Gaze Tracking on Any Surface with Your Phone

Jiani Cao, Chengdong Lin
City University of Hong Kong
Hong Kong, China

Yang Liu
University of Cambridge
Cambridge, United Kingdom

Zhenjiang Li
City University of Hong Kong
Hong Kong, China

ABSTRACT

This paper introduces ASGaze, a new gaze tracking system designed using the common RGB camera from mobile phones. In addition to improving the accuracy of existing RGB camera-based gaze tracking methods, a novelty of ASGaze is that it can be configured to track gaze points on various surface areas commonly required in different applications, such as mobile phone screens, computer displays or even non-electronic surfaces like whiteboards or paper - a situation that is difficult for existing RGB camera-based methods to handle. To achieve the design of ASGaze, we revisit the 3D geometric model of the eye, which is widely adopted by high-end and commercial gaze trackers, and it has the potential to achieve our design goals. To avoid the high cost of commercial solutions, we identify three key issues to be addressed when processing the eye model with an RGB camera, including how to first accurately extract eye iris boundary that is the meta-information in our gaze tracking design, and then how to remove gaze ambiguity from iris boundary to gaze point transformation, and finally how to precisely map gaze points to the target tracking surface. In this paper, we propose a series of effective techniques to address these issues. We develop a prototype system and conduct extensive experiments on three different typical tracking surfaces to show promising performance gains compared to the recent solution.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; *Interaction techniques.*

KEYWORDS

Gaze tracking, mobile sensing, eye model

ACM Reference Format:
Jiani Cao, Chengdong Lin, Yang Liu, and Zhenjiang Li. 2022. Gaze Tracking on Any Surface with Your Phone. In *The 20th ACM Conference on Embedded Networked Sensor Systems (SenSys '22), November 6–9, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3560905.3568544

1 INTRODUCTION

Gaze tracking is a technique [18] that uses camera(s) as a sensor to infer where a user is looking, known as the *gaze point*, by capturing video frames or images of the user's eyes. Gaze tracking can enable a wide range of useful applications, such as gaze-based input designs that can facilitate disabled people [51] or protect user typing privacy on mobile devices [32], rehabilitation aids for neurologically impaired or dyslexic patients [22], and analysis for the content of interest to different users [28, 42]. In addition to accuracy, gaze tracking may be required on various surfaces in different applications, such as mobile phone screens, computer displays, or even non-electronic screens like whiteboards or paper. For example [23, 38], zigzag lines or scrambled numbers are often printed on paper in rehabilitation for the eyes of dyslexic patients to follow. However, without gaze tracking capabilities on non-electronic surfaces, doctors cannot assess the effectiveness of rehabilitation.

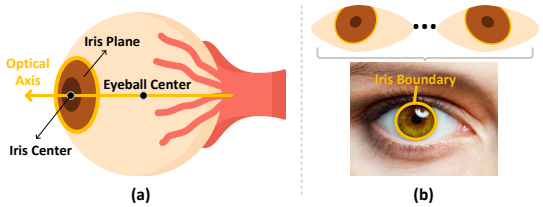

Figure 1: (a) Illustration of the 3D geometry of the eye, and (b) the iris boundary, which looks like an ellipse when the eye looks in different directions.

The classic principle of gaze tracking relies on a 3D geometric model of the eye [52], which can fulfill above design requirements. As shown in Figure 1(a), the optical axis of the eye is a line perpendicular to the iris plane that intersects the iris center and eyeball center. Once the optical axis is known, the actual visual axis is offset by a small kappa angle due to the structure of eye [7]. When we extend the visual axis, the intersection with the tracking surface gives a gaze point. However, the main challenge in gaze tracking design is that the direction of the optical/visual axis is highly subtle information, but the pose of the user head and the distance of the eyes to the camera can change during tracking. If the above geometric relation cannot be obtained with high accuracy, the tracking performance will drop significantly. Thus, commercial solutions like Tobii [5] utilize dedicated hardware (such as infrared light sources and receivers) to reliably reconstruct eye geometry. They can analyze and decode the specialized reflection patterns of infrared light on the cornea and pupil to achieve accurate gazing tracking. However, these commercial products are generally expensive.

To develop a low-cost solution, many recent designs [9, 13, 17, 21, 27, 39, 40, 62] have taken common RGB cameras from mobile devices or webcams and proposed an alternative, called *appearance-based* solutions, to bypass obtaining eye geometry. The main idea is to label a large dataset with the ground truth of the gaze points for the eye images captured in as many situations as possible, including different head poses, eye-to-device distances, *etc.*, and then leverage neural networks to mine sophisticated relationships from input eye images to output gaze points under these factors. Although this is an end-to-end approach without requiring specialized sensors and hardware, the overall search space for these uncertainties is

SenSys '22, November 6–9, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9886-2/22/11…$15.00
https://doi.org/10.1145/3560905.3568544

numerous. Thus, this requires a large dataset to cover the diversity brought by these factors, resulting in significant data collection overhead. Furthermore, existing appearance-based methods that estimate gaze points on tracking surfaces mainly work on electronic screens, as they need to display the ground truth of gaze points to collect training data.

To overcome these shortcomings, we revisit the geometric model of the eye used by commercial solutions and identify opportunities to enable a geometry-based tracking design by using common RGB cameras without any specialized sensor or hardware. It is feasible because the optical axis is perpendicular to the iris plane, and this direction can also be inferred from the shape of the iris boundary. As shown in Figure 1(b), the iris boundary is a circle, but it looks like an *ellipse* when the eye looks in different directions. In fact, existing works have derived the mathematical relationship between the elliptical parameters of an iris boundary and the direction of the optical axis, but it has not been widely used in gaze tracking design due to the difficulty to reliably extract the iris boundary before. However, recent breakthroughs in computer vision have shown that accurate iris boundaries can be extracted from RGB cameras through novel neural network designs and image-processing techniques [33, 54]. Therefore, we aim to leverage such recent breakthroughs to develop a ubiquitous gaze tracker on RGB cameras. However, existing iris tracking technologies cannot be directly applied to our system due to the following challenges:

1) Iris boundary is thin and occupies very few pixels on each eye frame [33]. Due to the influences of eyelids and eyelashes, existing iris tracking designs tend to miss some key boundary pixels or add non-boundary pixels by mistake. These errors may not be noticed from metrics evaluating the quality of extracted iris boundaries, but they can seriously affect the subsequent gaze tracking.

2) Even if high-quality iris boundaries can be obtained, each elliptical iris boundary is a 2D shape that needs to be mapped into 3D space for determining gaze direction. However, each 2D elliptical shape has two degrees of freedom to be embedded in 3D space, only one of which results in the correct gaze direction. Therefore, we need to exclude such ambiguity effectively.

3) We use the phone to capture the user's eye frames, but our gaze tracking is not limited to the phone screen. As stated before, gaze tracking may be required on other external areas, such as a computer screen nearby and even a whiteboard or paper (not an electronic screen — a situation that is difficult for existing appearance-based designs to handle). Therefore, we need to accurately map the gaze points tracked by the phone to the tracking surface.

We address above challenges by designing ASGaze. In this system, we introduce a novel processing pipeline that can be integrated into recent iris tracking method [33] to improve the quality of the extracted iris boundaries. It exploits the geometric features of boundary pixels and the temporal relationship between neighboring eye frames to select high-quality iris boundary pixels. To further remove the gaze direction ambiguity, we leverage a mathematical property [19] that when a circle is in translational motion, its normal vector direction keeps unchanged relative to a fixed point. Since the displacement of iris plane over two frames can be approximated as translational motion (§4), we can choose the optical axis that accumulates the least change as the result. Finally, we do not need to collect ground truth of gaze points to train ASGaze. We only need to label iris boundaries for a subset of eye frames for iris boundary detection, based on which gaze points can be derived directly. Hence, ASGaze can work on an external surface area after gaze points tracked by phone are mapped to this area. To this end, the user only needs to look at the four corners of the area during the setup phase when the phone screen is nearly parallel to the tracking area. ASGaze can determine the minimum number of eye frames to use when the user looks at each corner, reducing setup latency and ensuring the quality of gaze point mapping.

We implement a prototype system of ASGaze by using an iPhone 11 Pro and evaluate gaze tracking performance on three typical surfaces of a computer screen, a whiteboard, and a phone screen. We compare ASGaze with the recent appearance-based method EVE [40] on eight recruited volunteers and a public dataset released by the EVE's team. Overall, the tracking error of EVE is about 3.75–4.26 cm for these volunteers and 3.20 cm for the public dataset. ASGaze can reduce the error to only 1.69–2.40 cm. We also examine ASGaze under different settings, including different head poses, eye-to-camera distances, lighting conditions, *etc.*, and find that ASGaze can achieve good tracking performance reliably under various settings. The project site is at https://asgaze.github.io. In summary, this paper makes the following contributions:

- To the best of our knowledge, ASGaze is the first gaze tracking design based on geometric eye model and common RGB camera, whose tracking can be extended to various external tracking surfaces, not limited to phone screens.
- We propose efficient and practical techniques to address three unique challenges in ASGaze, including errors of iris boundary pixels, ambiguity of the gaze direction, and mapping of gaze points to the tracking surface area.
- We develop an ASGaze prototype and conduct extensive evaluations on different tracking surfaces. The results show promising performance gains compared to the recent method on different datasets.

2 PRELIMINARY AND BACKGROUND

2.1 Application Scenarios

Novel HCI systems. Gaze tracking can be used for novel and useful human-computer interaction (HCI) designs [49, 58]. For example, it allows people with disabilities to use the gaze as a mouse to interact with and control a computer [51]. In addition, gaze tracking can also enable eye-based input design [20, 63]. It can protect the user's typing privacy (such as passwords and sensitive personal information) on mobile devices, where the finger does not need to physically touch each button, and this can prevent nearby people from snooping on the information the user typed [32].

Smart health. For patients of vision impairment or dyslexic, they need long-term rehabilitation for their eye movement abilities [22, 46]. In rehabilitation materials, zigzag lines or scrambled numbers are printed on each page of the rehabilitation materials [23]. Patients need to force their eyes to follow the zigzag line to move or search numbers in increasing order to practice [38]. However, there are

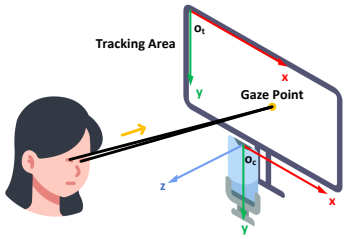

Figure 2: Illustration of the gaze tracking setup.

currently no gaze trackers working on non-electronic areas, and doctors cannot fully understand the effectiveness of rehabilitation.

