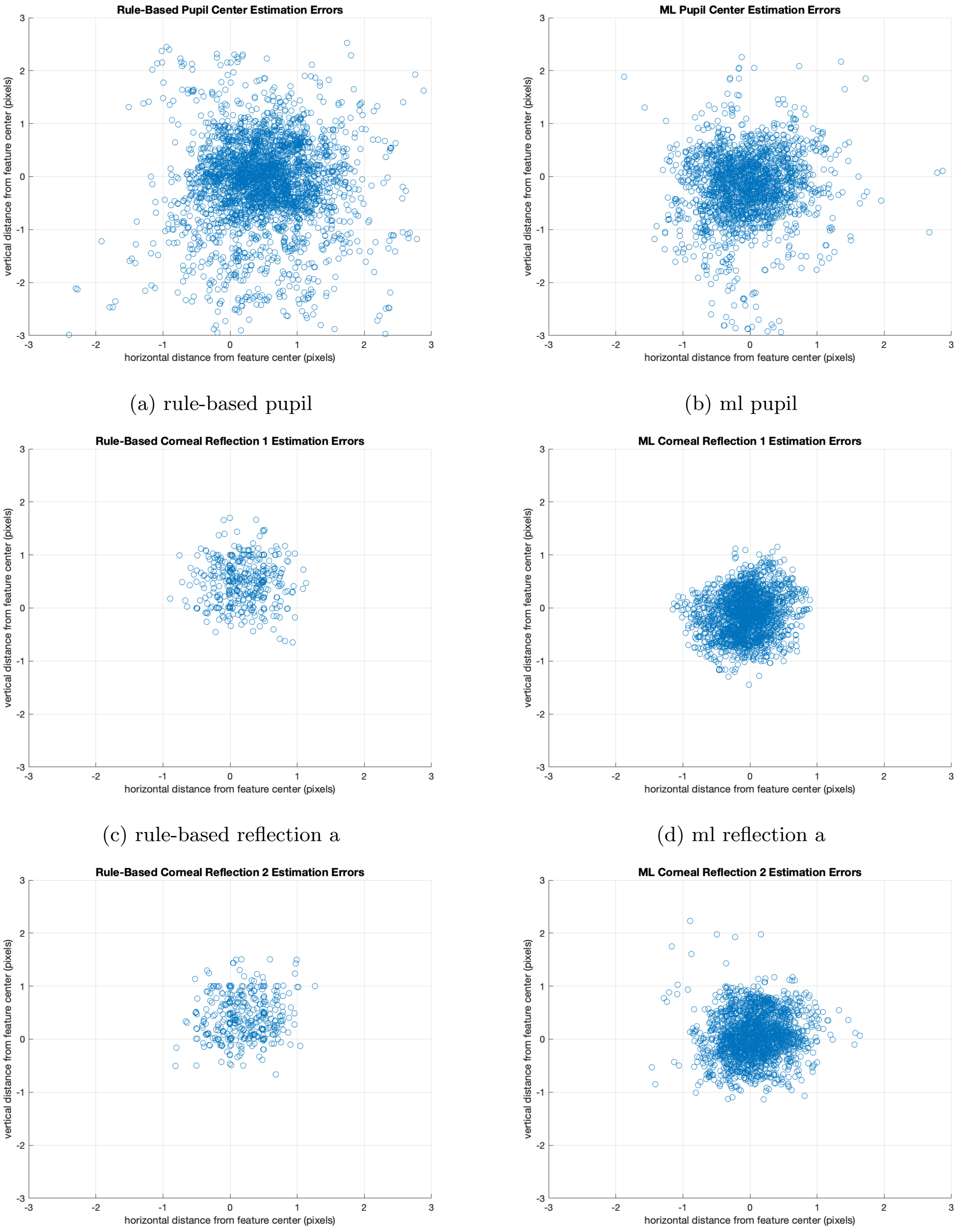
feature	method	error (px)	bias (px)	RMS (px)
pupil center	rule	1.13	0.52	1.36
	ml	0.72	**0.08**	0.89
reflection a	rule	0.66	0.51	0.74
	ml	0.47	**0.11**	0.55
reflection b	rule	0.62	0.45	0.78
	ml	0.49	**0.10**	0.58

Table 5.6: Feature Extractor Estimation Accuracy (N=4380)

Table 5.6 presents the average error, bias, and RMS produced by our rule-based feature extractor when estimating the pupil center and both corneal reflections on the eye regions images contained in our validation set. The corneal reflection estimation errors and RMS are similar for both the rule-based and machine learning-based methods at about 0.5 pixels. As discussed in Section 5.1.3 there will be some quantization error in the labeling of the corneal reflections. However the annotator is equally likely to have clicked the pixel to the left or the right of the true corneal reflection center therefore the lower bound for our bias over many estimates is still zero. This is where the machine learning method demonstrates its superiority with a corneal reflection estimation bias of only about 0.10-0.11 pixels vs 0.45-0.51 pixels for the rule-based method. The machine learning approach also bests the rule-based approach in all metrics for pupil center estimation as well. The average estimation error dropped from 1.13 to 0.72 and the RMS dropping from 1.36 to 0.89. The bias is the most significant improvement dropping from 0.52 to 0.08 pixels.

A visual representation of the estimation errors for all three features using both methods is shown in Figure 5.16. In this figure, each scatter point represents the estimation error associated with a single eye region image from our validation set. The rule-based method appears to be showing many fewer samples than the machine learning method, but this is not the case. The perception of fewer points is a result of the fact that our data set was augmented to have 10 random crops of each eye region with the pupil located in different regions. For the rule-based method, it is much more likely for each of these 10 crops to produce a feature estimate that is in the *exact* same position relative to the ground truth and thus many of the scatter points are directly overlapping.

In the next Chapter, we demonstrate how the reduction in feature estimation bias shown here has a positive impact on the end-to-end eye-tracking results.



(a) rule-based pupil

(b) ml pupil

(c) rule-based reflection a

(d) ml reflection a

(e) rule-based reflection b

(f) ml reflection b

Figure 5.16: Feature estimation error scatter plots for CNN hierarchy on 256x256 pixel image crops

Chapter 6

End-to-End Eye Tracking

This Chapter will present the performance of the full smartphone eye tracking system as described in Chapter 3 using the model presented in Chapter 4 and both feature extraction methods presented in Chapter 5.

We will present the end-to-end eye tracker results though a series of experiments which become successively more challenging scenarios for a mobile eye tracking system. In the cases where equivalent experiments were performed by the two other prominent mobile eye tracking systems discussed in Chapter 2.3 (Screen Glint [47], GazeCapture [46]) we will directly compare the results to theirs.

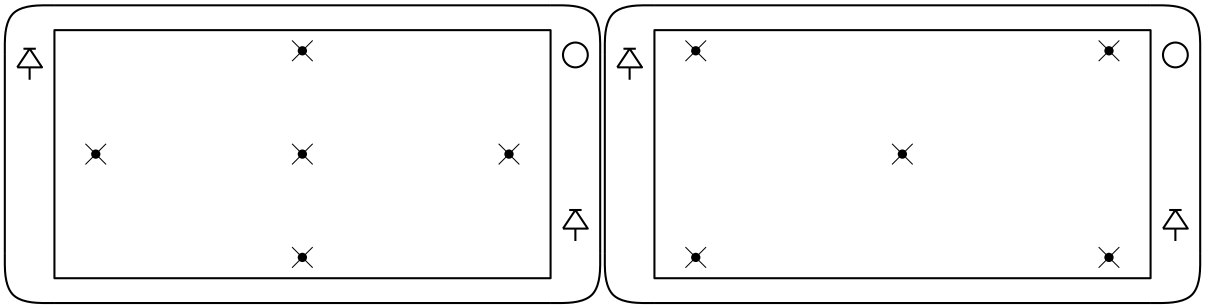
6.1 Supervised Static Depth

The first experiment is the least challenging, yet plausible smartphone eye tracking scenario: here a subject holds the device at some position and orientation, calibrates on the spot and then directly uses the eye tracker without intentionally moving or re-positioning it. In such a scenario there will still be minor natural head and device movements. This case will be referred to as the *supervised static depth* case.

The following experiment was conducted on eight subjects: each subject was instructed to hold the infrared prototype device, Littleboy (described in Section 3.1.2), in any position and angle at which they felt comfortable. Next, the experimenter helped them set the distance between the device and their head (z_{HCS}) to each of five specific distances; 20, 25, 30, 35, and 40cm.

At each distance two videos of the subject were recorded in which the subject was asked to look at five target locations for 50 frames each (for a total of 250 total frames captured). During the collection of the first video, which is used for calibration, the targets the five target locations formed the 'plus' pattern illustrated in Figure 6.1a. During the collection of the second video, which is used to generate gaze estimates to evaluate the gaze estimation process, the five target locations form an 'X' pattern illustrated in Figure 6.1b. The subjects were instructed not to intentionally re-position the cell-phone of move their heads in any significant way during the collection process. The collection of videos for post processing rather than generating live gaze estimates allows us to generate results using both the rule-based and machine-learning-based front-end feature extractor (described in the previous chapter) on the identical input data.

The results of this experiment is summarized in Table 6.1 and compared to existing mobile eye



(a) Five Target Plus Pattern

(b) Five Target X Pattern

Figure 6.1: Video Sample Capture Target Patterns

system	20cm	25cm	30cm	35cm	40cm	avg
GazeCapture	n/a	n/a	n/a	n/a	n/a	2.98°
ScreenGlint	n/a	2.56°	2.70°	n/a	2.06°	2.44°
Present Work (Rule)	0.57°	0.64°	0.69°	0.57°	0.68°	0.63°
Present Work (ML)	0.75°	0.80°	0.71°	0.62°	0.60°	0.70°

Table 6.1: Supervised Static Depth, **Gaze Bias** (degrees)

trackers that performed similar experiments. Table 6.1 presents the average gaze estimation bias at each static distance in degrees.

To compute average gaze bias we first compute the gaze bias at each of the 5 target locations individually using the same formula as presented for the feature extractor evaluation in eq 5.3. In this case x_i and y_i refer to the location of a gaze estimate on the display and the ground truth terms, gtx_i and gty_i, refer to the associated location of the target where the subject was instructed to gaze. For each subject the biases for the five targets were averaged and the result indicated the average Euclidean distance between the gaze estimates and the targets. The average gaze bias per subject was averaged across all 8 subjects. Finally the bias in millimeters was transformed into degrees using the known distance between the cell-phone and the subjects head. The results are presented in Table 6.1.

The GazeCapture system did not collect any data with subjects holding the device at known distances but an analysis of their dataset (see Appendix B) indicates that they used the cell-phone at a distances between 20-25cm from the subject's eyes. An approximate average gaze bias was computed based on this analysis and that is what is given in Table 6.1.

Table 6.1 shows that there is significant gaze bias improvement with our infrared system, both with the rule-based and ML-based feature extractor, when compared to the two non-infrared systems. The two prior systems achieved roughly 2.5-3° bias compared to the new systems with 0.6-0.7°. Notice that these results would suggest that rule-based feature extractor outperformed the ML-based feature extractor. The rule-based system is benefiting from the fact that calibration is done at every distance directly before the eye tracking session begins.