Analysis for content of interest. When a user is browsing the web or using an APP, if the user enables the gaze tracking function, the gaze points can tell valuable meta information, such as what the users are interested in and how long the user's eyes stay in each content [42]. These statistics are useful to provide personalized recommendations or layouts for users, and are also valuable for developers to optimize advertisement placement on web pages, APP layout design, article topic selection, and so on.

2.2 System Setting and Architecture

Setup. Mobile phone is placed next to the tracking surface, such as under a computer screen or under a whiteboard. ASGaze will work as long as the phone screen is nearly parallel to the tracking area (§5). During the setup process, there are two important steps:

- **Step-1**: a user looks freely at any positions in the tracking area, and the phone records video frames of the eyes. One advantage of iris geometry based tracking designs is that we do not need to collect ground truth of gaze points. We only need to label the iris boundaries for a subset of eye frames, which are used to train a neural network to extract iris boundaries, from which gaze points can be directly derived.
- **Step-2**: the user stares at the four corners of the tracking area in turn, which will be used to map gaze points from the camera coordinate system to the tracking area. As shown in Figure 2, after ASGaze determines the location of the iris center and the direction that the user looks at, the intersection of this gaze direction and the screen gives a gaze point. ASGaze can get the gaze points for each eye, and we use their average as the result in ASGaze.

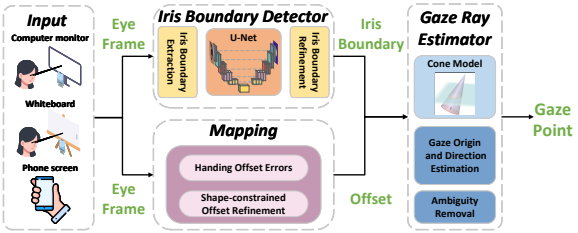

Figure 3: Architecture of the ASGaze design.

Architecture. Figure 3 illustrates the architecture of ASGaze with three major modules. Iris boundary detector contains our proposed processing pipeline integrated into a state-of-the-art iris tracking method. The detector is trained by using the labeled iris boundaries. After training, it reads video frames of the user's eyes as input and outputs high-quality iris boundaries extracted from each eye frame. The extracted iris boundaries are then passed to the gaze ray estimator to output gaze points. During the setup stage, the ambiguity removal component in gaze ray estimator learns a rule on how to determine the correct gaze direction based on the video frames collected from users, and the estimator can instantiate all the parameters to convert iris boundaries to gaze points. On the other hand, during system setup of the mapping module, when the user looks at each corner of the tracking area, this module can tell when the collection of video frames can be stopped. After the collection for all the corners is completed, it derives the relationship that maps gaze points in the camera coordinate system to the tracking area, which is a necessary step to ensure gaze tracking on any surface. In the following, we elaborate the design of each module.

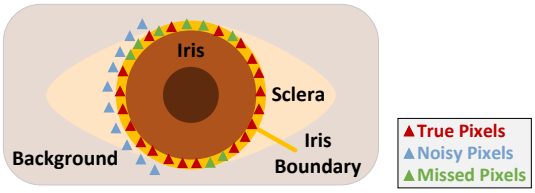

Figure 4: Illustration of iris boundary detection.

3 IRIS BOUNDARY DETECTOR

In this section, we introduce the design of *iris boundary detector* in Figure 3, which reads video frames of the user's eyes as input and outputs high-quality iris boundaries extracted from each frame.

3.1 Design Principle

The iris boundary is a thin ellipse. Missing some key boundary pixels or adding a few erroneous non-boundary pixels can distort the estimation of the ellipse parameters, which will in turn lead to large gaze tracking errors. Recent research uses neural networks to advance iris tracking, and a recent work, denoted as IrisTrack [33], achieves the state-of-the-art performance of iris boundary detection, but it cannot be applied to ASGaze directly. In the following, we first introduce the design rationale of IrisTrack and elaborate why it cannot be directly used. Then we introduce effective designs on top of IrisTrack to obtain accurate iris boundaries in §3.2.

Rationale of IrisTrack. The previous iris detection methods [54] mainly divide pixels from each eye frame into two classes: iris boundary pixels and other pixels, and adopt a neural network to recognize iris boundary pixels in an end-to-end manner. Because iris boundary is thin, it is difficult for a network to learn sufficient features from the boundary, which limits the accuracy of the detection result. To overcome this problem, IrisTrack [33] proposes to convert the task of iris boundary detection to a *segmentation* task, which divides all the pixels of each frame into three classes and classifies each pixel to class k, where $k \in \{iris, sclera, background\}$ as shown in Figure 4. IrisTrack finds that the iris boundary can be sketched using a *thin* stripe of the pixels that have high classification *uncertainties* belonging to the classes of iris and sclera[1], which can achieve the state-of-the-art iris detection performance.

[1]The network classifies every pixel among three classes $\{iris, sclera, background\}$. For each pixel, the network outputs the probability that it belongs to each of these three classes. A lower probability means a higher uncertainty to fall into this class.

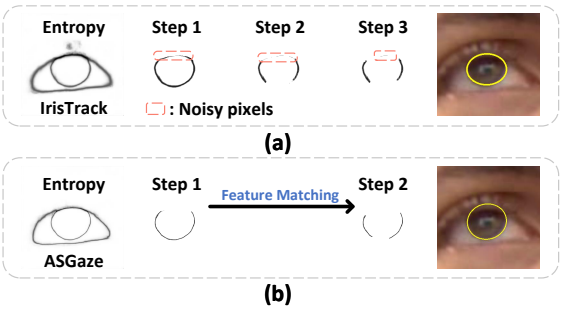

Figure 5: Intermediate results of iris boundary detection by (a) IrisTrack and (b) our proposed design.

To enable this design, IrisTrack employs a U-Net backbone [43] as the segmentation network, which takes T continuous video frames of user's eyes as input. For each frame x has I pixels, denoted as $x =< x(0), x(1), \ldots, x(I-1) >$. The neural network outputs a probability map p for frame x, which can be written as $p =< p_k(0), p_k(1), \ldots, p_k(I-1) >$, where $p_k(i)$ is the probability of pixel $x(i)$ belonging to class k. IrisTrack utilizes *cross-entropy* to select the boundary pixels — Because iris boundary is at the intersection between iris and sclera, the boundary pixels should have higher uncertainty belonging to these two classes than other pixels, which leads to high cross-entropy values as follows:

$$-\sum_k p_k(i) \times \log(p_k(i)). \tag{1}$$

To illustrate this design, Figure 5(a) shows an example. The first result shows the entropy values from an eye frame, in which the darker the color, the higher the uncertainty value. From the result, we can see that the entropy values already sketch the iris boundary (between iris and scelera) and the eyelid boundary (between background and iris/scelera). Then, the entropy values of the eyelid boundary can be set to zero to preserve the iris boundary only for the following operations, as shown by "Step 1" in Figure 5(a). To further improve the obtained boundary, IrisTrack proposes two mechanisms to remove noisy pixels ("Step 1" and "Step 2" in Figure 5(a)). The remaining pixels (in "Step 3") are used to fit an ellipse as the iris boundary, as depicted on the eye frame in Figure 5(a).

Observations. Although IrisTrack achieves the state-of-the-art iris detection result [33], our evaluation in §6 reveals that it may not lead to an accurate gaze tracking directly, where the tracking error of using IrisTrack can be greater than 3 cm. Through our study, we observe the following two limitations:

- Even though IrisTrack obtains sharp boundaries already, they are still not thin enough to infer gaze information and the boundary thickness should be further reduced because the thickness is the uncertain range when we fit an ellipse. The major reason that limits the boundary thickness is the loss function used to guard the feature extraction from each class, where only a cross-entropy loss is used by IrisTrack. However, this loss cannot reliably extract features for the pixels close to the iris boundary.
- IrisTrack utilizes the canny-edge detection method [10] to refine the obtained iris boundary pixels, which needs to calculate the gradient of each pixel in the frame and then remove all the pixels whose gradients are greater than an empirical threshold. However, it is common for the threshold becoming sub-optimal for an input eye frame. In this case, noisy pixels are not effectively removed, as shown in "Step 3" of Figure 5(a), which can distort the shape of the fitted ellipse and affect the subsequent gaze tracking performance.

Our solution. Since the methodology of IrisTrack to obtain iris boundary is effective, we still follow its two steps in ASGaze: 1) identify candidate pixels and 2) use entropy to sketch iris boundary. However, for the first step, we propose to design a new and novel mechanism to purify the candidate boundary pixels (§3.2). To this end, we design a set of efficient loss functions dedicated to gaze tracking and further leverage the geometric constraint of the iris boundary to guard the neural network to detect the iris-boundary pixels effectively. For the second step, we propose to further refine the obtained iris boundary by considering the temporal constraint across consecutive frames (§3.3).

3.2 Iris Boundary Extraction of ASGaze

In ASGaze, we still use U-Net as the backbone of the segmentation network. However, how to ensure efficient feature extraction needs to be investigated carefully. Using only a cross-entropy (CE) loss

$$L_{CE} = -\sum_k \sum_i I_k(i) \times \log(p_k(i)), \tag{2}$$

where $I_k(i)$ is a binary value, which equals to 1 if the label of pixel $x(i)$ is class k and 0 otherwise, the network tends to treat each pixel equally. This can cause an issue that the network extracts features more effectively from the pixels that are easier to be classified (*i.e.*, those are far from boundaries) and less effectively from the pixels close to boundaries, a typical class-imbalanced problem [11, 25, 34, 50]. To address this issue, we systematically improve feature extraction in ASGaze by the following design.

Mitigate missing iris boundary pixels. The first step is to reduce the chance of missing iris boundary pixels. To this end, the network should pay more attention to the pixels near iris boundary and extract features more effectively from these pixels.