Table 6.2 presents the RMS error of the gaze estimates at each of the five distances when using the rule-based and ML-based feature extraction front ends. The average RMS is notably lower for the ML-based front end at 0.92° vs 1.51°. The greater number of large gaze estimation errors produced by the rule-based systems negatively impacts the RMS more so than it impact the bias calculation. The performance of each systems can also be seen by inspection when examining scatter plots of the final

system	20cm	25cm	30cm	35cm	40cm	avg
Present Work (Rule)	1.87°	1.39°	1.01°	1.23°	2.05°	1.51°
Present Work (ML)	1.16°	1.06°	0.88°	0.75°	0.74°	0.92°

Table 6.2: Supervised Static Depth, **RMS** (degrees)

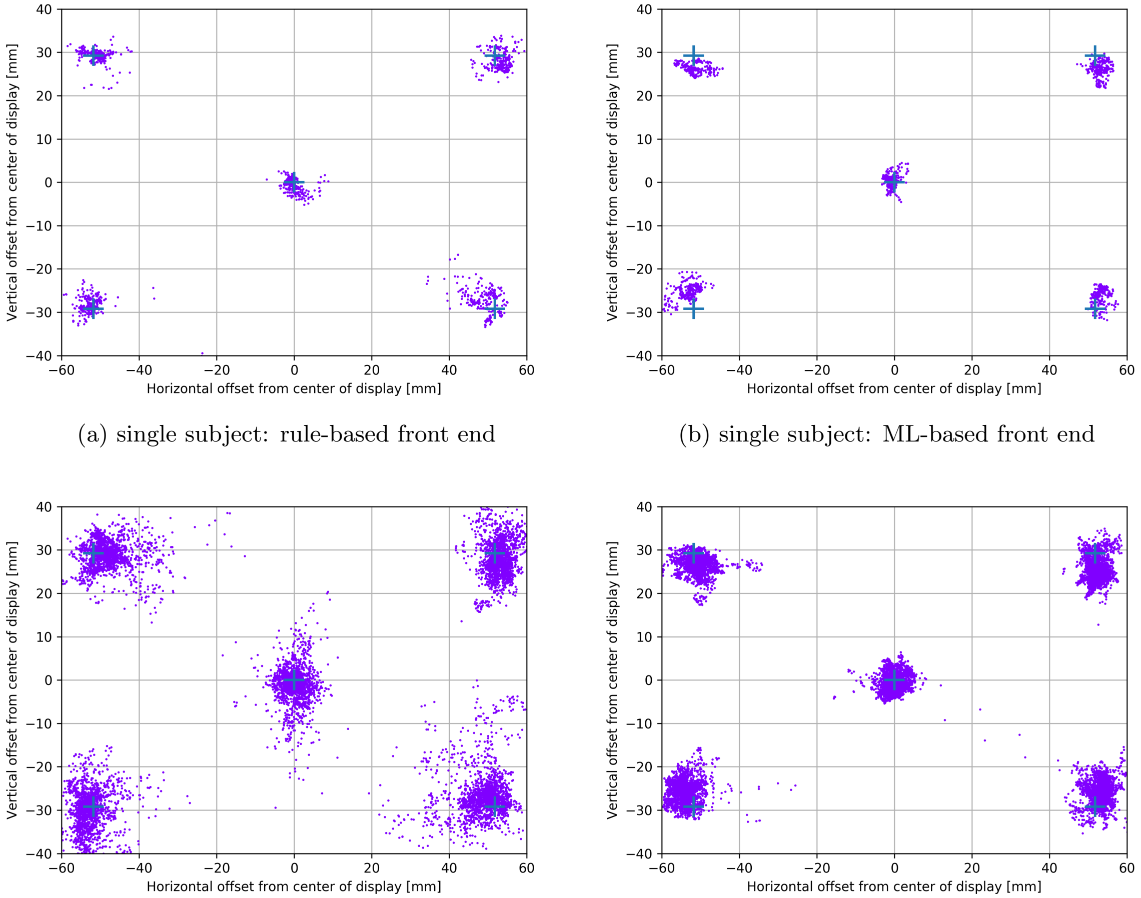
gaze estimates produced by each of the two configurations. The scatter plots in Figure 6.2 illustrates *each* gaze estimate as a single dot and the known gaze targets with ’plus’ signs. The horizontal and vertical axis of the figure represents displacement from the center of the smartphone display. Figure 6.2a and 6.2b display all the gaze estimates produced for a single subject (5 videos x 5 targets x 50 frames per target = maximum of 1250 possible gaze estimates) using the rule-based and the ML-based feature extractor front ends. By visual inspection, both methods produce good eye tracking results but there is more dispersion among the estimates from the rule-based front end. This additional dispersion is more apparent when we plot every gaze estimate for every video of every subject (maximum of 10000 possible gaze estimates), as shown in Figure 6.2c and 6.2d.

6.2 Supervised Dynamic Depth

The next experiment represents a more challenging, and more realistic eye tracking scenario. Here the subject calibrates at one device position and orientation, but eye gaze estimates are produced when the device position relative to the head varies. It is easy to imagine real-use scenarios in which a subject re-positions themselves in their chairs or switched the device from being held in the left hand to the right hand; each of these could significantly change the distance and orientation of the device with respect to the head. During these actions it is very desirable to continue functioning *without* the need re-calibration, as was implicit in the supervised static depth scenario above.

The evaluation procedure for this scenario is similar to the one above: At the beginning of the experiment subjects were instructed to hold the smartphone in any position and angle at which they felt comfortable. Then, the distance between the eye-tracker and their head was set to 30cm and a 5 target 250 frame video were captured and used for calibration. The same procedure was followed 5 additional times, once at each of the following 5 distance; 20, 25, 30, 35, and 40cm. These 5 video would be used to evaluate the eye tracker, but this time a single calibration result would be used across each of the 5 videos for a given subject. The same eight subjects from the previous experiment were used.

Table 6.3 presents the gaze estimation bias, in degrees, at each distance and compares that to the prior work mobile eye trackers. It is clear from Table 6.3 that both the rule-based and ML-based eye trackers developed in this work have much lower estimation bias than ScreenGlint or GazeCapture. The comparison between the rule-based and ML-based method also shows the improvement in robustness, as the ML-based eye tracker is able to maintain high gaze estimation accuracy in the presence of significant distance variation. Note that, at the calibration distance of 30cm, the rule-based method performed slightly better than the ML-based method with a gaze bias of only 0.57° gaze compared to 0.71° for the ML-based. However, changing the distance by 10cm resulted in significant performance degradation (bias as high as 1.90°) for the rule-based approach. Our ML-based eye tracker produced gaze estimation bias results with much smaller variation (0.65° to 0.81°) over the range of distances. Appendix C alternate versions of this experiment using different sets of selected networks which span a range of overall compute



(a) single subject: rule-based front end

(b) single subject: ML-based front end

(c) 8 subjects combined: rule-based front end

(d) 8 subjects combined: ML-based front end

Figure 6.2: Supervised Static Depth Gaze Estimates

costs.

In this experiment, in contrast to the supervised static distance case we discussed earlier, calibration only occurs at 30 centimetres. It is therefore not surprising to see both systems do well at that distance. The increased feature estimation bias of the rule-based feature extractor, discussed in Chapter 5, can be seen in the gaze estimation results in this experiment. A greater bias in the feature estimates will result in less physiologically accurate subject specific eye parameters produced during calibration. Instead, the calibration procedure will procedure subject specific parameters which compensate for the less accurate eye feature estimates to minimize the gaze estimation error at 30 centimeters specifically. As the device moves significantly away from the relative position in which it was calibrated, and the eye-tracker uses subject specific parameters which are not physiologically accurate, there will be an expected degradation in the gaze estimation accuracy. This is exactly what we see in Table 6.3.

Table 6.4 also presents the RMS of the gaze estimates at each of the five distances when using the rule-based and ML-based feature extraction front ends. The average RMS is still considerable lower for the ML-based front end at 1.17° vs 1.86°. This can also be seen clearly by inspection when examining scatter plots of the final gaze estimates produced by each of the two system configurations. Figure 6.3

system	20cm	25cm	30cm	35cm	40cm	avg
GazeCapture	n/a	n/a	n/a	n/a	n/a	2.98°
ScreenGlint	n/a	3.38°	3.32°	n/a	2.32°	3.01°
Present Work (Rule)	1.90°	1.15°	0.57°	0.90°	0.98°	1.10°
Present Work (ML)	0.81°	0.70°	0.71°	0.73°	0.65°	0.72°

Table 6.3: Supervised Dynamic Depth, **Gaze Bias** (degrees)

system	20cm	25cm	30cm	35cm	40cm	avg
Present Work (Rule)	2.27°	1.65°	1.00°	1.69°	2.67°	1.86°
Present Work (ML)	1.48°	1.36°	0.88°	1.07°	1.05°	1.17°

Table 6.4: Supervised Dynamic Depth, **RMS** (degrees)

again shows all of the gaze estimates for a single subject as well as for all subjects combined using both the rule-based and ML-based feature extractor.

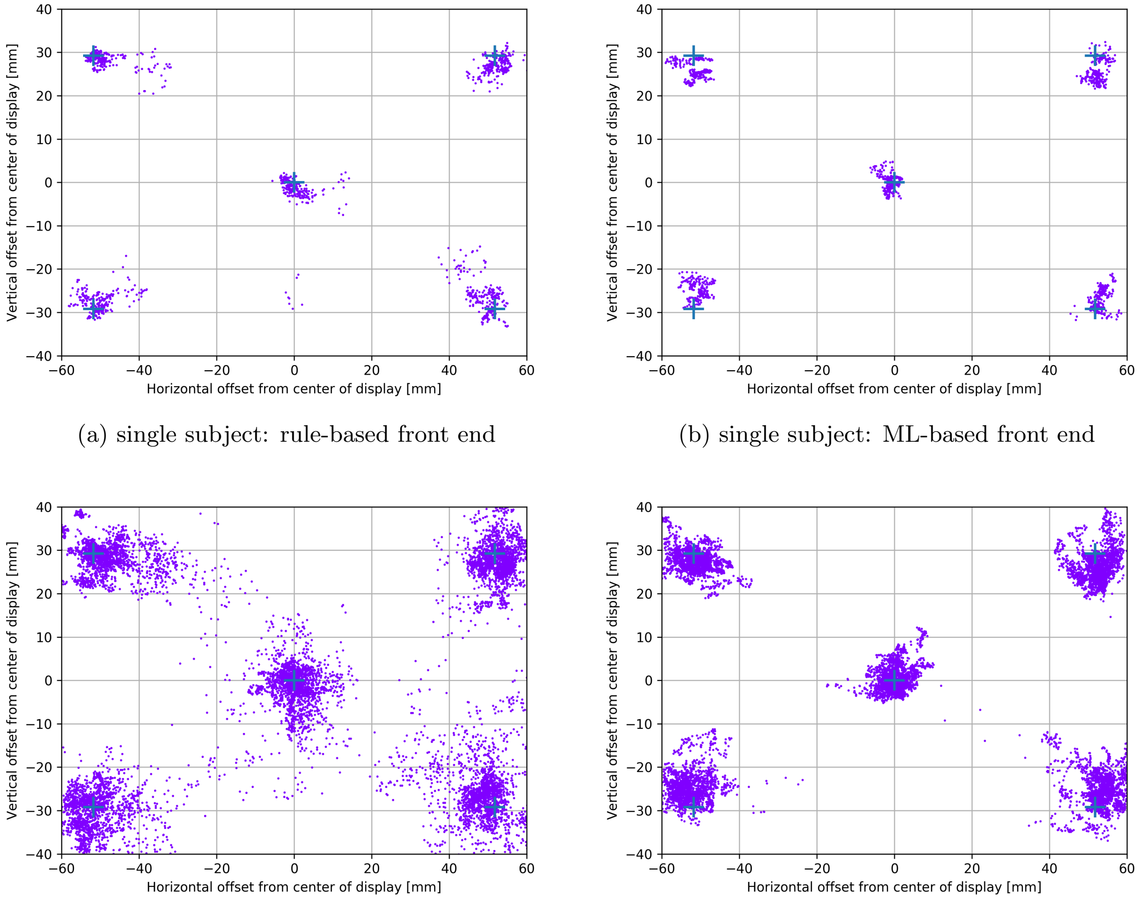
It is clear from Table 6.3 and 6.4 as well as Figure 6.2 and 6.3 that our hybrid machine learning-based front-end with an infrared model-based back end mobile eye tracking system has a much lower bias and estimation variance than other mobile systems in this more realistic scenario.

An individual breakdown of the gaze performance for each of the 8 subjects tested using our hybrid machine learning-based front-end with an infrared model-based back end mobile eye tracking system is shown in Table 6.5. This table also includes the eye color of each subject (light for blue hues, dark for brown hues). The contrast between the pupil and iris is worse for users with blue irises since their irises do not reflect as much IR as subjects with brown irises. This can lead to less robust feature extraction of the pupil-iris boundary and less accurate gaze estimates. The use of a CNN feature extractor trained on a wide range of iris colors mitigated this effect as the final gaze estimation errors are similar for both groups. The sample size of only two light eye subjects would need to be increased however to come to any definitive conclusions.

The data presented in this experiment show a marked benefit of our hybrid machine learning-based front-end with an infrared model-based back end mobile eye tracking system when compared with other mobile eye trackers. This experimental setup has several unrealistic characteristics, however. The experimenter helps the subject position the device at each distance and videos are captured in quick succession very shortly after calibration with the subject siting in the same environment with the same

Subject	Eye Color [dark/light]	20cm	25cm	30cm	35cm	40cm	average
01	dark	0.66°	0.58°	0.60°	0.59°	0.63°	0.61°
02	light	0.79°	0.81°	0.74°	0.66°	0.57°	0.71°
03	dark	0.85°	0.69°	0.62°	0.66°	0.51°	0.66°
04	dark	0.88°	0.75°	0.81°	0.83°	0.64°	0.78°
05	dark	0.73°	0.70°	0.72°	0.80°	0.62°	0.71°
06	light	0.79°	0.65°	0.62°	0.64°	0.72°	0.69°
07	dark	0.88°	0.74°	0.79°	0.83°	0.70°	0.79°
08	dark	0.95°	0.73°	0.81°	0.79°	0.78°	0.81°
combined	n/a	0.81°	0.70°	0.71°	0.73°	0.65°	**0.72°**

Table 6.5: HybridTrack Subject Specific Gaze Bias at Varied Eye Tracker Depth



(a) single subject: rule-based front end

(b) single subject: ML-based front end

(c) 8 subjects combined:rule-based front end

(d) 8 subjects combined: ML-based front end

Figure 6.3: Supervised Dynamic Depth Gaze Estimates

approximate posture. This does not reflect real-world unsupervised use.