Therefore, inspired by [11, 34], we use the network output $p_k(i)$, the probability of pixel $x(i)$ belonging to class k, to weight the entropy calculation during the training. In particular, a smaller probability $p_k(i)$ indicates a larger uncertainty, and we can differentiate pixels to make such low-uncertainty pixels (with a larger value of $(1-p_k(i))$, *e.g.*, near boundaries usually) contribute more in the loss function. By doing so, training receives more gains when loss function is minimized, which enforces the network to be more effective on these pixels. Hence, the first missing loss (ML) is:

$$L_{ML} = -\sum_k \sum_i (1-p_k(i))^\gamma \times I_k(i) \log(p_k(i)), \tag{3}$$

where γ is a parameter. With L_{ML}, the network can focus more on the pixels near boundaries. However, for gaze tracking, we only need boundary pixels for iris, so we further add a constraint to make the network focusing on the iris boundary. To achieve this, we can add a relatively large factor w to the loss value for the iris boundary pixels — iris boundary pixels[2] lead to higher gains to

[2]When we label the eye frames in the training dataset, each pixel is annotated with a label of iris, sclera and background. Then, we employ the boundary extraction algorithm [10] to further obtain the ground truth of the boundary on the eye frame. Therefore, each pixel has an additional label: it is on boundary or not.

minimize the loss in the training. Therefore, L_{ML} can be further improved as the loss L_{IML} for iris boundary:

$$L_{IML} = \begin{cases} w \cdot L_{ML}, & x(i) \in \{iris\_boundary\}, \\ L_{ML}, & x(i) \notin \{iris\_boundary\}, \end{cases} \quad (4)$$

where w is set to 20 empirically in our current design. With L_{IML}, the network becomes less likely to miss iris-boundary pixels compared to the IrisTrack, as shown by "Step 1" in Figure 5(b).

Remove noisy pixels. In addition to miss certain iris boundary pixels, it is also possible that some non-boundary pixels may be selected by the network by mistake. To address this issue, we introduce the following distance map loss L_{DML} by using the distance between each pixel and its nearest iris-boundary pixel

$$L_{DML} = \sum_k \sum_i D(i) \times p_k(i), \quad (5)$$

where $D(i)$ is the distance from pixel $x(i)$ to the nearest pixel on the iris boundary, calculated by $D(i) = \min\{Dis(x(i), x(j))\}$, where $x(j) \in \{iris\_boundary\}$. The rationale of L_{DML} is that when distance $D(i)$ is used as a loss, the non-boundary pixels become less likely to remain as candidate boundary pixels, because they can introduce a non-trivial value to the overall loss [25] but the final loss will be minimized during the training.

Generalized dice loss. In the early stage of training, the segmentation network is not well trained yet, which cannot recognize each pixel as one of three classes $\{iris, sclera, background\}$ reliably. In this case, the gain of removing noisy pixels (through L_{DML}) proposed above is limited. Therefore we should prioritize the original classification task first and then gradually focus on the boundary pixels. To this end, we propose to leverage the *intersection ratio* [50], for the classification result and the labeled ground truth, as an indicator to infer the classification quality. Therefore, we introduce a generalized dice loss L_{GDL} as follows:

$$L_{GDL} = 1 - 2\frac{\sum_k \beta_k \sum_i (I_k(i) \times p_k(i))}{\sum_k \beta_k \sum_i (I_k(i) + p_k(i))}, \quad (6)$$

where $I_k(i)$ is a binary value, which equals to 1 if the label of pixel $x(i)$ is class k and 0 otherwise, and β_k is the percentage of pixels in each class, *i.e.*, $\beta_k = \frac{1}{\sum_i (I_{k(i)})^2}$. If the classification result is perfect, the ratio in Eq. (6) is one, *i.e.*, a larger ratio indicates a better result. Therefore, we add a negative sign in Eq. (6), so that the ratio can be maximized when the overall loss is minimized in the training.

Overall loss. In summary, the loss function of the iris boundary detector in ASGaze is defined as follows:

$$L = L_{IML} + (1 - \alpha) \times L_{DML} + \alpha \times L_{GDL}, \quad (7)$$

where α is tuning factor to adjust the contributions of L_{DML} and L_{GDL}. In our current implementation, we set α to 1 initially. As training goes by, α is gradually decreased until reaching 0.5.

After training, the segmentation network outputs the probability for every pixel from the input eye frame falling into each class. Similar to IrisTrack, we can still select the pixels with high uncertainties between *iris* and *sclera* as the candidates of iris boundary pixels.

3.3 Iris Boundary Refinement

Figure 5(b) shows the candidates of iris boundary pixels ("Step 1") by our design proposed above. We can see that they form a thin boundary compared to IrisTrack in Figure 5(a). In this subsection, we find that we can further improve it by leveraging the temporal constraint between consecutive eye frames.

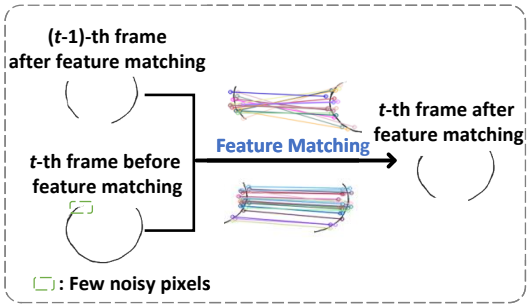

Figure 6: Illustration of the boundary refinement by using the temporal constraints between consecutive eye frames.

From the result of "Step 1" in Figure 5(b), we notice that most noisy pixels have been removed except for very few ones, as illustrated and zoomed in by the green rectangle of Figure 6. We further exclude such noises by considering the temporal relations among consecutive eye frames $\{x^{t-1}, x^t\}$. To this end, we adopt the SURF operator [8] to perform feature extraction for the iris boundary pixel candidates obtained from both eye frames. It can extract the key pixels that best represent the iris boundary and ignore the less important pixels. After that, *feature matching* [24] is performed between the boundary pixels from these two frames. Since the number of noisy pixels is small and it is less likely to have the same noisy pixels in each frame, this operation essentially leverages the temporal relation to treat the common pixels from both eye frames as the actual iris boundary pixels, and the unmatched pixels are removed as noise in this refinement. As shown in Figure 6, the (t-1)-th frame has been refined with the previous frame, which is used as a reference to refine the t-th frame. Because the first frame lacks a reference, feature matching is performed for the first two frames together. According to our experiments in evaluation (§6), this refinement scheme can decrease the gaze tracking error by 9.8%–32.4% under various settings.

Finally, we use the refined iris boundary pixels to fit an ellipse [14] to obtain its parameters $\{c_x, c_y, sa, sb, \varphi\}$, where c_x, c_y are the center coordinates, sa, sb are semi-major axis length and semi-minor axis length respectively, and φ is the angle from the positive horizontal axis to the ellipse's major axis. The expression of an ellipse can be obtained from above parameters as follows:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0, \quad (8)$$

where A, B, C, D, E, F can be calculated from $\{c_x, c_y, sa, sb, \varphi\}$.

This parameterized ellipse is viewed as the iris boundary and is used in the next system module to derive gaze points (§4). Figure 5(b) illustrates the final result ("Step 2") of the iris boundary detector for the input eye frame example. We can see that it contains much less noisy pixels compared to IrisTrack, which can reduce gaze tracking error significantly (§6).

4 GAZE RAY ESTIMATOR

This module further determines the user's gaze information according to the iris boundary obtained from iris boundary detector.

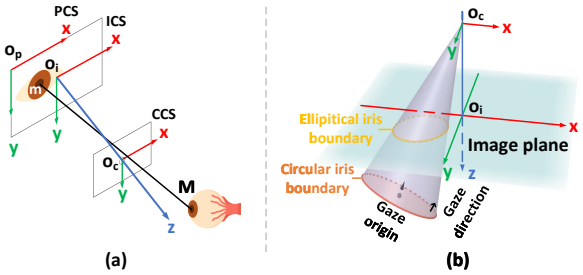

Figure 7: (a) Illustration of different coordinate systems. (b) Illustration of using a cone to estimate the gaze ray.

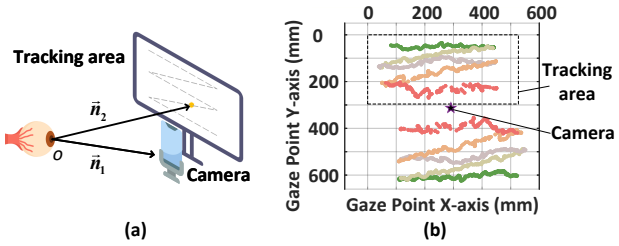

Figure 8: (a) Setup to examine the ambiguity issue of gaze direction. (b) Gaze points from both the correct and false gaze directions.

4.1 Problem Statement

The user's gaze information can be represented as a (normal) vector, denoted as *gaze ray*, containing the following two parts:

- **Gaze origin**: the center of the user's iris plane that is also the origin of the gaze ray.
- **Gaze direction**: the direction of the gaze ray.

After we know the user's gaze ray, the intersection between the gaze direction and the tracking surface gives the **gaze point** for the current gaze ray. Therefore, in this section, we first introduce how to estimate gaze ray according to the iris boundary information obtained from iris boundary detector. In the next section, we focus on how to determine the location of an external tracking surface.

Coordinate systems. There are three coordinate systems related to the gaze ray estimation, as illustrated in Figure 7(a):

- *Camera coordinate system (CCS)*: The focal of the camera is usually the origin o_c of this coordinate system, and its x-y plane is perpendicular to the focal direction. The unit of distance in CCS is millimeter (mm).
- *Image coordinate system (ICS)*: For each eye frame or image, its center is the origin o_i of this coordinate system, and each of its three axes is parallel to that of CCS. The unit of distance in ICS is also millimeter (mm).
- *Pixel coordinate system (PCS)*: This coordinate system shifts its origin o_p to the top-left pixel of the eye frame. Different from CCS and ICS, the unit of distance in PCS is pixel (px).

The transformation from any point in CCS to its projected point on the x-y plane of ICS/PCS has been well studied by the camera imaging principle [16], such as the iris center M is projected to m on the eye frame in Figure 7(a), which is used in the implementation.

Problem. The gaze ray should be determined in the 3D camera coordinate system, CCS. As stated above, gaze ray is a vector and we thus need to know: 1) the position of its origin o, and 2) the orientation or direction of this vector $\vec{n}$. Because ASGaze utilizes the iris boundary extracted from each eye frame to determine the corresponding gaze ray, the gaze ray estimator needs to know — how to estimate gaze ray $[o, \vec{n}]$ in the 3D CCS from the parameterized ellipse of an iris boundary on the 2D plane of PCS?

4.2 Determine Gaze Ray

To estimate gaze ray, an effective approach from previous studies, such as [44], is to introduce a cone C in CCS, where the vertex of the cone is placed at the origin of CCS (o_c). The base of the cone is a circle that represents the user's iris, and the normal vector of the base plane is the gaze direction. As the user's eye looks at one direction, it is equivalent to rotating the cone relative to its vertex so that the central axis of the cone is in that direction. As illustrated by Figure 7(b), the cone can intersect with an image plane and the cross-section is an ellipse, representing the iris boundary extracted from the eye frame. Therefore, we can search for the orientation of the cone, so that the intersected ellipse best matches the iris boundary obtained from iris boundary estimator.