6.3 Unsupervised Eye Tracker Use

This experiment was designed to evaluate the performance of the hybrid ML-based front-end with a model-based back-end infrared eye tracker from the perspective of a software developer who wants to make use of the eye-tracker in an application. When deploying an application that uses eye tracking, the user can be given guidance on to how to hold and use the device, but it will be used in an unsupervised environment where the position and orientation of the device are unknown. Additionally, when the eye tracking system indicates a subject is looking at a specific target, the application developer will be interested in the certainty of such an observation.

With this in mind, the following experiment was designed: the screen of the display was first segmented into a grid of boxes that are approximately square target regions. Then, in a random order each grid box (target) was highlighted, indicating that subjects should direct their gaze to that region, as illustrated in Figure 6.4. Following a 500ms delay after a target was highlighted, the system recorded

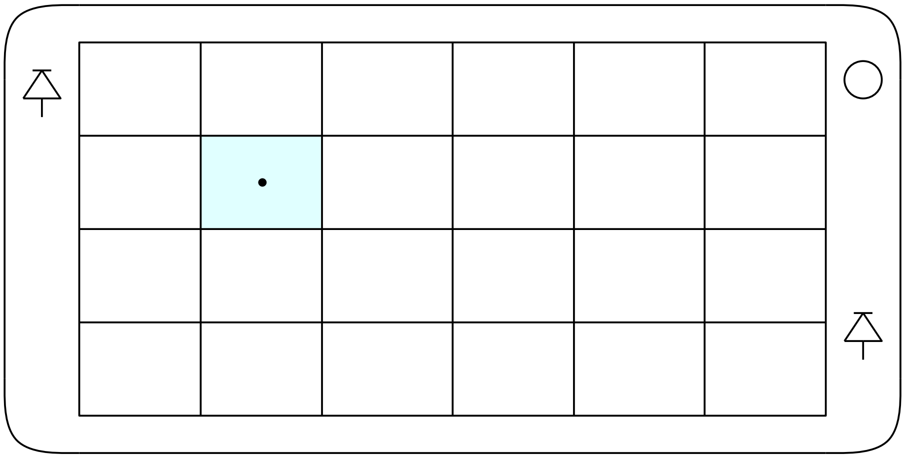


Figure 6.4: Smartphone Display Segmented into 6x4 Square Regions

15 gaze estimates. This procedure will be referred to as a grid test. One metric derived from this test is the percentage of gaze estimates that were inside the highlighted box. It is common, in eye tracking, to also compute a single gaze estimate from a sequence of N consecutive estimates, to reduce the error through averaging. We use values of N from 2 to 10. Unlike the previous two experiments, this test was not conducted with video recording but rather generated final gaze estimates live during the test.

The experimental procedure for the grid test was as follows: Each subject was given the device, and asked to hold it in a comfortable manner. Next, the calibration procedure was performed. The calibration parameters were stored and used in each subsequent session with that subject. Then, at three different times over the next 5 days, each subject was asked to perform the grid test four times, using four different grid sizes; 3x2 (38.3 x 32.5mm boxes), 4x3 (28.8 x 21.6 mm boxes), 6x4 (19.2 x 16.3mm boxes), and 8x6 (14.4 x 10.8 mm boxes). Once the instructions for the test were given, the remainder of the test was done with no further supervision.

Figure 6.5 shows, for different sizes of the grid ranging from 3x2 up to 8x6, the percentage of gaze estimates that were within the correct target box as a function of the filter length N. Each point on Figure 6.5 is the combined data for all 8 subjects for each experiment (i.e. a specific grid size and filter length). For example, consider the 6x4 case with a filter length of 4. Figure 6.5 indicates that this case had a 90% estimation accuracy. For a filter length of 4, given that we collected 15 gaze estimates for each target box, there are 11 filtered gaze estimates per target. An accuracy of 90% indicates that of the 2112 total filtered gaze estimates (11 filtered estimates per target x 24 targets per subject x 8 subjects) about 1900 of them were located in the correct corresponding target boxes.

For the 3x2 grid Figure 6.5 indicates that 95-100% of all gaze estimates (depending on N) were inside the highlighted boxes. The percentage drops with each increase in grid dimension and corresponding decrease in box size: 89-96% accuracy was achieved for the 4x3 grid, 83-93% 6x4 grid and 66-76% for the 8x6 grid. The performance difference between the 6x4 and 8x6 grid size indicates the limit of the effective performance of the system and testing smaller grid sizes would not be valuable.

Figure 6.6 shows the results of the grid test on a 8x6 grid, by giving the percent correct estimates at each grid location. These results are across all 8 subjects with N (the filter size) set to 10. While the average across all boxes was 76% one can observe significant variations in performance for specific locations on the screen. Gaze estimates in the central area of the display are more likely to be correct,

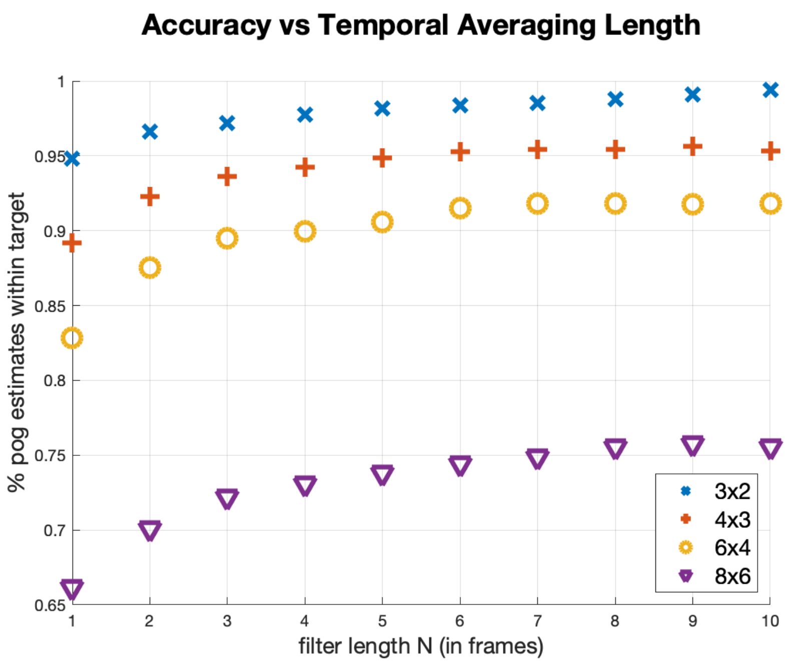


Figure 6.5: Target Identification Accuracy vs Filter Length

with percentages in the high 70s to low 80s. Estimates closer to the corners of the display, can be quite a lot less, with the worst occurring at the bottom left corner box, with only 38% being correct within the box. We suggest that future eye trackers include this level of detail when evaluating a system, rather than a single average.

We hypothesize that the principal explanation for the lower accuracy's is associated with the physical locations of the LEDs on the prototype smartphone. The two LEDs on our prototype are located near the top right and bottom left of the display and these are the two regions with the worst performance. When looking at those regions, one of the corneal reflections, being furthest from the pupil center, may not be created solely by the spherical region of the cornea. The 3D model [40] we used assumes that the two corneal reflections are created by a spherical cornea, and so this can result in additional gaze bias. One solution to this problem employed on desktop systems is to have more LEDs (more than two) and to select the two corneal reflections closest to the pupil center when estimating the gaze. This solution might be less desirable on a smartphone due to power and space constraints.

6.4 Summary

Integrating the contributions outlined in Chapter 4 and 5 in a real physical device we have demonstrated a new hybrid eye-tracking system with a gaze estimation bias of 0.72° in realistic unsupervised scenarios. This performance is achieved with a single calibration, operating over a realistic range of device positions (20 - 40 cm) and orientations. When handing the device to subjects with little to no instructions, the system could achieve better than 90% accuracy distinguishing between targets that are 20 millimetres apart (6x4 grid, using simple averaging of three or more gaze estimates). Our hybrid eye-tracking system (which has the advantage of using artificial infrared illumination) achieves significantly better

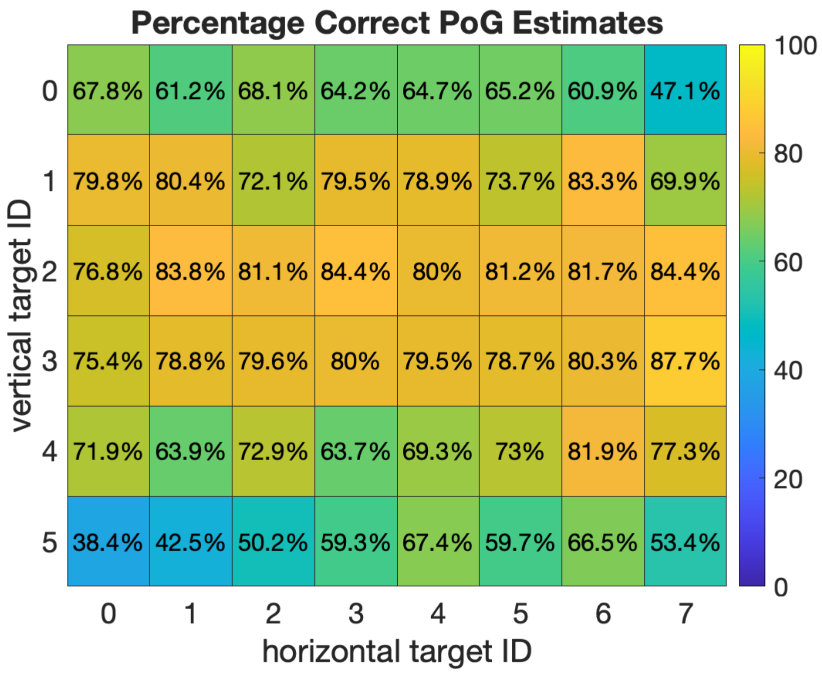


Figure 6.6: Spatial Distribution of 14mm Target Identification

accuracy (more than 400% better) than the accuracy achieved by previous systems that use natural light and appearance-based methods to estimate gaze position. These results were obtained during use of the eye tracker in indoor environments with subjects holding the device within the constraints outlined in Table 5.1. We would expect performance deterioration outside of these constraints.

Chapter 7

Future Work and Conclusion

Many avenues are available to extend and improve the capability of the novel mobile eye tracking system that was presented in this work.

7.1 Glasses

All of the testing and analysis of the eye tracking system and components presented in this work was performed with subjects not wearing eye glasses. Figure 7.1 illustrates the reason for this as eye glasses can result in large infrared reflections which can completely occlude eye features. This is particularly a problem in a smartphone environment where the camera, LEDs, and screen are co-planner and subjects naturally tend to orient the device at or near parallel to the plane of the face.

Figure 7.2 shows the zoomed in eye regions from the head regions depicted in Figure 7.1. In some cases there are no reflections, or the reflections are far enough from the pupil that they would have no impact on the current feature extraction techniques. In other cases the reflections are near or slightly occlude the pupil, which can strongly affect pupil-iris gradients and make it significantly more challenging to estimate the eye features. Finally, these reflections can in some cases totally occlude the pupil which will prevent any of the eye features from being captured.