Following this design principle, after searching, the center of the base and the orientation of the cone represent the gaze origin and gaze direction, respectively. To this end, we define the cone C in CCS as follows:

$$ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2ux + 2vy + 2wz + d = 0,$$

where parameters $a, b, c, d, f, g, h, u, v, w, d$ can be calculated from the ellipse parameters θ in ICS. (The ellipse parameters θ are originally defined in the pixel coordinate system PCS, which can be easily transformed to ICS.) This definition does not tell the height of the cone. We can use the iris diameter as a constraint to determine the height.[3] With this constraint, after the cone orientation is known, the center of cone's base is the gaze origin o.

Ambiguity issue of gaze direction. To determine the cone's orientation, the cross-section of the cone (intersected with the image plane) can be expressed as:

$$lx + my + nz = 0,$$

where (l, m, n) is the unit normal vector of the cross-section, *i.e.*, $l^2 + m^2 + n^2 = 1$. Then, we can use the 3D location estimation method of circular features [44] to find the cross-intersection on the image plane that best matches the iris boundary. However, the solution to this matching problem is not unique. We find that we can always obtain two different gaze directions ($\vec{n}_1$ and $\vec{n}_2$) with the same gaze origin o, but only one of them is the correct direction.

Figure 8(a) shows an example, where the user's eyes follow a moving point on the computer monitor and the mobile phone is placed under the monitor to capture eye frames. For each frame, we calculate the gaze origin o and two possible gaze directions ($\vec{n}_1$ and $\vec{n}_2$). By using the mapping design of gaze points proposed in the next section, we plot the gaze point trajectories for both gaze

[3]The height can be determined by setting the diameter of the cone's base similar to the iris diameter, and there are two feasible ways to obtain the iris diameter. Because the difference of the iris size is not significant among different people, its diameter is about 6 mm usually [12]. In our current design, we adopt this empirical value and find that it works well among different users in evaluation (§6). If ASGaze is a customized tracker for one particular user, we can measure the iris diameter of this user and use this precise value in the design. We will examine such an opportunity in the future.

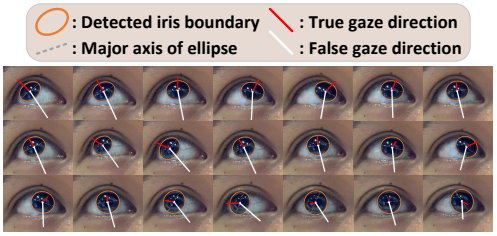

Figure 9: Illustration of the correct and false gaze directions.

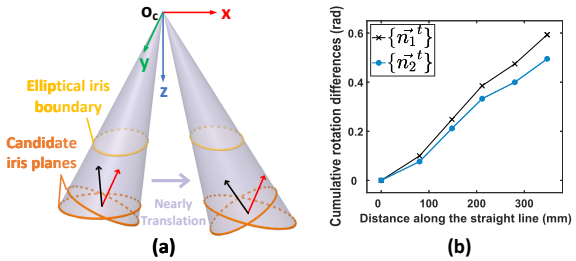

Figure 10: (a) Differences of the iris planes obtained from different gaze directions. (b) Accumulated rotation differences of the correct and false gaze directions.

directions in Figure 8(b). We can see that one trajectory is within the tracking area all the time, while the other one is always out of the tracking area, which should be identified and avoided.

The reason of this phenomenon is because when a 3D cone is projected to a 2D image plane, there are always two symmetric directions that can result in the same 2D ellipse (*i.e.*, the iris boundary). In the context of gaze tracking, only one of them represents the correct gaze direction. Figure 9 shows the examples when the user looks at different directions. These two directions $\vec{n}_1$ and $\vec{n}_2$ are always symmetric with respect to the major axis of the ellipse.

To remove the ambiguity of gaze direction, existing works have proposed "one-circle" [53] and "two-circle" [55] methods. The "one-circle" method observes that the center of the eyeball has nearly equal distance to the two corners of the eye. However, the eyeball center (not the iris center) is difficult to obtain using RGB cameras. The "two-circle" method assumes that the gaze directions of two eyes are nearly parallel. Therefore, the gaze points obtained from correct gaze directions should have the smallest angle compared to other angles formed by using other gaze directions. However, this assumption is valid mainly when the user looks at a distance. For gaze tracking, the eye-to-camera distance is short (such as 30–40 cm), and we find this method leads to large tracking errors (§6).

Observation. To remove the gaze direction ambiguity, we leverage a mathematical property [19] that when a circle is in translational motion, its normal vector direction keeps unchanged relative to a fixed point. Since the eye movement across two neighboring frames is much smaller than the eye-to-camera distance, the displacement of the iris plane over these two frames is nearly translational motion. Thus, we can choose the gaze direction that accumulates the least rotation change as the result, as shown by the red vectors in Figure 10(a). Another iris plane, associated to the false gaze direction, does not completely follow the translational motion (it also has certain rotations). Therefore, the false direction leads to a larger rotation change, as shown by the black vectors in Figure 10(a).

To verify this observation, we conduct an experiment, where a user stares at a straight line of length 35 cm on the screen. For the first frame, we can obtain $\vec{n}_1^1$ and $\vec{n}_2^1$, where $\vec{n}_2^1$ is the true gaze direction according to our collected ground truth. Then, for any following frame t, where $t = 1+\Delta t, 1+2\Delta t, 1+3\Delta t, \ldots$ and $\Delta = 20$ in this experiment, we calculate the accumulated rotation differences for the sequence of false gaze directions $\{\vec{n}_1^t\}$ and the sequence of correct directions $\{\vec{n}_2^t\}$. Figure 10(b) shows that the accumulated rotation difference of $\{\vec{n}_1^t\}$ is much larger than that of $\{\vec{n}_2^t\}$.

Solution. Inspired by this observation, we can remove the ambiguity during the system setup stage. As introduced before in §2.2, in the first step of setup, the user looks freely at any positions on the tracking area, and the phone records eye frames. Therefore, we can obtain two sequences of gaze directions, *i.e.*, $\{\vec{n}_1^t\}$ and $\{\vec{n}_2^t\}$. Then, we calculate $CRD_1 = \sum_{t=1}^{T-1} ||\vec{n}_1^{t+1} - \vec{n}_1^t||$ and $CRD_2 = \sum_{t=1}^{T-1} ||\vec{n}_2^{t+1} - \vec{n}_2^t||$ to quantify the cumulative rotation differences for $\{\vec{n}_1^t\}$ and $\{\vec{n}_2^t\}$ respectively, and make the following comparison to obtain the correct gaze direction $\vec{n}$:

$$\vec{n} = \begin{cases} \vec{n}_1, & CRD_1 < CRD_2, \\ \vec{n}_2, & CRD_1 > CRD_2. \end{cases} \tag{9}$$

After the setup above, as long as the relative position of the phone to the tracking area is unchanged, the relative direction of $\{\vec{n}_1^t\}$ and $\{\vec{n}_2^t\}$ are also unchanged. Hence, the ambiguity removal does not need to be conducted at run time. The gaze ray estimator can directly fetch the correct gaze direction $\vec{n}$ obtained in the setup. If the position of the mobile is changed, the setup needs to be performed again to update $\vec{n}$ for the subsequent gaze tracking.

5 MAPPING

In this section, we introduce the design of the mapping module.

5.1 Mapping Principle

The mapping module aims to determine the relative position of the tracking surface and the camera coordinate system. With this information, we can obtain the gaze point on the tracking surface, which is the intersection between tracking surface and gaze ray. The mapping relationship is also built during system setup.

The tracking surface has its own coordinate system (TCS), as shown in Figure 11(a), and the mapping module wants to determine the position for the intersection of the gaze ray and the x-y plane of TCS, denoted as $p_{tcs} = (x_t, y_t)$. In practice, users can place their mobile phone nearly parallel to the tracking surface for the easy of mapping. Therefore, the x-y plane of TCS is parallel to that of CCS, and we can first compute the intersection $p_{ccs} = (x_c, y_c)$ of gaze ray $[\vec{n}, o]$ and the x-y plane of CCS as follows:

$$\begin{cases} x_c = x_o - z_o \frac{n_x}{n_z}, \\ y_c = y_o - z_o \frac{n_y}{n_z}, \end{cases} \tag{10}$$

where x_o, y_o and z_o are the coordinates of gaze origin o along three axes and n_x, n_y and n_z are the coordinates of gaze direction $\vec{n}$ along three axes. Then, gaze point $p_{tcs} = (x_t, y_t)$ can be obtained by:

$$\begin{cases} x_t = x_c + \Delta x, \\ y_t = y_c + \Delta y, \end{cases} \tag{11}$$

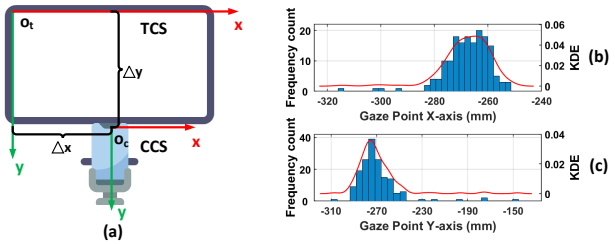

Figure 11: (a) Offsets between TCS and CCS. (b-c) Distribution of x_c and y_c measurements along x- and y-axis, respectively.

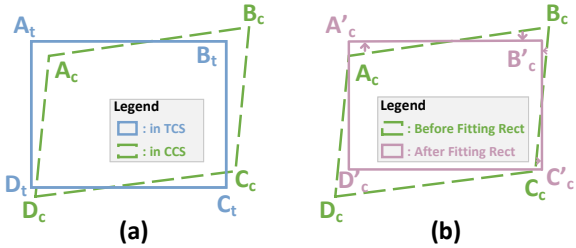

Figure 12: (a) Illustration of the offset deviation. (b) Selecting top-3 estimation results to fit a rectangle to refine the offsets Δx and Δy for more accurate gaze point mapping.

where Δx and Δy are the offset between CCS's origin and TCS's origin along x and y axes, respectively. The mapping module aims to obtain the values of Δx and Δy.

5.2 Proposed Mapping Method

The offsets Δx and Δy are only determined by the relative position between the mobile phone and the tracking surface. As long as this position remains the same, these two offsets keep unchanged. Thus, we can compute them in advance during the system setup phase.