In the stationary desktop eye tracking systems, these types of reflections are less common due to the relative orientation of the camera, subject, and light sources. A common solution is to have more than two infrared light sources and then for each head/device position rotate between the pairs of LEDs that illuminate the face until one gets an image of the eye that can be processed. In this way, for a given

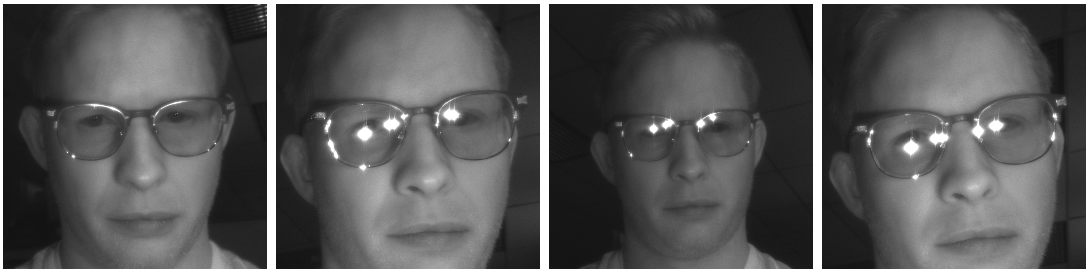
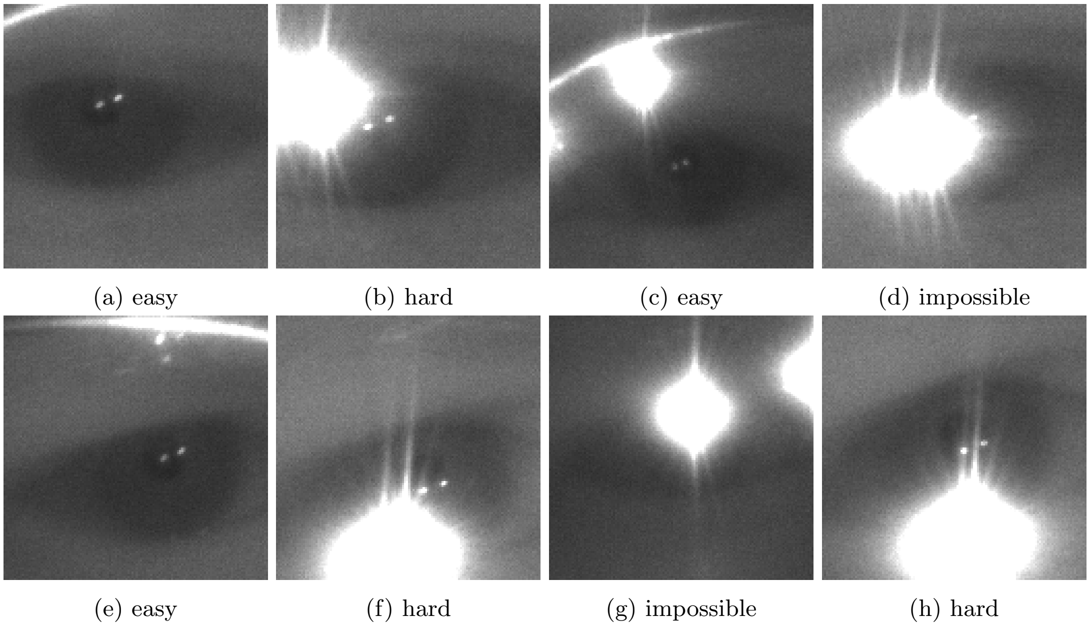


Figure 7.1: Head Region Images with Glasses


(a) easy (b) hard (c) easy (d) impossible

(e) easy (f) hard (g) impossible (h) hard

Figure 7.2: Difficulty in Feature Extraction for Eye Region Image with Glasses

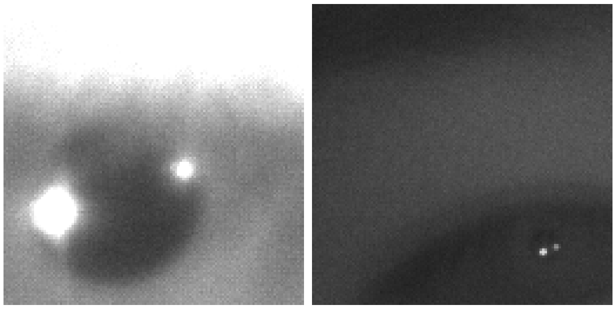
positioning of subject and system, there will hopefully be at least one pair of light sources which don't cause total pupil occlusion. This may be a useful approach on mobile system as well but would require a different hardware configuration than the one available for this work.

It remains to be seen just how much of a problem glasses would cause in practice for a two-infrared LED smartphone use case however. Working with partially occluded eye regions like those shown in Figure 7.2b, 7.2h, and 7.2f would be challenging with a rule-based feature extractor, especially considering that developing one without these types of reflections was in itself difficult. The pupil center and corneal reflections are both visible in these two examples however and could be annotated by a human. This means, at a fundamental level, that the information required to locate these features are available and there is no reason to believe a similar neural network-based approach would not succeed.

A key element of future work on this front is to collect a second dataset of subjects with eye glasses. One would first analyze the percentage of eye regions that would classify as impossible to process and determine how often that case occurs on both the left and right eye simultaneously since only a gaze estimate from a single eye is needed to determine the point-of-gaze for a person with healthy visual system. The goal would then be to determine the impact of infrared reflections from the glasses on the accuracy of the front-end feature extraction. This would determine the impact that glasses have on the end-to-end eye tracking experience.

7.2 Dynamic Illumination Power

Recall that one of the principal sources of image variation in mobile eye trackers is a result of large changes in the position and orientation between the head and the device during use. Both the area of facial features in the captured images and (if the infrared LED power output remains constant during operation) the amount of infrared light which illuminates the eye regions is inversely proportional to the square of the distance between the device and the eyes.



(a) Close (b) far

Figure 7.3: Eye Region Illumination Difference with Static Illumination Power

Figure 7.3 illustrates an example difference between an eye region captured when the device was about 15cm from the head vs another subject with an eye region captured at about 45cm from the head when the illumination power remains constant. For close device positions, such as in Figure 7.3a, large corneal reflections can significantly interfere with the boundaries of the pupil. In these cases lowering the total illumination power on the device would reduce this interference but would maintain a strong pupil-iris gradient and should make estimating the pupil centre more accurate. In far device position images, such as Figure 7.3b, both the pupil is smaller and the gradient between the pupil and iris is weaker which makes accurate pupil estimation more difficult. In these cases increasing the total illumination power should make estimating the pupil center more accurate.

Dynamically adjusting the total infrared illumination power could be used to achieve a number of different objectives. It could help increase the operating range of the mobile eye tracker, and within any given operating range it could improve the overall eye region image quality which may improve feature estimation and (by extension gaze estimation) accuracy. Conversely, it could be used to lower the illumination power as long as the eye region classifier is still validating the eye regions. This would result in most eye regions being quite dim and feature estimation accuracy suffering, but would decrease the total power requirements of using the mobile eye tracker.

A model which adjusted the illumination power output based on the distance between the device and the head would only require modest accuracy in the depth estimation of the head. Using a combination of measurements of the inter-pupillary distances and the position of the centre of rotation of the eyes (both of which are computed during the gaze estimation process) should be more than sufficient for estimating device distance to within 5 centimeters.

7.3 Counter Roll Model

As discussed in Section 4.5 the estimation of the relative roll angle between the device and the eye is required to accurately estimate the visual axis of the eye. This would ideally be done by tracking the roll of the eye directly (using the iris pattern) but this seems unrealistic on modern smartphones given the typical resolution of the front-facing cameras.

Our proposed method first uses a head tracker to measure the relative angle between the device and the subject's head. We assume that the subject is sitting or is in a near upright position with respect to gravity (i.e. in a chair). For a truly mobile eye tracking experience a user should for example by able lie down horizontally on a couch or bed. In this extreme case ocular counter roll becomes a significant source of error and could result in gaze estimation error increases of between 0.2° to 0.9° depending on subject specific eye parameters (the deviation between the optical and visual axis). Even at the low end of this range – 0.2° – additional gaze estimation error would be a 25% increase from the current accuracy of our eye-tracking system (about 0.8°).

In this situation the estimation of the relative roll angle between the device and the eyes alone will fail to account for ocular counter roll (which is a function of the tilt of the head with respect to gravity). To measure the relative roll in this case one can use a measure of the tilt of the head relative to gravity in a ocular counter roll model to estimate the counter roll of the eye. Then one can use the accelerometer to measure the tilt of the phone relative to gravity, the head tracker to measure the tilt of the head relative to the device and the model output to estimate the counter-roll to estimate the relative roll between the eye-tracker and the subjects head.

The amount of ocular counter roll is non-linear and varies between subjects. Any attempt to integrate counter roll into the estimation of the relative roll angle would require first developing a counter roll model. This may start with a fixed model which is not subject-dependent, but eventually involve estimating some subject specific counter roll parameters during a calibration procedure.

7.4 Commercial Devices

At the time this work began there were no commercially available smartphones which had the hardware necessary for infrared model-based eye tracking. As a result this work was done on an infrared enabled prototype device provided to us by Huawei, which included a single infrared camera and multiple infrared LEDs. We speculated, at the time, that infrared cameras and infrared LEDs would eventual make their way into commercial devices for use with biometric security tasks and gesture tracking/interactions tasks. This prediction has come true as high-end devices from the two largest device manufactures, Apple [50] and Google [49], both contain some form of infrared camera and illumination which they use to aid in facial unlock. This hardware is not accessible to the average developer however and therefore cannot be used by third parties to explore different types of functionality, such as eye tracking.

Google's new flagship smartphone, the Pixel 4 [49], follows a similar direction to the Apple devices by including infrared cameras, light sources and radar sensors into their device 'notch' cutout at the top of the display. An illustration of this hardware is shown in Figure 7.4. If it was possible to access this hardware on the Pixel 4 it would be great device for implementing the infrared model-based eye tracker similar to what was presented in this work.

Figure 7.4 indicates the Pixel 4 has a single infrared flood illuminator (LED) and two infrared cameras, in contrast to what was presented in this work which used two infrared LEDs and a single infrared camera. Both of these cases can be used to estimate the optical axis of the eye in the presence of head and device movement [40] (using a different set of equations) once a calibration procedure has been completed to estimate some subject-specific parameters of the subject's eyes. Once the optical axis is estimated the visual axis can be estimated using the model extension presented in Chapter 5 for accurate gaze estimation on a smartphone display which can exhibit relative roll between the device and the subject's eyes.

We believe it is still only a matter of time before the infrared hardware will be made available to third party developers on some device. If and when that happens it will be possible to test the performance of the techniques presented in this work commercially available hardware that could be easily deployed

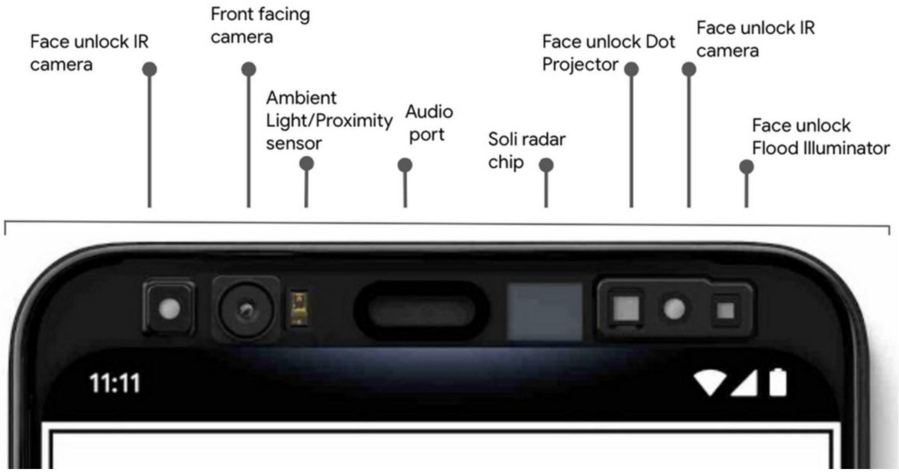


Figure 7.4: Google Pixel 4 Smartphone Face Unlock Hardware

to perform in a variety of eye tracking applications.

7.5 Conclusion

Eye-tracking has been used in a wide variety of domains and represents an increasingly useful technique for an extensive array of medical applications. Enabling it as a sensor on a modern smartphone with the accuracy and robustness required to enable these applications is challenging. A key challenge for smartphone eye-trackers is the presence of sizeable relative motions between a hand-held device and the subject's head that lead to high variability in eye-image appearance. This work has presented contributions in both the front-end image processor and back-end gaze estimation model to address these challenges.

A 3D gaze estimation model was extended to include invariance to the relative roll (R-Roll) of the device in a subject's hands. Using this new model we demonstrated an average decrease in the gaze estimation bias of four subjects by over 2.5° during an R-Roll of 90°. Even during less extreme, unintentional amounts of R-Roll the use of this new model dramatically increases overall gaze estimation performance. Invariance to this type of motion is critical for hand-held eye trackers.

Robust and efficient determination of the location of eye features that are required by the back-end model to estimate the point-of-gaze are difficult to obtain due to the motion of a smartphone. Using a hierarchy of small CNNs and a novel center of mass regression output layer we demonstrated the capability to estimate eye features from a representative mobile dataset with 0.5-0.7 pixel accuracy and nearly zero bias. We were able to achieve this level of performance while requiring 60 megaflops of total computation per eye region, which is well within the capabilities of modern neural accelerators engines on current smartphones.