5.2.1 Handling offset errors. In principle, when a user looks at one point in TCS with known location x_t and y_t (such as a corner), if we can obtain the accurate gaze point in CCS (x_c and y_c in Eq. (10)), the offsets Δx and Δy can be obtained through Eq. (11) directly, *i.e.*, $\Delta x = x_t - x_c$ and $\Delta y = y_t - y_c$. However, the estimated x_c and y_c contain errors, which can lead to an inaccurate offset estimation.

Observation. To address this issue, we observe that although each individual (x_c and y_c) estimation may contain an uncertain error, if the user can look at this point for a while, we can obtain a series of x_c and y_c measurements and their average values could give a more reliable and accurate estimation.

Figure 11(b) and (c) show one example, where the user stares at one point $x_c = -237$ mm and $y_c = -272$ mm for five seconds (we use a ruler to measure the offsets in this example). We obtain a set of $\{x_c\}$ and a set of $\{y_c\}$. For all the measurements in $\{x_c\}$, we plot their distributions along the x-axis. In particular, we divide the entire x-axis into multiple bins and compute the frequency of $\{x_c\}$ falling into each bin in Figure 11(b). We plot a similar histogram for the y-axis in Figure 11(c). From the results, we observe that the average of measurements along each axis is indeed closer to the intended point. Therefore, we can use the average of multiple measurements to estimate the offsets Δx and Δy. However, we need to know how many measurements are enough. It is related to the latency of the setup phase, which should be minimized.

Solution. To this end, we fit these two distributions using kernel density estimation [41]. We can see that each histogram approximately follows a Gaussian distribution. Thus, we can leverage the t-distribution [15] to quantify the possibility that the average value of the fitted Gaussian distribution gets close enough to the true (yet unknown) mean value of this Gaussian distribution [32]. Next, we use the x-axis to explain the design, which can be applied to the y-axis directly. If there are n estimations on the x-axis, we have:

$$P(-t_{\frac{\alpha}{2}} < t = \frac{\bar{x} - \mu}{s/\sqrt{n}} < t_{\frac{\alpha}{2}}) = 1 - \alpha, \tag{12}$$

where $\bar{x}$ is the mean value of these n estimations, μ and s are the mean value and the standard deviation of the fitted Gaussian distribution respectively, $1 - \alpha$ represents a pre-defined confidence level, and $t_{\frac{\alpha}{2}}$ are some probability values that can be obtained from the look-up table of the t-distribution [3].

Eq. (12) suggests that the confidence range of μ is $(\bar{x} - \frac{s}{\sqrt{n}} t_{\frac{\alpha}{2}}, \bar{x} + \frac{s}{\sqrt{n}} t_{\frac{\alpha}{2}})$. When more measurements are obtained, n is increased and the confidence range becomes smaller, which suggests that the estimated average value μ of the Gaussian distribution is in a smaller neighborhood range to its true average value. Therefore, with a threshold η for the confidence range, when the condition of $\frac{s}{\sqrt{n}} t_{\frac{\alpha}{2}} \leq \eta$ is met, this μ is used as the final estimation of x_c. A similar calculation can be performed for y_c along the y-axis.

5.2.2 Shape-constrained offset refinement. For any point in TCS, the solution proposed above can obtain the offsets Δx and Δy already. In this subsection, we observe another opportunity to further enhance the accuracy of the estimated offsets. The key idea is that the tracking surface is usually a rectangle (or square). If the users looks at the four corners of the tracking surface, we can further leverage its shape as an additional constraint to improve the accuracy of the computed Δx and Δy.

As shown in Figure 12(a), we denote the x and y coordinates of the tracking surface's four corners in TCS as (x_t^A, y_t^A), (x_t^B, y_t^B), (x_t^C, y_t^C) and (x_t^D, y_t^D). Then, for each corner, we can use the solution proposed above to obtain our estimated coordinates (x_c^A, y_c^A), (x_c^B, y_c^B), (x_c^C, y_c^C) and (x_c^D, y_c^D). Due to the estimation error, the four estimated points A_c, B_c, C_c and D_c cannot form a regular rectangle. Because we can use any three points to form a regular rectangle, we propose to select only three estimated points, from which the formed rectangle has the smallest difference compared to the actual rectangle on the tracking surface as follows.

Step 1: We first use points A_c, B_c, C_c to fit a rectangle as shown in Figure 12(b). Its vertexes are: $A'_c(x_c^A, \frac{y_c^A + y_c^B}{2})$, $B'_c(\frac{x_c^B + x_c^C}{2}, \frac{y_c^A + y_c^B}{2})$, $C'_c(\frac{x_c^B + x_c^C}{2}, y_c^C)$ and $D'_c(x_c^A, y_c^C)$.

Step 2: We repeat the operation of Step 1 three times by selecting B_c, C_c, D_c and A_c, C_c, D_c and A_c, B_c, D_c, respectively.

Step 3: For each of the four fitted rectangles above, we calculate the width and height differences compared to the actual rectangle, denoted as Δw and Δh. We choose the fitted rectangle $\hat{A}_c, \hat{B}_c, \hat{C}_c, \hat{D}_c$ with the smallest difference of $\frac{\Delta w + \Delta h}{2}$. Therefore, the offsets can

be further refined as follows:

$$\begin{cases} \Delta x = \frac{1}{4} \sum_{j \in \{A,B,C,D\}} ||\hat{x}_c^j - x_t^j||, \\ \Delta y = \frac{1}{4} \sum_{j \in \{A,B,C,D\}} ||\hat{y}_c^j - y_t^j||, \end{cases}$$

After obtaining the offsets Δx and Δy between TCS and CCS, the gaze points on the tracking surface can be obtained by Eq. (11).

6 EVALUATION

6.1 Experimental Setup

We implement ASGaze using the RGB camera of iPhone 11 Pro. The captured eye video frames are pre-processed prior to gaze tracking, where the eye aspect ratio (EAR) [48] is first calculated to remove the frames with eye blinks and the Dlib library [45] is then used to detect facial landmarks to locate and save the region of the user's two eyes from each frame. Each eye region is cropped to the resolution of 321×321 pixels. Before the network training, all training data (to be elaborated below) are augmented through flipping, translation, rotation and adding noises to improve the accuracy and generalization of the neural network. We implement the neural network of ASGaze using PyTorch 1.10. We adopt the same backbone VGG-16 [47] for U-Net in iris boundary detector, and use stochastic gradient descent to optimize the network with an initial learning rate of 0.1 in the training. We then decay the learning rate every twenty steps according to the strategy of $rate = initial\_rate \times (decay\_rate)^{epoch/step\_size}$, with a decay rate of 0.5. The minimum learning rate is set to 0.0005 and the network is trained for 200 epochs with a GeForce RTX 2080Ti GPU.

Data collection. For a comprehensive evaluation, we examine the performance of ASGaze using our collected datasets from three different tracking surfaces (Figure 13) and one public dataset. We recruit eight users to participate into our data collection with the ethics approval by our university. They include three female and five male users, aged between 21 and 28. Three of them have normal vision, and the remaining users are near-sighted. These users have an iris radius of 5.45–6.75 mm. One advantage of using iris boundaries to derive gaze points is that we do not need to collect the ground truth of gaze points to train ASGaze. Instead, we only need to annotate three areas (iris, sclera or background) for each eye frame from the training dataset. We have developed a tool for this annotation with a few mouse clicks. Such labeled data are used to train the iris boundary detector, which outputs iris boundaries to derive gaze points. Furthermore, for each tracking surface, users also look at four corners sequentially during data collection. We use the minimum set of frames, determined in §5, to map gaze points to the tracking surface. Next, we introduce the three tracking surfaces.

1) Computer monitor (CMonitor). We place the mobile phone under a 24-inch computer monitor and use its telephoto camera to capture user's eyes at 30 frames per second (fps). Each user naturally looks at the monitor and the user's eyes follow the trajectories displayed on the monitor, which covers almost the entire monitor. During the data collection, we also use a commercial gaze tracker, Tobii [4], to collect the ground truth for evaluation. The default eye-to-screen distance is about 40 cm. We have collected 24 eye-movement video clips with 14760 frames, where each frame can produce two images for the user's left and right eyes. We have labeled 2036 eye images (1018 images for each eye) as the training dataset and leave all remaining ones as the testing dataset.

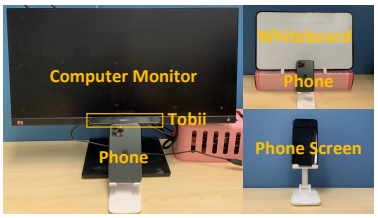

Figure 13: Experimental setup and tracking surfaces.

2) Whiteboard (WBoard). We place the mobile phone under a whiteboard, which is 42 cm × 30 cm, and use phone's telephoto camera to capture user's eyes. The default eye-to-screen distance is about 40 cm. Because Tobii cannot work on the whiteboard to collect the ground truth of gaze points for evaluation. We draw a 3×3 grid on the whiteboard and use each grid point as anchors for evaluation, where the location of each grid point on the whiteboard is known in advance. Through our gaze point mapping design (§5), we already setup the coordinate system for the whiteboard and the location of each grid point can be marked in the coordinate system. Later in the evaluation, users look at each of these grid points and the distance between the estimated gaze points and the targeted grid point measures the tracking error. For whiteboard, we have collected nine video clips from each user. We annotate 10% frames for training and use the remaining ones for evaluation.

3) Phone screen (PScreen). Finally, we track the user's eyes on the screen of the mobile phone directly. The screen size is 7.14 cm × 14.4 cm, and the default eye-to-screen distance is 30 cm. Similar to whiteboard, we cannot use Tobii to collect ground truth. Thus, we adopt the same method by using a 3 × 3 grid for evaluation. For PScreen, we also collect nine videos from each user. We annotate 10% frames for training and use the remaining ones for evaluation. Both PScreen and CMonitor are electronic screens. Their difference is that in CMonitor the eyes are captured by the phone but the tracking plane is on a different device (computer screen), while in PScreen they are on the same device.

In addition to our collected datasets, we also use a public dataset, EVE-DS [40], for evaluation. EVE-DS is a video-based gaze tracking dataset on computer monitor, which can directly evaluate our system components that require temporal features from consecutive frames. EVE-DS is collected with 54 users from three webcams and one machine vision camera at 1920 × 1080 resolution. Four cameras cover four viewpoints: below, up center, up left, and up right. Because the associated gaze tracking design of this dataset can leverage the content displayed on the screen, which is also recorded in the dataset. For evaluation, we randomly select 20 videos (with 5271 frames) of the below viewpoint from five users. We annotate 10% frames for training and leave the remaining ones for evaluation.