Combining both back-end and front-end contributions, we built a new hybrid eye-tracking system producing gaze estimates with an average bias of 0.72° degrees across 8 subjects in realistic unsupervised scenarios. This performance is achieved with a single calibration, operating over a realistic range of device positions and orientations with little to no instruction. Our infrared mobile eye-tracking system achieves 400% better gaze estimation bias than state-of-the-art eye-tracking systems that use natural light and appearance-based methods to estimate gaze position. This work has aimed to demonstrate what is possible with a mobile infrared system but acknowledges that wider deployment of such a system would require a significantly larger set of training and testing subjects to validate and refine the system.

The mobile infrared eye-tracking system developed in this work was shown to produce accurate and robust gaze estimates, in challenging scenarios. This level of performance begins to approach that of desktop eye-tracking systems, which motivated this work [3, 4, 40]. If smartphone device manufactures utilized the existing infrared hardware on commercial devices for eye-tracking, then smartphones have the potential to become useful for the assessment of cognitive, psychiatric, and neurological states of individuals.

Appendices

Appendix A

Android Infrared USB Camera Bar

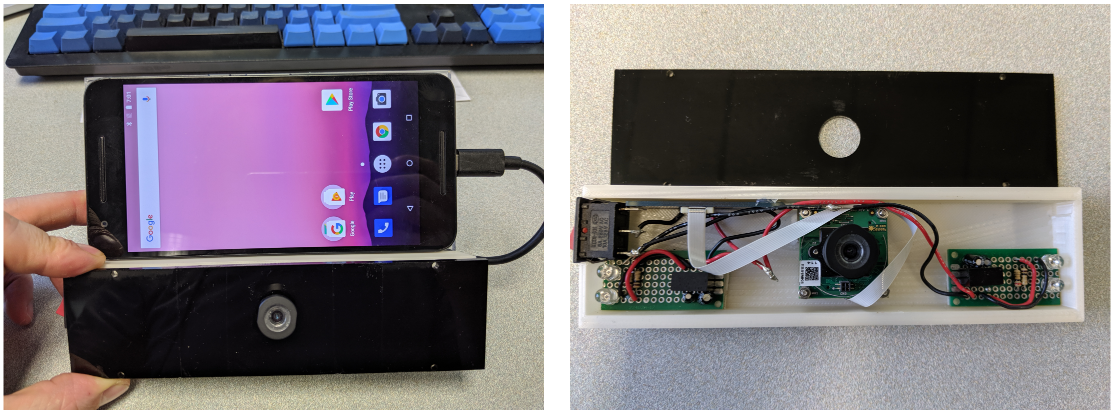
One method to enable IR eye-tracking on commercially available smartphones is to develop a USB external camera bar. This attachment would include at least two infrared LEDs and a USB camera which can capture infrared images. Such a bar was needed because no existing commercial smartphones have developer accessible infrared front-facing IR cameras or LEDs.

The benefits of an external camera bar are that they would be controlled and built by us to our specifications and can, in theory, be used interchangeably with many different smartphones to enable infrared eye-tracking. Additionally, we can manufacture as many as needed for future experiments (which would not be the case for the integrated industrial prototype). Disadvantages stem from the limitations and difficulties of communicating with a camera over USB on Android. However, the benefits of having control of the platform from a research perspective made this approach worth pursuing.

A.1 Camera Bar Hardware

Figure A.1 shows the camera bar we constructed for this work connected to a stock Android smartphone as well as the internal layout. The black plate over the top of the bar is transparent to infrared and do not impede infrared LEDs from illuminating the subjects face. We constructed a camera bar for this work which consists of an Econ Systems USB 3.0 monochrome 5.0MP camera [186], a 65° field of view lens, and two sets of infrared LEDs on either side of the bar. The camera resolution was initially limited by USB bandwidth to 1280x720 pixels at 12 frames per second (fps) over a USB 2.0 connection (most smartphones circa 2015-16). The performance would improve to 1920x1080 pixels at 30 fps on modern devices when they began to support the USB 3.0 specification.

Our camera sensor has an optional global shutter release with hardware strobe, which means we could enable the external LEDs only momentarily during frame captures. This operation both significantly saves energy (compared with leaving the lights on continuously) and minimize image smearing during camera motion. Without strobing, one would expect high amounts of image smearing when holding a device in the hand which moves. The monochrome nature of the camera sensor is advantages as the larger pixel pitch (from not needing to split pixels into separate regions to collect red green and blue light separately) results in and better low light performance. Low light camera performance is a concern since external pins on the camera provide power form the USB bus for the two LEDs. The USB specification [187] for this class of device only allows for 400mA of continuous power draw at 5v. The



(a) Connected to smartphone

(b) Internal layout

Figure A.1: External USB infrared camera bar

camera itself consumes about 250mA leaving 150mA left to power the infrared LEDs.

Importantly the camera was also a USB Video Class Device (UVC [188]) which is a standard interface for interacting with USB cameras. Any camera with propriety device drivers would not have had an Android driver in 2015-16, so the UVC interface was the only practical communication interface. Before that can be discussed however a brief primer on android development is useful.

A.2 Android Development

Developing high-performance applications on Android smartphones is a more complicated process than on a typical desktop computer. A small primer on Android development is useful before discussing the details of the camera bar communication interface. On Android, the primary software-development language for user applications is Java. Applications are written in Java then converted to bytecode and interpreted by the Android Runtime (ART [189]). ART is conceptually similar to a Java Virtual Machine (JVM [190]) except it is register-based and optimized for a reduced memory footprint.

Interpreted bytecode is excellent for software portability, and this is one reason Android is found running on so many kinds of devices. However, it negatively impacts performance when compared with natively compiled instructions for a specific CPU architecture. To address this limitation, Google released the Android native development kit (NDK [191]). The NDK enables c/c++ functions to be natively compiled for common CPU architectures. When using the NDK computationally-expensive functions of Android applications can be written in c/c++, with some limitations. Firstly, the NDK contains only a subset of the c++ standard library. Therefore, large c/c++ libraries often can not cross-compile without some modifications to remove the usage of non-supported standard functions. Secondly, there is a communication and code-complexity overhead when passing data by reference between Java functions and natively-compiled functions using the Java native interface (JNI [192]). The performance advantage of a native implementation for advanced image-processing tasks, including all overheads, is more than five times [193]. JNI and the NDK are necessary for our work, despite the increase in technical complexity. The Android external camera interface relies on both the JNI and the NDK.

A.3 Communication Interface

One of the essential properties of Android, during the planning stages of our work, was USB On-The-Go (OTG) support [194]. USB OTG is a specification which enables USB devices (such as tablets or smartphones) to act as a host and let other USB devices (such as flash drives or cameras) attach to them. USB OTG support is required to attach and communicate with a USB camera from a smartphone. It was this feature that allowed us to communicate with a UVC camera as a host on Android.

On a desktop version of Linux, opening a UVC camera and accessing a stream of frame data is a simple process. Contained in the Linux kernel is an implementation of the Video for Linux (V4l2 [166]) driver, a robust fully-featured camera driver which supports UVC devices. Using a user-mode interface library called libv4l2 a ‘hello world’ program to access the camera is just a handful of lines. The kernel of Android does include V4l2, and it is used by the operating system under the hood when accessing the native on-device camera. The Android NDK does not provide a user-mode libv4l2 library and linking directly against v4l2 interface from native application code requires kernel-mode permission which Android does not grant to user applications. We, therefore, needed to find and compile a complete end-to-end user mode UVC camera driver.

Additionally, we must contend with the fact that the security model on native c++ code in Android apps is far more strict than a standard desktop version of Linux. Permissions to read or write to the filesystem, open/close USB devices, or interact with any memory-mapped IO must be explicitly granted (with interaction from the user) in the Java layer. After the user grants the permission, the paths or memory address can then provided through the Java Native Interface (JNI [192]) to native code. Most large user mode c++ libraries that have any interaction with the host system won’t work when directly compiled for Android and require modifications to interact with the JAVA layer and adhere to the stricter security policies.

Our Android USB camera driver combined two main libraries. First, a young cross-platform open-source UVC camera driver called libuvc [195]. Second, libuvc’s main dependency a well-established user-mode cross-platform USB communication library called libusb [196]. These libraries combined are on the order of 100,000 lines of code and at the time of developing our Android camera driver neither had been tested on or included build files for Android (both do now). Due to additional security policies of Android and the NDKs lack of full C++ standard library support, we needed to debug many issues in the processes cross-compiling these libraries.

Once we had these libraries modified and compiled, we still cannot just call the libuvc open() function to open the camera as we would on Linux. This is because native c++ on Android does not have permission to access USB devices. The following steps are required to adhere to the increased security policies and access the camera from our app:

1. Application Java code scans the USB device chain to find the camera.
2. The user is asked explicitly to allow the app to access that USB device by clicking yes to a system pop up dialog.
3. The USB device is opened, and a device descriptor is returned.
4. A pointer to the USB device descriptor is passed to native code via the JNI.
5. The USB device chain is scanned again to find the raw file path to the USB device (this is not visible during the scan at the Java layer)

6. A modified version of the libuvc (and subsequently libusb) open() function is called. This version, rather than opening the device, initializes internal data structures with the device descriptor and USB device chain address provided as input.

7. Then, use libuvc as one would on a standard Linux machine to access frames data.

This approach worked until the end of 2016 with the release of Android version 7.0 [165]. This version tightened access restrictions to USB devices in C++ and broke our methodology. Therefore on an old phone (without Android 7.0), the driver would work but only at USB 2.0 speeds while new phones with USB 3.0 (which would run Android 7.0) would not work. In mid-2017, after a significant amount of searching for workarounds, we decided to abandon the external camera bar and focus on LittleBoy for our work. Without the ability to use the newest commercial devices with both USB 3.0 speeds and the newest versions of Android, the camera bar setup was just a less portable version of Littleboy.

In late 2018 Android released official external UVC camera support (Although we only became aware of this in 2019) which internally used the kernel-mode V4l2 drivers we had wanted to use from the start. If a similar project to this thesis is developed in the future connecting to an external USB camera bar on Android would not be a significant undertaking.

Appendix B

GazeCapture User-Camera Distance Approximation

The mobile eye-tracking system named GazeCapture [46] is a crucial point of comparison for our work. The unsupervised crowd-sourced videos used in that work prevented the measurement of the distance between the user and the phone. The results in that work described average gaze estimation errors in centimeters, rather than in degrees which is traditional in the eye-tracking world. Measurements in degrees are more general as they are less dependent on the distance between the display and the user.

We can estimate the distance each subject was from the device using the inter-pupillary distance (in pixels, and mm), the resolution of the image, the field of the view of the lens and simple trigonometry. To understand the operating range from the GazeCapture publication, we analyzed 120 sample images from that dataset. In 118 of the samples, both of the subject's left and right eyes were visible, and we measured the inter-pupillary distance (in pixels) directly. The inter-pupillary distance in millimeters was assumed to be the average physiological distance for males and females of 64.7mm and 62.3mm, respectively [197]. The field of view on standard iPhone models supported at the time ranged from 54.2° to 64.2° when capturing images at the highest available resolution. Taking a mean between these two values gives us a field of view of 59.2° which we used in the distance computation.

An approximate distance for each of the 118 subjects was computed using this methodology, and Figure B.1 shows a histogram of these results. The average distance of all samples was 21.5cm with 80% of samples having the device positioned less than 25cm. Being conservative, we use a 25cm distance when converting the GazeCapture results to degrees before comparing the results with our work.