Evaluation metric. In the evaluation, each estimated gaze point is the average from both eyes, and we use the Euclidean distance between the estimated gaze point and its ground truth value as the performance metric to measure the tracking error. As stated above, the ground truth can be obtained from Tobii for CMonitor and the grid points marked for both WBoard and PScreen. For the public dataset EVE-DS, the ground truth of gaze points is provided.

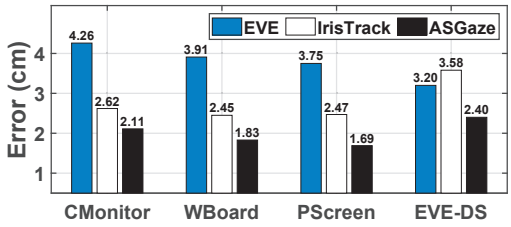

Figure 14: Tracking performance for each tracking surface compared with EVE [40] and IrisTrack [33]

6.2 Overall Performance

First, we compare ASGaze with the following two methods:

- **EVE** [40]: it is a recent appearance-based gaze tracking method using eye video frames.
- **IrisTrack** [33]: it is the state-of-the-art iris boundary detection method. We supplement our designs that convert iris boundaries to gaze points to IrisTrack in the evaluation.

In Figure 14, the average tracking error of EVE is 3.20–4.26 cm.[4] The error on EVE-DS is smaller than that of other three tracking surfaces, because EVE can leverage the contents displayed on the screen, such as images, videos and Wikipedia, to improve accuracy. Since such contents are not always available in practice, we include this mechanism for its own dataset EVE-DS but skip it for other three tracking surfaces (where such contents are not available). Compared to EVE, IrisTrack reduces the tracking error to 2.45–3.58 cm. IrisTrack achieves a relatively large error for the dataset of EVE-DS. Through our investigation, we find that it is mainly because EVE-DS contains more low-quality (*e.g.*, blurred) and low-resolution eye images, which may affect the accuracy of iris boundaries obtained by IrisTrack. With the designs proposed in this paper, ASGaze outperforms these two methods and further reduces the error to 1.69–2.40 cm, improving EVE and IrisTrack by 25%–55% and 19%–33%, respectively.

In Figure 15, we visualize the results of ASGaze for a zigzag moving trajectory. We can see that the estimated gaze points follow the ground truth trajectory very well. From the result, we also observe that the tracking performance is better for the upper-part of the trajectory. This is because the mobile phone is placed under the computer monitor, the iris is covered less by eyelids when the user looks at the upper-part of the monitor, where the average tracking error is only around 1 cm for the upper-part trajectory. Thus, if only a partial area needs the gaze tracking function in certain applications, we can arrange the upper-part of the monitor as a tracking area for better performance.

6.3 Ablation Study

Next, we conduct an ablation study to understand the efficacy of each technical designs proposed in ASGaze.

Iris boundary detector module. Iris boundary detector contains three major components, including the segmentation network, new loss functions and refinement of iris boundaries. Therefore, we develop three intermediate versions of ASGaze as follows:

- **BL**: this is a baseline version that is developed by removing our loss functions and refinement of iris boundaries.
- **BL+RF**: this version adds the refinement of iris boundaries on top of the previous baseline version.
- **FULL**: this is the full version by adding our loss functions on top of the previous "BL+RF" version.

	CMonitor	WBoard	PScreen	EVE-DS
BL	2.65 cm	2.66 cm	2.91 cm	6.18 cm
BL+RF	2.39 cm	2.18 cm	2.27 cm	4.18 cm
FULL	2.11 cm	1.83 cm	1.69 cm	2.40 cm

Table 1: Efficacy of the iris boundary detector module.

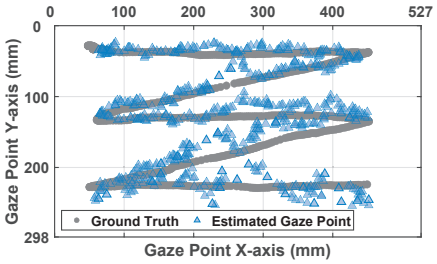

Figure 15: Visualization of estimated gaze points by ASGaze.

Table 1 summarizes the results. The baseline version achieves relatively large tracking error 2.65–6.18 cm on average. With the refinement scheme, the tracking error is reduced to 2.18–4.18 cm, with the improvement of 9.8%–32.4%. The new loss functions can further reduce the tracking error to 1.69–2.40 cm, with the improvement of 11.7%–42.6%. The loss functions and refinement scheme bring comparable performance gains in ASGaze.

	CMonitor	WBoard	PScreen	EVE-DS
"two-circle"	13.04 cm	11.53 cm	6.05 cm	12.27 cm
ASGaze	2.11 cm	1.83 cm	1.69 cm	2.40 cm

Table 2: Efficacy of the ambiguity removal module.

Ambiguity removal of gaze points. To remove the ambiguity when iris boundaries are converted to gaze points, we have proposed a method in §4.2. As stated before, there is an existing "two-circle" method. Its main idea is that when a user looks at infinity, the gaze directions of two eyes are nearly parallel. Therefore, the gaze points from two eyes that lead to the smallest angle are treated to the correct ones. These two methods are compared in Table 2. We can see that "two-circle" has much larger errors (up to 13.04 cm) compared to ASGaze, because the gaze directions are not parallel when the user looks at a screen or monitor in practice.

	CMonitor	WBoard	PScreen
Measured	2.10 cm	1.77 cm	1.65 cm
ASGaze	2.11 cm	1.83 cm	1.69 cm

Table 3: Efficacy of the mapping module.

[4] As stated in §1, it is difficult for appearance-based methods to work on a non-electronic surface. For the evaluation purpose on WBoard, we use the locations of all the grid points as ground truth to train EVE. Thus, EVE is mainly effective to derive gaze points close to these grids (not the entire whiteboard), which is the limitation of appearance-based methods, while IrisTrack and ASGaze do no suffer this limitation.

Mapping of gaze points. Finally, we study the effectiveness when we map gaze points from the coordinate system of camera to the tracking surface. To this end, we use a ruler to measure the actual offset between the origins of these two different planes. Then, we use the computed result to map gaze points to each tracking surface. Table 3 shows that the tracking error of ASGaze is similar to that obtained from the offset measurement, which suggests that our proposed mapping scheme is effective.

6.4 Micro-benchmarks

In this subsection, we examine the performance of ASGaze under a series of related factors in practice.

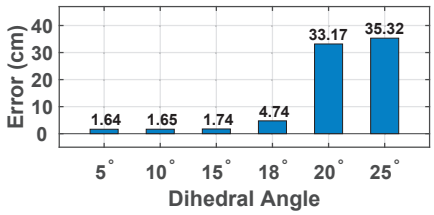

Figure 16: Performance with different dihedral angles.

Different dihedral angles. In our current design, the dihedral angle between phone and tracking surface should be nearly parallel. In Figure 16, we examine the system performance by changing this angle. We can see that when the dihedral angle is relatively small, *e.g.*, within 15°, the influence is limited, and the increased error is small. This indicates that the tracking performance of ASGaze is robust and can tolerate certain dihedral angles in practice. However, when this angle is further increased, *e.g.*, more than 15°, the tracking error increases rapidly. This is because when the dihedral angle is large, the relative position between the planes of the phone and the surface cannot simply be compensated in the same way as when they were in parallel. In this case, the "Mapping" module cannot compensate the offset between them well, resulting in the deterioration of the tracking performance. In the future, we plan to further reduce this system setup requirement.

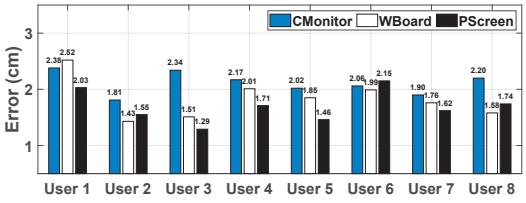

Figure 17: Tracking performance for each individual user.

Different users. Figure 17 shows the detailed tracking performance for each individual user on each tracking surface. Even though users have different sizes of eye and iris, the geometry-based gaze tracking designs are robust to such differences. Therefore, ASGaze can achieve similar tracking performance among various users on each tracking surface. User 1 has a larger error on WBoard and PScreen than other users because this user sometimes looked elsewhere during the data collection, rather than focusing on the

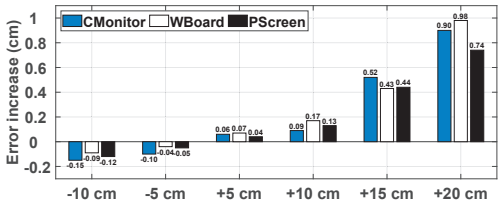

Figure 18: Tracking performance at different working distances on each tracking surface.

required locations (Tobii cannot work on these two screens), and we did not exclude such frames from our evaluation.

Different working distances. In this experiment, we investigate the system performance at different working distances, as shown in Figure 18. For "CMonitor" and "WBoard", the default working distance is set to 40 cm. For "PScreen", the default distance is 30 cm. We vary the working distance by increasing or decreasing it relative to the default distance. Figure 18 shows that when the working distance offset is moderate, such as from a decrease of 10 cm to an increase of 10 cm, the tracking error changes of "CMonitor", "WBoard" and "PScreen" are 0.06–0.15 cm, 0.04–0.17 cm, 0.04–0.13 cm, respectively. The error change is small within this range, indicating that ASGaze can perform well at moderate working distances. When the distance is further increased, the increased error becomes larger, such as 0.43–0.52 cm. When the distance offset reaches 20 cm, the increased tracking error is large. This is because the eyes are small in the captured eye frames, which reduces the accuracy of the iris detected by the "Iris Boundary Detector".

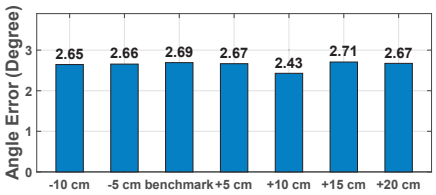

Figure 19: Angular errors at different working distances.

In addition, we also examine the angular error of our system at different working distances. Since the angular error needs to be measured with Tobii, we only conduct this experiment for "CMonitor". Figure 19 shows that the average angular error varies from 2.43–2.69° when the working distance offset (relative to the default working distance of 40 cm) is -10 cm to 10 cm. Given a fixed Euclidean error on the screen, as the distance between the user's eye and screen increases, the corresponding angular error decreases. Since the Euclidean error increases slower than the working distance in Figure 18, the angular error does not increase significantly when the working distance offset is relatively large, *e.g.*, 15–20 cm.