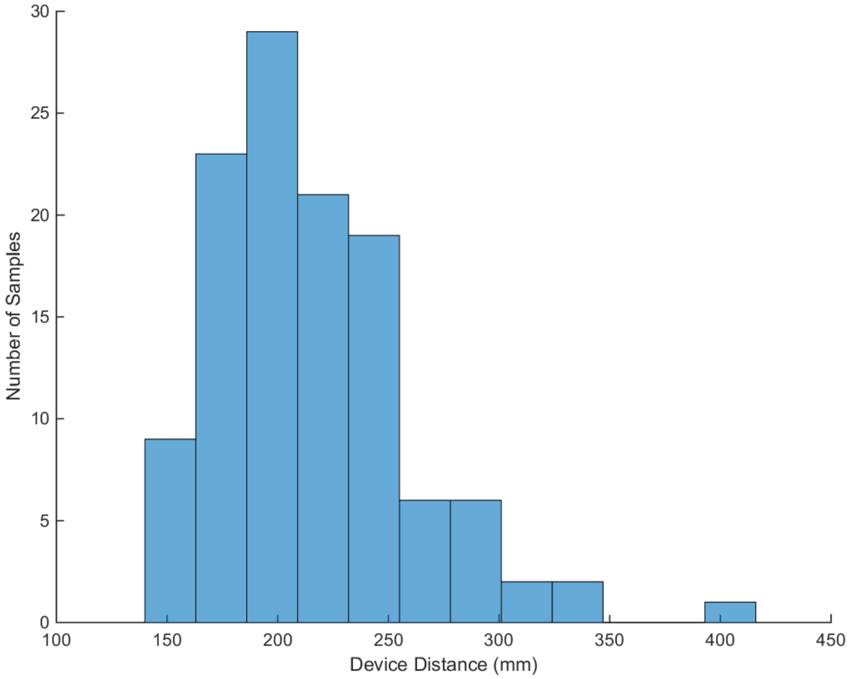


Figure B.1: Histogram of GazeCapture distances

Appendix C

Alternate Networks

A single set of neural networks were chosen for use in our hybrid eye-tracking system presented in this work. Those networks were chosen to select a good trade-off between performance and computational cost of the forward inference. In this Appendix we will present alternative selections, at different levels of computational effort, and show the impact on feature extractor and gaze estimation performance that would be achieved.

C.1 Network Sets

The six networks that comprised the ML feature extractor presented in Section 5.4 were named as follows: *pupil coarse*, *pupil medium*, *pupil fine*, *corneal ref a*, *corneal ref b*, and *classifier*. Of these networks the first two (*pupil coarse* and *pupil medium*) are only used to help localize the eye region when frame to frame tracking fails. They will be ignored in this analysis, and we will focus on the 4 networks which impact performance. All four of these networks were trained with the 64x64 image crop dataset since they are run after the localization process.

Three sets of networks were selected for further analysis (in addition to the set used described in Chapter 5.4). These sets are labelled based on the relative overall forward inference computation cost as *high*, *good*, *low*, *very low*. With the *good* set of networks being the set used in the Chapter 5. The amount of image scaling and the specific architectures for each set of networks is presented in Table C.1.

C.2 Feature Extractor Performance

Each of the four sets of networks presented in the previous section were tested in an identical manner to what was described in Section 5.4 using the validation set of images from the dataset produced for this thesis. The performance of each set is presented in Table C.2.

The classification accuracy across all four classifiers ranged from 94.2% to 98.9%, which is in line with our expectations of the labeling uncertainty. The pupil estimation accuracy ranged from 1.03 to 0.67 pixels at computational cost range between 2.58 to 620 megaflops. Similarly the corneal reflection estimation accuracy ranges from 0.74 to 0.49 at a computational cost between 0.93 to 476 megaflops. There is a 35% improvement in all the features estimation accuracy from the smallest network to the

network	scale	cnn l1		cnn l2		dense l1		dense l2		mflops)
		k	w	k	w	n	dropout	n	dropout	
pupil fine	2x	8	9x9	n/a	n/a	256	0.5	64	0.5	2.58
corneal ref a	2x	8	5x5	4	3x3	128	0.4	n/a	n/a	0.93
corneal ref b	2x	8	5x5	4	3x3	128	0.4	n/a	n/a	0.93
classifier	4x	4	3x3	n/a	n/a	128	0.5	128	0.7	0.13

(a) 'Very Low' Computational Cost Set

network	scale	cnn l1		cnn l2		dense		mflops
		k	w	k	w	n	dropout	
pupil fine	1x	16	4x4	8	3x3	128	0.5	6.26
corneal ref a	1x	4	4x4	8	7x7	128	0.4	7.02
corneal ref b	1x	4	4x4	8	7x7	128	0.4	7.02
classifier	1x	8	3x3	n/a	n/a	256	0.4	4.85

(b) 'Low' Computational Cost Set

network	scale	cnn l1		cnn l2		cnn l3		cnn l4		dense		mflops
		k	w	k	w	k	w	k	w	n	dropout	
pupil fine	1x	8	4x4	32	6x6	8	3x3	32	6x6	1024	0.7	32.0
corneal ref a	2x	64	5x5	8	3x3	64	5x5	n/a	n/a	1024	0.5	11.6
corneal ref b	2x	16	4x4	48	3x3	48	6x6	n/a	n/a	1024	0.4	12.6
classifier	1x	8	7x7	n/a	n/a	n/a	n/a	n/a	n/a	1024	0.4	20.1

(c) 'Good' Computational Cost Set

network	scale	cnn l1		cnn l2		cnn l3		cnn l4		dense		mflops
		k	w	k	w	k	w	k	w	n	dropout	
pupil fine	1x	128	7x7	64	5x5	128	5x5	24	8x8	2048	0.6	620.5
corneal ref a	1x	64	5x5	128	4x4	48	6x6	128	9x9	256	0.5	476.2
corneal ref b	1x	64	5x5	128	4x4	48	6x6	128	9x9	256	0.5	476.2
classifier	1x	24	4x4	64	6x6	n/a	n/a	n/a	n/a	256	0.5	125.1

(d) "High" Computational Cost Set

Table C.1: Network Architecture Sets

largest. However, the accuracy of larger models is likely much better than indicated by this metric, for the reasons described below.

The feature location labeling, done by a human, will indicate the lowest value we could expect to be produced for the average estimation error metric. For example, imagine a scenario in which a network existed that could perfectly estimate the precise center of every pupil with zero error. If the average pupil labeling uncertainty was 0.5 pixels in the validation set than the average pupil estimation error would be computed as 0.5 pixels (even though in reality it was 0). Therefore as the average pupil estimation error metric for a network approaches the labeling uncertainly the *true* error can be thought to be approaching zero. The dramatic increase in inference cost between the *good* and *very high* networks with only very marginal improvements in feature estimation error indicates networks which are approaching the uncertainty in the feature labeling by our human annotator and therefore the *true* error has likely fallen dramatically.

Interpreting the numbers in Table C.2 also requires an understanding of the imbalance in the eye region dataset used for training and validation. Due to the method used to collect the eye region image dataset there will not be an equal number of images across all distances and poses. There are many more images with the device held at 25-35 centimeter distance than held at 20 or 40 centimeters (which many find uncomfortably close/far). Therefore, as the trained networks make significant improvements in estimating features for eyes that are very close, or very far away the impact on average estimation error across the whole validations set will be small. However, when a subject is instructed to hold the device at either 20cm or 40cm for end-to-end eye tracking experiments these feature estimation improvements will be essential for producing high quality gaze estimates.

In the next section each of the four sets of networks will be evaluated with respect to their end-to-end eye tracking performance in the hybrid eye-tracker presented in Chapter 6.

C.3 Eye-Tracking Performance

Each of the four sets of networks were tested in an identical manner to what was described in Section 6.2 using the videos from eight subjects each holding the phone at one of five distances, with a calibration procedure being performed on a separate video taken at a subject distance of 30 centimeters. The average gaze bias and RMS of the gaze errors are presented in Table C.3.

Moving from the *very low* to the *low* network sets reduced the average gaze bias only marginally from 1.95° to 1.87°. The *very low* set performed relatively poorly with a 3.35° gaze bias at 40cm, while the *low* set performed relatively poorly with a 2.46° at 20cm. The transition to the *good* network set demonstrates dramatic improvements in gaze estimation bias across the board with an average of 0.72° and no specific distances that show relatively poor performance. This is why networks of that size were selected for use in Chapter 5. The final *high* set shows virtually no improvement in gaze bias when compared to the *good* set, but does reduce the variance in the gaze estimates slightly improving from 1.17° to 0.87° RMS. The level of improvement between the *good* and *high* sets of networks is unlikely to be worth a 22 times increase in compute costs in the vast majority of use cases.

network	average error (px)	cost (mflops)
pupil fine	1.03	2.58
corneal ref a	0.66	0.93
corneal ref b	0.74	0.93
	accuracy	
classifier	94.2%	0.13

(a) ‘Very Low’ Computational Cost Set

network	average error (px)	cost (mflops)
pupil fine	0.98	6.26
corneal ref a	0.54	7.02
corneal ref b	0.56	7.02
	accuracy	
classifier	96.4%	4.85

(b) ‘Low’ Computational Cost Set

network	average error (px)	cost (mflops)
pupil fine	0.75	32.0
corneal ref a	0.52	11.6
corneal ref b	0.53	12.6
	accuracy	
classifier	97.9%	20.1

(c) ‘Good’ Computational Cost Set

network	average error (px)	cost (mflops)
pupil fine	0.67	620.5
corneal ref a	0.49	476.2
corneal ref b	0.49	476.2
	accuracy	
classifier	98.9%	125.1

(d) ‘Very High’ Computational Cost Set

Table C.2: Network Sets: Feature Extraction Performance and Cost

Front End	20cm	25cm	30cm	35cm	40cm	avg
Rule-Based	1.90°	1.15°	0.57°	0.90°	0.98°	1.10°
ML ‘Very Low’	1.67°	1.53°	1.41°	1.79°	3.35°	1.95°
ML ‘Low’	2.46°	1.59°	1.55°	1.71°	2.04°	1.87°
ML ‘Good’	0.81°	0.70°	0.71°	0.73°	0.65°	0.72°
ML ‘High’	0.82°	0.94°	0.73°	0.53°	0.52°	0.71°

(a) **Gaze Bias** (degrees)

Front End	20cm	25cm	30cm	35cm	40cm	avg
Rule-Based	2.27°	1.65°	1.00°	1.69°	2.67°	1.86°
ML ‘Very Low’	2.17°	1.89°	1.72°	2.03°	3.69°	2.30°
ML ‘Low’	2.95°	1.98°	1.75°	1.89°	2.55°	2.23°
ML ‘Good’	1.48°	1.36°	0.88°	1.07°	1.05°	1.17°
ML ‘High’	0.95°	1.06°	0.95°	0.69°	0.68°	0.87°

(b) **RMS** (degrees)

Table C.3: Supervised Dynamic Depth Eye Tracking Performance

Bibliography

[1] Helga Kolb. Gross anatomy of the eye. In *Webvision: The Organization of the Retina and Visual System*. University of Utah Health Sciences Center, 2007. [CrossRef].

[2] Dong Hyun Yoo et al. Non-contact eye gaze tracking system by mapping of corneal reflections. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 94–99. IEEE, 2002. [CrossRef].

[3] Elias Daniel Guestrin. *A novel head-free point-of-gaze estimation system.* PhD thesis, Departments of Edward S. Rogers Sr. Department of Electrical and Computer Engineering and Institute of Biomaterials and Biomedical Engineering, University of Toronto, 2003. [CrossRef].

[4] Elias Daniel Guestrin. *Remote, non-contact gaze estimation with minimal subject cooperation.* PhD thesis, 2010. [CrossRef].

[5] Dongheng Li et al. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 79–79. IEEE, 2005. [CrossRef].

[6] Rasoul Kheirolahy et al. Robust pupil boundary detection by optimized color mapping for iris recognition. In *Computer Conference, 2009. CSICC 2009. 14th International CSI*, pages 170–175. IEEE, 2009. [CrossRef].

[7] Guillaume Hervet et al. Is banner blindness genuine? eye tracking internet text advertising. *Applied Cognitive Psychology*, 25(5):708–716, 2011. [CrossRef].

[8] Marc Resnick et al. The impact of advertising location and user task on the emergence of banner ad blindness: an eye-tracking study. *International Journal of Human-Computer Interaction*, 30(3):206–219, 2014. [CrossRef].

[9] Keith Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372–422, 1998. [CrossRef].

[10] Geoffrey Duggan et al. Skim reading by satisficing: evidence from eye tracking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1141–1150. ACM, 2011. [CrossRef].

[11] Andrea Mazzei et al. 3d model-based gaze estimation in natural reading: a systematic error correction procedure based on annotated texts. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 87–90. ACM, 2014. [CrossRef].

[12] M Eizenman et al. A new methodology for the analysis of eye movements and visual scanning in drivers. *Transport Canada Contract Report*, 1999.

[13] Stijn De Beugher et al. Automatic analysis of eye-tracking data using object detection algorithms. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 677–680. ACM, 2012. [CrossRef].

[14] Mark Wilson et al. Eye movements drive steering: Reduced eye movement distribution impairs steering and driving performance. *Journal of motor behavior*, 40(3):190–202, 2008. [CrossRef].

[15] Paul A Wetzel et al. Instructor use of eye position based feedback for pilot training. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 42, pages 1388–1392. SAGE Publications Sage CA: Los Angeles, CA, 1998. [CrossRef].

[16] Nadine B Sarter et al. Pilots' monitoring strategies and performance on automated flight decks: An empirical study combining behavioral and eye-tracking data. *Human factors*, 49(3):347–357, 2007. [CrossRef].

[17] Nadir Weibel et al. Let's look at the cockpit: exploring mobile eye-tracking for observational research on the flight deck. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 107–114. ACM, 2012. [CrossRef].

[18] Jian-Gang Wang et al. Estimating the eye gaze from one eye. *Computer Vision and Image Understanding*, 98(1):83–103, 2005. [CrossRef].

[19] Lisa A Frey et al. Eye-gaze word processing. *IEEE Transactions on systems, Man, and Cybernetics*, 20(4):944–950, 1990. [CrossRef].

[20] Outi Tuisku et al. Now dasher! dash away!: longitudinal study of fast text entry by eye gaze. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 19–26. ACM, 2008. [CrossRef].

[21] Chi-Shin Hwang et al. An eye-tracking assistive device improves the quality of life for als patients and reduces the caregivers burden. *Journal of motor behavior*, 46(4):233–238, 2014. [CrossRef].

[22] Pamela Collins et al. Grand challenges in global mental health. *Nature*, 475(7354):27, 2011. [CrossRef].

[23] Moshe Eizenman et al. A naturalistic visual scanning approach to assess selective attention in major depressive disorder. *Psychiatry research*, 118(2):117–128, 2003. [CrossRef].

[24] Canan Karatekin et al. Exploratory eye movements to pictures in childhood-onset schizophrenia and attention-deficit/hyperactivity disorder (adhd). *Journal of Abnormal Child Psychology*, 27(1):35–49, 1999. [CrossRef].

[25] Monica E Calkins et al. Eye movement dysfunction in schizophrenia: a heritable characteristic for enhancing phenotype definition. *American Journal of Medical Genetics*, 97(1):72–76, 2000. [CrossRef].

[26] Ana García-Blanco et al. Attentional biases toward emotional images in the different episodes of bipolar disorder: an eye-tracking study. *Psychiatry Research*, 215(3):628–633, 2014. [CrossRef].

[27] Andrew D Peckham et al. Eye tracking of attention to emotion in bipolar i disorder: links to emotion regulation and anxiety comorbidity. *International journal of cognitive therapy*, 9(4):295–312, 2016. [CrossRef].

[28] Jonathan Chung. A novel test of implicit memory; an eye tracking study. *International Journal of Applied Mathematics, Electronics and Computers*, 2(4):45–48, 2014. [CrossRef].

[29] Deborah Riby et al. Looking at movies and cartoons: eye-tracking evidence from williams syndrome and autism. *Journal of Intellectual Disability Research*, 53(2):169–181, 2009. [CrossRef].

[30] Laurent Itti. New eye-tracking techniques may revolutionize mental health screening. *Neuron*, 88(3):442–444, 2015. [CrossRef].

[31] Terje Falck-Ytter et al. Eye tracking in early autism research. *Journal of neurodevelopmental disorders*, 5(1):28, 2013. [CrossRef].

[32] Tomer Shechner et al. Attention bias of anxious youth during extended exposure of emotional face pairs: An eye-tracking study. *Depression and anxiety*, 30(1):14–21, 2013. [CrossRef].

[33] Ad de Jongh et al. The impact of eye movements and tones on disturbing memories involving ptsd and other mental disorders. *Journal of Behavior Therapy and Experimental Psychiatry*, 44(4):477–483, 2013. [CrossRef].

[34] Chloé Prado et al. The eye movements of dyslexic children during reading and visual search: impact of the visual attention span. *Vision research*, 47(19):2521–2530, 2007. [CrossRef].

[35] Katrin E Giel et al. Attentional processing of food pictures in individuals with anorexia nervosaan eye-tracking study. *Biological psychiatry*, 69(7):661–667, 2011. [CrossRef].

[36] Leora Pinhas et al. Attentional biases to body shape images in adolescents with anorexia nervosa: An exploratory eye-tracking study. *Psychiatry research*, 220(1):519–526, 2014. [CrossRef].

[37] Lancet Global Mental Health Group. Scale up services for mental disorders: a call for action. *The Lancet*, 370(9594):1241–1252, 2007. [CrossRef].

[38] HU Wittchen et al. Towards a better understanding of the size and burden and cost of brain disorders in europe., 2005. [CrossRef].

[39] Masao Kakihara. Grasping a global view of smartphone diffusion: An analysis from a global smartphone study. In *ICMB*, page 11, 2014. [CrossRef].

[40] Elias Daniel Guestrin et al. General theory of remote gaze estimation using the pupil center and corneal reflections. *IEEE Transactions on biomedical engineering*, 53(6):1124–1133, 2006. [CrossRef].

[41] Corey Holland et al. Usability evaluation of eye tracking on an unmodified common tablet. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 295–300. ACM, 2013. [CrossRef].

[42] Erroll Wood et al. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 207–210. ACM, 2014. [CrossRef].

[43] Chiao-Wen Kao et al. An adaptive eye gaze tracker system in the integrated cloud computing and mobile device. In *2011 International Conference on Machine Learning and Cybernetics*, volume 1, pages 367–371. IEEE, 2011. [CrossRef].

[44] Corey Holland et al. Eye tracking on unmodified common tablets: challenges and solutions. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 277–280. ACM, 2012. [CrossRef].

[45] Shoya Ishimaru et al. Where are you looking at?-feature-based eye tracking on unmodified tablets. In *2013 2nd IAPR Asian Conference on Pattern Recognition*, pages 738–739. IEEE, 2013. [CrossRef].

[46] Kyle Krafka et al. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2176–2184, 2016. [CrossRef].

[47] Michael Xuelin Huang et al. Screenglint: Practical, in-situ gaze estimation on smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2546–2557. ACM, 2017. [CrossRef].

[48] Qiong Huang et al. Tabletgaze: dataset and analysis for unconstrained appearance-based gaze estimation in mobile tablets. *Machine Vision and Applications*, 28(5-6):445–461, 2017. [CrossRef].

[49] Google. Google product blog. https://blog.google.com, 2019. [CrossRef].

[50] MacRumors Staff. Apple's new "Pro" iPhones for those who want the best smartphone features. www.macrumors.com, 2019. [CrossRef].

[51] Apple. Framework: Arkit. http://www.apple.com/. [CrossRef].

[52] Huawei Private Communication, 2016. [CrossRef].

[53] David A Robinson. A method of measuring eye movemnent using a scieral search coil in a magnetic field. *IEEE Transactions on bio-medical electronics*, 10(4):137–145, 1963. [CrossRef].

[54] Robert S Allison et al. Combined head and eye tracking system for dynamic testing of the vestibular system. *IEEE Transactions on Biomedical Engineering*, 43(11):1073–1082, 1996. [CrossRef].

[55] Moritz Kassner, William Patera, and Andreas Bulling. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*, pages 1151–1160. ACM, 2014. [CrossRef].

[56] Dongheng Li et al. openeyes: a low-cost head-mounted eye-tracking solution. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 95–100. ACM, 2006. [CrossRef].

[57] Reuben M Aronson et al. Eye-hand behavior in human-robot shared manipulation. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 4–13. ACM, 2018. [CrossRef].

[58] Peter Kearney and Wen-Chin Li. Multiple remote tower for single european sky: The evolution from initial operational concept to regulatory approved implementation. *Transportation Research Part A: Policy and Practice*, 116:15–30, 2018. [CrossRef].

[59] Mohamed Khamis et al. Vrpursuits: interaction in virtual reality using smooth pursuit eye movements. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*, page 18. ACM, 2018. [CrossRef].

[60] Carlos H Morimoto and Marcio RM Mimica. Eye gaze tracking techniques for interactive applications. *Computer vision and image understanding*, 98(1):4–24, 2005. [CrossRef].

[61] G Daunys. Report on new approaches to eye tracking. *Cogain Reports*, 2006. [CrossRef].

[62] Dan Witzner Hansen et al. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2009. [CrossRef].

[63] Dan Witzner Hansen et al. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2009. [CrossRef].

[64] Jian-Gang Wang et al. Gaze determination via images of irises. *Image and Vision Computing*, 19(12):891–911, 2001. [CrossRef].

[65] Jochen Heinzmann et al. 3-d facial pose and gaze point estimation using a robust real-time tracking paradigm. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 142–147. IEEE, 1998. [CrossRef].

[66] Shumeet Baluja et al. Non-intrusive gaze tracking using artificial neural networks. Technical report, DTIC Document, 1994. [CrossRef].

[67] Andrew T Duchowski. Eye tracking methodology. *Theory and practice*, 328(614):2–3, 2007. [CrossRef].

[68] Jian-Gang Wang et al. Study on eye gaze estimation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(3):332–350, 2002. [CrossRef].

[69] Laura Sesma et al. 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*, pages 217–220. ACM, 2012. [CrossRef].

[70] Yoshio Matsumoto et al. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 499–504. IEEE, 2000. [CrossRef].

[71] Hirotake Yamazoe et al. Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 245–250. ACM, 2008. [CrossRef].

[72] Li Xia and other. Accurate gaze tracking from single camera using gabor corner detector. *Multimedia Tools and Applications*, 75(1):221–239, 2016. [CrossRef].

[73] Rhys Newman et al. Real-time stereo tracking for head pose and gaze estimation. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 122–128. IEEE, 2000. [CrossRef].

[74] Xuehan Xiong et al. Eye gaze tracking using an rgbd camera: a comparison with a rgb solution. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 1113–1121. ACM, 2014. [CrossRef].

[75] David Eigen et al. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. [CrossRef].

[76] Rostam Affendi Hamzah et al. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016, 2016. [CrossRef].

[77] Kar-Han Tan et al. Appearance-based eye gaze estimation. In *Sixth IEEE Workshop on Applications of Computer Vision, 2002.(WACV 2002). Proceedings.*, pages 191–195. IEEE, 2002. [CrossRef].

[78] Feng Lu et al. A head pose-free approach for appearance-based gaze estimation. In *BMVC*, pages 1–11, 2011. [CrossRef].

[79] Feng Lu et al. Learning gaze biases with head motion for head pose-free gaze estimation. *Image and Vision Computing*, 32(3):169–179, 2014. [CrossRef].

[80] Chao Gou et al. Learning-by-synthesis for accurate eye detection. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 3362–3367. IEEE, 2016. [CrossRef].

[81] Kang Wang et al. A hierarchical generative model for eye image synthesis and eye gaze estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 440–448, 2018. [CrossRef].

[82] Kang Wang et al. Deep eye fixation map learning for calibration-free eye gaze tracking. In *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications*, pages 47–55, 2016. [CrossRef].

[83] Kang Wang et al. Generalizing eye tracking with bayesian adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11907–11916, 2019. [CrossRef].

[84] Jixu Chen et al. 3d gaze estimation with a single camera without ir illumination. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008. [CrossRef].

[85] Kang Wang et al. Real time eye gaze tracking with 3d deformable eye-face model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1003–1011, 2017. [CrossRef].

[86] Alan Cowey. The basis of a method of perimetry with monkeys. *Quarterly Journal of Experimental Psychology*, 15(2):81–90, 1963. [CrossRef].

[87] EyeGaze LC Technologies, Inc. Eyegaze. http://www.eyegaze.com, 2015. [CrossRef].

[88] Tobii Technology. http://www.tobii.com/, 2015.

[89] ASL Inc. Applied Science Laboratories. http://www.asleyetracking.com/, 2015. [CrossRef].

[90] SMI Inc. SensoMotoric Instruments. http://www.smivision.com/, 2015. [CrossRef].

[91] John Merchant et al. Remote measurement of eye direction allowing subject motion over one cubic foot of space. *Biomedical Engineering, IEEE Transactions on*, (4):309–317, 1974. [CrossRef].

[92] K Preston White Jr et al. Spatially dynamic calibration of an eye-tracking system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(4):1162–1168, 1993. [CrossRef].

[93] Carlos Hitoshi Morimoto et al. Keeping an eye for hci. In *Computer Graphics and Image Processing, 1999. Proceedings. XII Brazilian Symposium on*, pages 171–176. IEEE, 1999. [CrossRef].

[94] Juan J Cerrolaza et al. Taxonomic study of polynomial regressions applied to the calibration of video-oculographic systems. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 259–266. ACM, 2008. [CrossRef].

[95] Akira Sugioka et al. Noncontact video-based eye-gaze detection method allowing large head displacements. In *Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 2, pages 526–528. IEEE, 1996. [CrossRef].

[96] Yoshinobu Ebisawa et al. Proposal of a zoom and focus control method using an ultrasonic distance-meter for video-based eye-gaze detection under free-head conditions. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 2, pages 523–525. IEEE, 1996. [CrossRef].