Different head poses. In this experiment, we investigate the impact of user's head poses by varying head pose intentionally. For a comprehensive evaluation, head pose is rotated along both vertical and horizontal two directions by 10°, 15°, 20° and 25°. For each rotation angle, we measure the increased error with respect to the error obtained without rotation. We can see from Figure 20(a) that during natural vertical rotation of user's head pose, *e.g.*, within 20° for "CMonitor" and "WBoard" and within 10° for "PScreen",

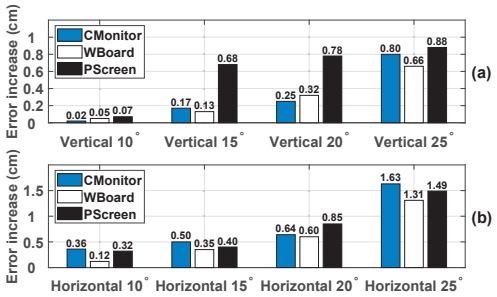

Figure 20: Tracking performance with different (a) vertical and (b) horizontal rotation angles on each tracking surface.

ASGaze is robust, where the maximum error increase is only 0.32 cm. However, when the rotation becomes larger, the error increases rapidly. This is mainly because a larger vertical rotation can significantly affect the size of captured iris. Figure 20(b) further shows the effect of the head horizontal rotations, where the error increase is small for all three surfaces when the rotation angle is within 15°. In addition, the effect of horizontal rotation is slightly larger than the vertical rotation, because horizontal rotation may cause the iris to move closer to the corner of the eye, which causes a larger portion of the iris boundary to be obscured by the eyelashes and eyelid.

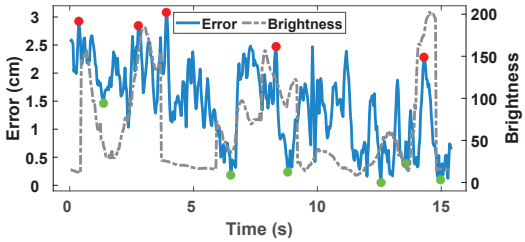

Figure 21: Performance under different screen brightness.

Screen brightness. In this experiment, the screen is playing a firework explosion video with high content dynamics and varying brightness. When users watch this video, their gaze points can move freely, and we use Tobii to collect the ground truth. From the firework explosion video, we obtain the HSB (Hue, Saturation, Brightness) value of each frame to calculate its average B-channel value, which represents the overall brightness of each frame, as shown by the grey dashed line in Figure 21. The corresponding tracking performance of ASGaze is plotted by the blue curve. From the results, we can see that when the screen brightness increases, the tracking error also tends to increase, as shown by the red markers in the figure. This is mainly because strong screen brightness can cause reflections on the eyeball that affect the quality of the captured eye frames. When the brightness is relatively small, the tracking error tends to decrease, as shown by green markers.

Environmental illuminations. Since ASGaze uses a common RGB camera, we also examine the tracking performance under different environmental illuminations. In particular, we conduct this experiment in three typical scenarios: indoor under normal indoor light (~300 Lux), indoors with the overhead light turned off (~10 Lux) and next to the window clear sunlight (~1100 Lux). Figure 22 shows that ASGaze performs best under the normal light

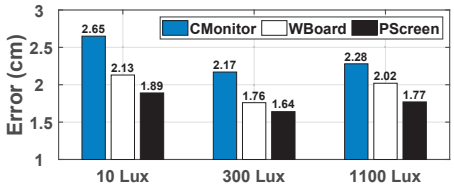

Figure 22: Tracking performance for different environmental illuminations on each tracking surface.

condition (300 Lux). When illumination becomes stronger, the error is increased slightly, *i.e.*, 0.11–0.26 cm. When illumination is very weak, the tracking error becomes larger, *i.e.*, 0.25–0.48 cm. We find that this is because the cropped eye images become darker when the illumination is weak. The segmentation network then may confuse between background and iris, which causes more tracking errors.

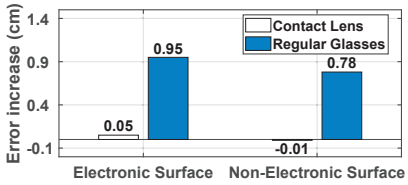

Figure 23: Performance when wearing glasses.

Wearing glasses. In Figure 23, we examine system performance when users wear contact lens and regular glasses on both a non-electronic screen ("WBoard") and an electronic screen ("CMonitor"). Compared to the performance when users do not wear glasses on these screens, we can see that wearing contact lens does not affect tracking performance, while wearing regular glasses increases tracking error by 0.78 cm and 0.95 cm, respectively. This is mainly due to the distorted iris boundary shape when capturing the spherical aberration of the eye through glasses, thereby degrading the tracking performance. In addition, reflections of external light on the glasses can also affect tracking performance.

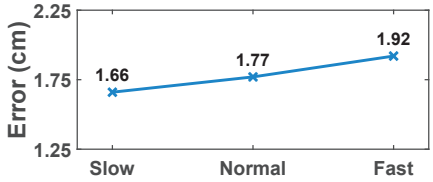

Figure 24: Performance under different moving speeds.

Different moving speeds with a phone. When ASGaze tracks gaze points on the phone's screen, it is possible that ASGaze is used when the user holds the phone and is walking. In this experiment, we investigate the tracking performance when the user walks under three different moving speeds: slow (less than one step per second), normal (~1.5 steps per second) and fast (~two steps per second). Figure 24 shows that the error becomes larger as the speed increases. This is because with a faster movement, the phone is more likely to be shaken by the user's hand, which may produce more blurred eye images and cause more differences between consecutive frames to affect the gaze tracking performance.

Computation overhead. Finally, we examine the computation overhead of ASGaze. In our current development, computations

are performed on a computer with an i-7 CPU. We will transfer all the computations to the mobile phone in the future. Table 4 shows that the setup (for ambiguity removal and gaze point mapping) needs 4.19 second to complete, which is an one-time effort. During gaze tracking, the execution time of the segmentation network and refinement in iris boundary detector for one frame are 47.58 ms and 36.25 ms, respectively. The execution time to convert one iris boundary to the corresponding gaze point is 5.30 ms per frame. Therefore, ASGaze can output 11 gaze points per second with our current implementation. Note that our current design prioritizes tracking accuracy and is still heavy to execute directly on the mobile phone. In the future, we plan to reduce its computational overhead by exploring a more efficient backbone network, leveraging compression techniques and conducting code optimization to make system implementation more efficient.

Setup (one-time effort)	4.19 s (= 0.62 s + 3.57 s)
Iris detector (per frame)	83.83 ms (= 47.58 ms + 36.25 ms)
Gaze estimation (per frame)	5.30 ms

Table 4: Computation overhead of each module in ASGaze.

7 RELATED WORKS

Gaze tracking methods. In the literature, gaze tracking systems were developed on smart glasses, such as iGaze [61], iShadow [36], CIDER [37], LiGaze [30], battery-free tracker [31], etc. Because smart glasses are close to eyes and the distance between camera-to-eye keeps relatively stable, these systems can achieve accurate tracking. Moreover, many of them use near-infrared (NIR) lights [31, 37], which can further improve the performance. Parallel to the wearable-based solutions, people are also interested in gaze tracking on other platforms, such as computers, mobiles, etc. However, the relative rotation and distance between eyes and camera can change, which fundamentally challenges the gaze tracking design on such platforms. Commercial trackers, such as Tobii [5], EyeLink [1], Pupil Labs [2], can cope with this issue by using dedicated sensors and hardware, which however are expensive usually.

Recently, due to the achievements of deep learning, a variety of appearance-based methods have been proposed, by using a common RGB webcam [9, 17, 40, 62] or the camera of a mobile phone [13, 21, 27, 39]. The key idea of the appearance-based methods is to collect a large number of eye images and the ground truth of the corresponding gaze points, which try to cover different head poses, eye-to-camera distances and user diversities. With such a dataset, they adopt a neural network to learn the transformation from eye images to gaze points. The merit of these methods is to avoid dedicated sensors and hardware. However, their performance depends on large datasets to cover these factors that usually cause high data collection overhead. Furthermore, existing appearance-based methods that estimate gaze points on tracking surfaces mainly work on electronic screens, as they need to display the ground truth of gaze points to collect training data. To overcome these shortcomings, vGaze [57] recently leverages the visual saliency with the depth camera of a phone to achieve accurate tracking. In this paper, we revisit the geometric model of eyes to achieve accurate tracking with a common RGB camera and further extend the gaze tracking ability on external tracking surfaces.

Iris boundary detection. Iris boundary detection is a crucial component of ASGaze. The traditional iris boundary detection methods are mainly based on image processing, such as Hough transform [59] and contour-based technique [26], which are sensitive to environmental conditions. Recently, researchers employ attention-based neural network [54] to achieve iris detection, which greatly improves the performance. However, they annotate a thick iris boundary to provide enough features for neural network to learn, which could cause the result to contain many non-boundary noisy pixels. Therefore, the the state-of-the-art method [33] further converts the iris boundary detection problem to a segmentation problem and use entropy to select boundary pixels to achieve more accurate results. In this paper, we find that the results of [33] are still not sufficient for gaze tracking and we propose a series of effective designs to achieve a better iris boundary detection result.

Applications of gaze tracking. Gaze tracking can enable a variety of useful applications. For HCI, it can be used to protect the user's typing privacy [32] and correct the text input error [20, 63]. For example, gaze tracking data can accelerate the navigation menu [58] and improve the path prediction [49] in virtual reality applications. For smart health, analysing gaze movements can detect driver's distraction [35, 60]. For people with disabilities who cannot move their limbs, gaze tracking can be used as alternative input method [51]. The gaze information can also assist computer-aided diagnosis [56]. For analysis of user's interests, gaze tracking can provide useful information of students, so that teachers can adjust teaching strategies in time [6]. Gaze information can be used in retail stores as well to optimize the placements of goods [42]. Moreover, the gaze information can also be used to sense the cognitive behaviors of a user, such as watching a video or reading article [28, 29].

8 CONCLUSION

This paper presents ASGaze, a new gaze tracking design that follows the eye model-based approach employed by commercial gaze tracking solutions. ASGaze demonstrates the ability to achieve good tracking accuracy with a common RGB camera on a variety of tracking surfaces commonly required by different applications. Unlike commercial solutions with dedicated camera sensors and hardware, the design of ASGaze encounters three unique challenges, and we propose effective solutions to address all these challenges in this paper. We develop a prototype system of ASGaze and conduct extensive experiments to evaluate its performance. The results show encouraging performance under a variety of datasets and settings and outperforms the recent method.

ACKNOWLEDGEMENTS

We sincerely thank anonymous reviewers for their helpful comments to improve the quality of this paper. We also thank all volunteers for their participation in our experiments. This work is supported by the GRF grants from Research Grants Council of Hong Kong (CityU 11217420 and CityU 11213622) and the CityU Applied Research Grant (ARG) under Grant 9667205. Corresponding authors: Yang Liu and Zhenjiang Li.

REFERENCES

[1] Eyelink products. https://www.sr-research.com/hardware/.
[2] Pupil labs products. https://pupil-labs.com/products/.
[3] T-distribution table. https://www.statisticshowto.com/tables/t-distribution-table/.
[4] Tobii pro nano. https://www.tobiipro.com/product-listing/nano/.
[5] Tobii products. https://www.tobiipro.com/product-listing/.
[6] K. Ahuja, D. Shah, S. Pareddy, F. Xhakaj, A. Ogan, Y. Agarwal, and C. Harrison. Classroom digital twins with instrumentation-free gaze tracking. In *Proc. of ACM CHI*, 2021.
[7] P. Artal. Optics of the eye and its impact in vision: a tutorial. *Advances in Optics and Photonics*, 2014.
[8] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proc. of Springer ECCV*, 2006.
[9] P. Biswas et al. Appearance-based gaze estimation using attention and difference mechanism. In *Proc. of the IEEE CVPR*, 2021.
[10] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 1986.
[11] A. K. Chaudhary, R. Kothari, M. Acharya, S. Dangi, N. Nair, R. Bailey, C. Kanan, G. Diaz, and J. B. Pelz. Ritnet: Real-time semantic segmentation of the eye for gaze tracking. In *Proc. of IEEE ICCVW*, 2019.
[12] A. Consejo, C. Llorens-Quintana, H. Radhakrishnan, and D. R. Iskander. Mean shape of the human limbus. *Journal of Cataract & Refractive Surgery*, 2017.
[13] L. Dai, J. Liu, and Z. Ju. Binocular feature fusion and spatial attention mechanism based gaze tracking. *IEEE Transactions on Human-Machine Systems*, 2022.
[14] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on pattern analysis and machine intelligence*, 1999.
[15] C. Forbes, M. Evans, N. Hastings, and B. Peacock. Student'st distribution. *Statistical Distributions*, 2010.
[16] D. Forsyth and J. Ponce. *Computer vision: A modern approach*. Prentice hall, 2011.
[17] J. Gideon, S. Su, and S. Stent. Unsupervised multi-view gaze representation learning. In *Proc. of the IEEE CVPR*, 2022.
[18] D. W. Hansen and P. Majaranta. Basics of camera-based gaze tracking. In *Gaze interaction and applications of eye tracking: Advances in assistive technologies*. 2012.
[19] D. He and B. Benhabib. Solving the orientation-duality problem for a circular feature in motion. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 1998.
[20] Z. He, C. Lutteroth, and K. Perlin. Tapgazer: Text entry with finger tapping and gaze-directed word selection. In *Proc. of ACM CHI*, 2022.
[21] S. Huynh, R. K. Balan, and J. Ko. imon: Appearance-based gaze tracking system on mobile devices. *Proc. of ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2021.
[22] F. Jafarlou, F. Jarollahi, M. Ahadi, and V. Sadeghi-Firoozabadi. Effects of oculomotor rehabilitation on the cognitive performance of dyslexic children with concurrent eye movement abnormalities. *Early Child Development and Care*, 2022.
[23] F. Jafarlou, F. Jarollahi, M. Ahadi, V. Sadeghi-Firoozabadi, and H. Haghani. Oculomotor rehabilitation in children with dyslexia. *Medical journal of the Islamic Republic of Iran*, 2017.
[24] A. Jakubović and J. Velagić. Image feature matching and object detection using brute-force matchers. In *Proc. of IEEE ELMAR*, 2018.
[25] H. Kervadec, J. Bouchtiba, C. Desrosiers, E. Granger, J. Dolz, and I. B. Ayed. Boundary loss for highly unbalanced segmentation. In *Proc. of MIDL*, 2019.
[26] J. Koh, V. Govindaraju, and V. Chaudhary. A robust iris localization method using an active contour model and hough transform. In *Proc. of the IEEE ICPR*, 2010.
[27] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *Proc. of IEEE CVPR*, 2016.
[28] G. Lan, B. Heit, T. Scargill, and M. Gorlatova. Gazegraph: Graph-based few-shot cognitive context sensing from human visual behavior. In *Proc. of ACM SenSys*, 2020.
[29] G. Lan, T. Scargill, and M. Gorlatova. Eyesyn: Psychology-inspired eye movement synthesis for gaze-based activity recognition. In *Proc. of ACM/IEEE IPSN*, 2022.
[30] T. Li, Q. Liu, and X. Zhou. Ultra-low power gaze tracking for virtual reality. In *Proc. of ACM SenSys*, 2017.
[31] T. Li and X. Zhou. Battery-free eye tracker on glasses. In *Proc. of ACM MobiCom*, 2018.
[32] Z. Li, M. Li, P. Mohapatra, J. Han, and S. Chen. itype: Using eye gaze to enhance typing privacy. In *Proc. of the IEEE INFOCOM*, 2017.
[33] C. Lin, X. Li, Z. Li, and J. Hou. Finding stars from fireworks: Improving non-cooperative iris tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.
[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proc. of IEEE ICCV*, 2017.
[35] M. LRD, A. Mukhopadhyay, and P. Biswas. Distraction detection in automotive environment using appearance-based gaze estimation. In *Proc. of ACM IUI*, 2022.
[36] A. Mayberry, P. Hu, B. Marlin, C. Salthouse, and D. Ganesan. ishadow: design of a wearable, real-time mobile gaze tracker. In *Proc. of ACM MobiSys*, 2014.
[37] A. Mayberry, Y. Tun, P. Hu, D. Smith-Freedman, D. Ganesan, B. M. Marlin, and C. Salthouse. Cider: Enabling robustness-power tradeoffs on a computational eyeglass. In *Proc. of ACM MobiCom*, 2015.
[38] P. Nawrot, K. P. Michalak, and A. Przekoracka-Krawczyk. Does home-based vision therapy affect symptoms in young adults with convergence insufficiency? *Optica Applicata*, 2013.
[39] J. Park, S. Park, and H. Cha. Gazel: Runtime gaze tracking for smartphones. In *Proc. of IEEE PerCom*, 2021.
[40] S. Park, E. Aksan, X. Zhang, and O. Hilliges. Towards end-to-end video-based eye-tracking. In *Proc. of Springer ECCV*, 2020.
[41] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 1962.
[42] S. Rallapalli, A. Ganesan, K. Chintalapudi, V. N. Padmanabhan, and L. Qiu. Enabling physical analytics in retail stores using smart glasses. In *Proc. of ACM MobiCom*, 2014.
[43] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of Springer MICCAI*, 2015.
[44] R. Safaee-Rad, I. Tchoukanov, K. C. Smith, and B. Benhabib. Three-dimensional location estimation of circular features for machine vision. *IEEE Transactions on Robotics and Automation*, 1992.
[45] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: Database and results. *Image and vision computing*, 2016.
[46] S. Schuett, C. A. Heywood, R. W. Kentridge, and J. Zihl. Rehabilitation of hemianopic dyslexia: are words necessary for re-learning oculomotor control? *Brain*, 2008.
[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
[48] T. Soukupova and J. Cech. Eye blink detection using facial landmarks. In *Proc. of CVWW*, 2016.
[49] N. Stein, G. Bremer, and M. Lappe. Eye tracking-based lstm for locomotion prediction in vr. In *Proc. of IEEE VR*, 2022.
[50] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. 2017.
[51] M. S. H. Sunny, M. I. I. Zarif, I. Rulik, J. Sanjuan, M. H. Rahman, S. I. Ahamed, I. Wang, K. Schultz, and B. Brahmi. Eye-gaze control of a wheelchair mounted 6dof assistive robot for activities of daily living. *Journal of NeuroEngineering and Rehabilitation*, 2021.
[52] J. S. Suri. A geometric model of the 3d human eye and its optical simulation. *Human Eye Imaging and Modeling*, 2012.
[53] R. Venkateswarlu et al. Eye gaze estimation from a single image of one eye. In *Proc. of IEEE ICCV*, 2003.
[54] C. Wang, J. Muhammad, Y. Wang, Z. He, and Z. Sun. Towards complete and accurate iris segmentation using deep multi-task attention network for non-cooperative iris recognition. *IEEE Transactions on information forensics and security*, 2020.
[55] J.-G. Wang and E. Sung. Gaze determination via images of irises. *Image and Vision Computing*, 2001.
[56] S. Wang, X. Ouyang, T. Liu, Q. Wang, and D. Shen. Follow my eye: Using gaze to supervise computer-aided diagnosis. *IEEE Transactions on Medical Imaging*, 2022.
[57] S. Yang, Y. He, and M. Jin. vgaze: Implicit saliency-aware calibration for continuous gaze tracking on mobile devices. In *Proc. of IEEE INFOCOM*, 2021.
[58] X. Yi, Y. Lu, Z. Cai, Z. Wu, Y. Wang, and Y. Shi. Gazedock: Gaze-only menu selection in virtual reality using auto-triggering peripheral menu. In *Proc. of IEEE VR*, 2022.
[59] D. Young, H. Tunley, and R. Samuels. *Specialised hough transform and active contour methods for real-time eye tracking*. University of Sussex, Cognitive & Computing Science Brighton, UK, 1995.
[60] G. Yuan, Y. Wang, H. Yan, and X. Fu. Self-calibrated driver gaze estimation via gaze pattern learning. *Knowledge-Based Systems*, 2022.
[61] L. Zhang, X.-Y. Li, W. Huang, K. Liu, S. Zong, X. Jian, P. Feng, T. Jung, and Y. Liu. It starts with igaze: Visual attention driven networking with smart glasses. In *Proc. of ACM MobiCom*, 2014.
[62] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. *IEEE transactions on pattern analysis and machine intelligence*, 2017.
[63] M. Zhao, H. Huang, Z. Li, R. Liu, W. Cui, K. Toshniwal, A. Goel, A. Wang, X. Zhao, S. Rashidian, et al. Eyesaycorrect: Eye gaze and voice based hands-free text correction for mobile devices. In *Proc. of ACM IUI*, 2022.