[97] JJ Kang et al. Simplifying the cross-ratios method of point-of-gaze estimation. In *30th Canadian medical and biological engineering conference (CMBEC30)*, 2007. [CrossRef].

[98] Elias D Guestrin et al. Analysis of subject-dependent point-of-gaze estimation bias in the cross-ratios method. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 237–244. ACM, 2008. [CrossRef].

[99] Flávio Luiz Coutinho et al. Augmenting the robustness of cross-ratio gaze tracking methods to head movement. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 59–66. ACM, 2012. [CrossRef].

[100] Flavio L Coutinho et al. Improving head movement tolerance of cross-ratio based eye trackers. *International journal of computer vision*, 101(3):459–481, 2013. [CrossRef].

[101] Jia Huang et al. Towards accurate and robust cross-ratio based gaze trackers through learning from simulation. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 75–82. ACM, 2014. [CrossRef].

[102] Takehiko Ohno et al. Freegaze: a gaze tracking system for everyday gaze interaction. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 125–132. ACM, 2002. [CrossRef].

[103] Sheng-Wen Shih et al. A calibration-free gaze tracking technique. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 201–204. IEEE, 2000. [CrossRef].

[104] Jie Yang et al. Real-time face and facial feature tracking and applications. In *AVSP'98 international conference on auditory-visual speech processing*, 1998. [CrossRef].

[105] Rainer Stiefelhagen et al. A model-based gaze tracking system. *International Journal on Artificial Intelligence Tools*, 6(02):193–209, 1997. [CrossRef].

[106] Rainer Stiefelhagen et al. Tracking eyes and monitoring eye gaze. In *Proc. Workshop on Perceptual User Interfaces*, pages 98–100, 1997. [CrossRef].

[107] Kyung-Nam Kim et al. Vision-based eye-gaze tracking for human computer interface. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, volume 2, pages 324–329. IEEE, 1999. [CrossRef].

[108] Ravi Kothari et al. Detection of eye locations in unconstrained visual images. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pages 519–522. IEEE, 1996. [CrossRef].

[109] Mark Nixon. Eye spacing measurement for facial recognition. In *Applications of Digital Image Processing VIII*, volume 575, pages 279–286. International Society for Optics and Photonics, 1985. [CrossRef].

[110] Antonio Perez et al. A precise eye-gaze detection and tracking system. 2003. [CrossRef].

[111] Roberto Valenti et al. Accurate eye center location and tracking using isophote curvature. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. [CrossRef].

[112] David Young et al. *Specialised hough transform and active contour methods for real-time eye tracking*. University of Sussex, Cognitive & Computing Science, 1995. [CrossRef].

[113] John Daugman. The importance of being random: statistical principles of iris recognition. *Pattern recognition*, 36(2):279–291, 2003. [CrossRef].

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

[115] John G Daugman. High confidence visual recognition of persons by a test of statistical independence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1148–1161, 1993. [CrossRef].

[116] Ehsan Mohammadi Arvacheh et al. Iris segmentation: Detecting pupil, limbus and eyelids. In *2006 International Conference on Image Processing*, pages 2453–2456. IEEE, 2006. [CrossRef].

[117] Martin A Fischler et al. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. [CrossRef].

[118] Wolfgang Fuhl et al. Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pages 123–130. ACM, 2016. [CrossRef].

[119] Thiago Santini et al. Pure: Robust pupil detection for real-time pervasive eye tracking. *Computer Vision and Image Understanding*, 170:40–50, 2018. [CrossRef].

[120] Thiago Santini et al. Purest: robust pupil tracking for real-time pervasive eye tracking. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, page 61. ACM, 2018. [CrossRef].

[121] Paul Bao, Lei Zhang, and Xiaolin Wu. Canny edge detection enhancement by scale multiplication. *IEEE transactions on pattern analysis and machine intelligence*, 27(9):1485–1490, 2005. [CrossRef].

[122] Alan L Yuille et al. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992. [CrossRef].

[123] Gareth J Edwards et al. Face recognition using active appearance models. In *European conference on computer vision*, pages 581–595. Springer, 1998. [CrossRef].

[124] Timothy F Cootes et al. Active shape modelssmart snakes. In *BMVC92*, pages 266–275. Springer, 1992. [CrossRef].

[125] Xangdong Xie et al. On improving eye feature extraction using deformable templates. *Pattern recognition*, 27(6):791–799, 1994. [CrossRef].

[126] Kin-Man Lam et al. Locating and extracting the eye in human face images. *Pattern recognition*, 29(5):771–779, 1996. [CrossRef].

[127] James P Ivins et al. A deformable model of the human iris for measuring small three-dimensional eye movements. *Machine Vision and Applications*, 11(1):42–51, 1998. [CrossRef].

[128] Carlo Colombo et al. Real-time head tracking from the deformation of eye contours using a piecewise affine camera. *Pattern Recognition Letters*, 20(7):721–730, 1999. [CrossRef].

[129] Kristen Grauman et al. Communication via eye blinks-detection and duration analysis in real time. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1. IEEE, 2001. [CrossRef].

[130] Peter W Hallinan. Recognizing human eyes. In *Geometric Methods in Computer Vision*, volume 1570, pages 214–227. International Society for Optics and Photonics, 1991. [CrossRef].

[131] Zhiwei Zhu et al. Real-time eye detection and tracking under various light conditions. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 139–144. ACM, 2002. [CrossRef].

[132] Ferdinando Samaria et al. Hmm-based architecture for face identification. *Image and vision computing*, 12(8):537–543, 1994. [CrossRef].

[133] PM Hillman et al. Global fitting of a facial model to facial features for model-based video coding. In *3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the*, volume 1, pages 359–364. IEEE, 2003. [CrossRef].

[134] Weimin Huang et al. Face detection and precise eyes location. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 722–727. IEEE, 2000. [CrossRef].

[135] Zhanpeng Zhang et al. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014. [CrossRef].

[136] Amin Jourabloo et al. Large-pose face alignment via cnn-based dense 3d model fitting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4188–4196, 2016. [CrossRef].

[137] Yue Wu et al. Facial landmark detection with tweaked convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):3067–3074, 2017. [CrossRef].

[138] Wolfgang Fuhl et al. Pupilnet: Convolutional neural networks for robust pupil detection. *Computer Vision and Pattern Recognition*, 2016. [CrossRef].

[139] Shaharam Eivazi et al. Improving real-time cnn-based pupil detection through domain-specific data augmentation. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, page 40. ACM, 2019. [CrossRef].

[140] FJ Vera-Olmos et al. Deepeye: Deep convolutional network for pupil detection in real environments. *Integrated Computer-Aided Engineering*, 26(1):85–95, 2019. [CrossRef].

[141] Ali Sharif Razavian et al. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. [CrossRef].

[142] Ravi Garg et al. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. [CrossRef].

[143] Jun-Cheng Chen et al. Unconstrained face verification using deep cnn features. In *2016 IEEE winter conference on applications of computer vision (WACV)*, pages 1–9. IEEE, 2016. [CrossRef].

[144] Huaizu Jiang et al. Face detection with the faster r-cnn. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 650–657. IEEE, 2017. [CrossRef].

[145] Y-Lan Boureau et al. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010. [CrossRef].

[146] Sergey Ioffe et al. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 2015. [CrossRef].

[147] Razvan Pascanu et al. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2, 2012. [CrossRef].

[148] Martin T Hagan et al. *Neural network design*, volume 20. Pws Pub. Boston, 1996. [CrossRef].

[149] Nitish Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. [CrossRef].

[150] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992. [CrossRef].

[151] Bing Xu et al. Empirical evaluation of rectified activations in convolutional network. *Computer Vision and Pattern Recognition*, 2015. [CrossRef].

[152] Caglar Gulcehre et al. Noisy activation functions. In *International conference on machine learning*, pages 3059–3068, 2016. [CrossRef].

[153] Barry L Kalman et al. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE, 1992. [CrossRef].

[154] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. [CrossRef].

[155] John Duchi et al. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. [CrossRef].

[156] Diederik P Kingma et al. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014. [CrossRef].

[157] Derrick Nguyen et al. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 21–26. IEEE, 1990. [CrossRef].

[158] Dmytro Mishkin and Jiri Matas. All you need is a good init. *The International Conference on Learning Representations*, 2015. [CrossRef].

[159] Kaiming He et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. [CrossRef].

[160] Christian Szegedy et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. [CrossRef].

[161] Roel Vertegaal et al. Designing attentive cell phone using wearable eyecontact sensors. In *Human factors in computing systems*, pages 646–647. ACM, 2002. [CrossRef].

[162] Takashi Nagamatsu et al. Mobigaze: Development of a gaze interface for handheld mobile devices. In *Human Factors in Computing Systems*, pages 3349–3354. ACM, 2010. [CrossRef].

[163] Oliver Hohlfeld et al. On the applicability of computer vision based gaze tracking in mobile scenarios. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 427–434. ACM, 2015. [CrossRef].

[164] Vytautas Vaitukaitis et al. Eye gesture recognition on portable devices. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 711–714. ACM, 2012. [CrossRef].

[165] Google. What is android, 2019. [CrossRef].

[166] The kernel development community. Video for Linux 2. https://www.kernel.org, 1999. [CrossRef].

[167] Google. Google mobile vision api. https://developers.google.com/, 2019. [CrossRef].

[168] Visage. Visage Technologies. http://visagetechnologies.com/, 2002. [CrossRef].

[169] Wageeh W Boles et al. A human identification technique using images of the iris and wavelet transform. *IEEE transactions on signal processing*, 46(4):1185–1188, 1998. [CrossRef].

[170] James S Maxwell et al. Adaptation of torsional eye alignment in relation to head roll. *Vision research*, 39(25):4192–4199, 1999. [CrossRef].

[171] Klaus Hepp. On listing's law. *Communications in Mathematical Physics*, 132(1):285–292, 1990. [CrossRef].

[172] Evangelos Alexandridis et al. *The pupil*. Springer, 1985. [CrossRef].

[173] Adrian Kaehler and Gary Bradski. *Learning OpenCV*. O'Reilly Media, Inc., 2014. [CrossRef].

[174] ER Davies. Machine vision: Theory, algorithms and practicalities, 1990. [CrossRef].

[175] Pietro Perona et al. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990. [CrossRef].

[176] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011. [CrossRef].

[177] John Schavemaker et al. Image sharpening by morphological filtering. *Pattern Recognition*, 33(6):997–1012, 2000. [CrossRef].

[178] Nick Kanopoulos et al. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988. [CrossRef].

[179] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. [CrossRef].

[180] Daniel D Polsby et al. The third criterion: Compactness as a procedural safeguard against partisan gerrymandering. *Yale L. & Policy Review*, 9:301, 1991. [CrossRef].

[181] Ian Goodfellow et al. *Deep learning*. MIT press, 2016. [CrossRef].

[182] Xiaomeng Li and other. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674, 2018. [CrossRef].

[183] Martín Abadi et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation)*, pages 265–283, 2016. [CrossRef].

[184] Andrew Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *Computer Vision and Pattern Recognition*, 2017. [CrossRef].

[185] Mark Sandler et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. [CrossRef].

[186] e-con Systems. See3CAM CU51 5MP Custom Lens Monochrome USB3 Camera. https://www.e-consystems.com, 2012. [CrossRef].

[187] Inc USB Implementers Forum. The usb 3.2 specification. www.usb.org, 2017. [CrossRef].

[188] Sreekrishnan Venkateswaran. *Essential Linux device drivers*. Prentice Hall Press, 2008. [CrossRef].

[189] Michael Backes et al. Artist: The android runtime instrumentation and security toolkit. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 481–495. IEEE, 2017. [CrossRef].

[190] Bill Venners. *Inside the Java Virtual Machine*. McGraw-Hill, New York, 1998. [CrossRef].

[191] Sylvain Ratabouil. *Android NDK: beginner's guide*. Packt Publishing Ltd, 2015. [CrossRef].

[192] Rob Gordon. *Essential JNI: Java Native Interface*. Prentice-Hall, Inc., 1998. [CrossRef].

[193] Andreas Pålsson. Art vs. ndk vs. gpu acceleration: A study of performance of image processing algorithms on android, 2017. [CrossRef].

[194] Terrence B Remple. Usb on-the-go interface for portable devices. In *IEEE International Conference on Consumer Electronics*, pages 8–9. IEEE, 2003. [CrossRef].

[195] Ken Tossel. A cross-platform library for usb video devices. https://github.com/, 2013. [CrossRef].

[196] Johannes Erdfelt. libusb developers guide. https://libusb.info, 2005. [CrossRef].

[197] Claire Gordon et al. Anthropometric survey of us army personnel: methods and summary statistics 1988. Technical report, DTIC Document, 1989. [CrossRef].