Kalɛido: Real-Time Privacy Control for Eye-Tracking Systems

Jingjie Li, Amrita Roy Chowdhury, Kassem Fawaz, and Younghyun Kim
University of Wisconsin–Madison
{jingjie.li, roychowdhur2, kfawaz, younghyun.kim}@wisc.edu

Abstract

Recent advances in sensing and computing technologies have led to the rise of eye-tracking platforms. Ranging from mobiles to high-end mixed reality headsets, a wide spectrum of interactive systems now employs eye-tracking. However, eye gaze data is a rich source of sensitive information that can reveal an individual's physiological and psychological traits. Prior approaches to protecting eye-tracking data suffer from two major drawbacks: they are either incompatible with the current eye-tracking ecosystem or provide no formal privacy guarantee. In this paper, we propose Kalɛido, an eye-tracking data processing system that (1) provides a formal privacy guarantee, (2) integrates seamlessly with existing eye-tracking ecosystems, and (3) operates in real-time. Kalɛido acts as an intermediary protection layer in the software stack of eye-tracking systems. We conduct a comprehensive user study and trace-based analysis to evaluate Kalɛido. Our user study shows that the users enjoy a satisfactory level of utility from Kalɛido. Additionally, we present empirical evidence of Kalɛido's effectiveness in thwarting real-world attacks on eye-tracking data.

1 Introduction

Recent advances in sensing and computing technologies have facilitated the rapid adoption of eye tracking as a hands-free interface in augmented, virtual, and mixed reality settings. It offers users control over virtual components [84], events [51], and digital avatars [80], especially in settings where hand-based control is either impractical or infeasible [89]. Interactive systems are now capable of performing continuous eye tracking using off-the-shelf webcams [66], smartphones [61], tablets [32], desktops [62], wearable glasses [93], and mixed reality headsets such as the HTC VIVE and Microsoft HoloLens.

From a stream of eye gaze positions in a scene, eye-tracking applications precisely estimate what the user is viewing to trigger events, prefetch scenes, or perform actions in the virtual environment. One's eye gaze streams, however, are vulnerable to potential privacy threats. Previous research has demonstrated that psychological and physiological factors direct the formation of unique patterns in the user's eye gazes. For instance, researchers were able to infer insights about the user's behavioral traits [49, 75, 77], diagnose Alzheimer's disease and autism spectrum disorder [30, 41], understand the user's familiarity of a scene [78], infer mental status during social interaction [76], detect personality traits [10], and deliver personalized advertisements [16, 24, 92].

(a) Raw data
(b) Noisy data from **Kalɛido**

Figure 1: Eye gaze heatmaps from an individual user with and without Kalɛido's noising effect on a web page.

Third-party applications that use eye gaze streams can extract information beyond their intended core functionality, posing significant privacy threats to the users. For example, Figure 1(a) shows the heatmap of eye gazes on a web page from an individual user. While an application can help the user scroll up/down the web page, the aggregated eye gaze positions can reveal the user's interest. Unfortunately, the eye-tracking platforms do not offer users the ability to control their privacy. They relay the raw eye gaze streams to the applications without much regard to the embedded sensitive information.

Researchers have developed privacy-preserving mechanisms for eye gaze streams [12, 13, 29, 53, 79] to alleviate these concerns. These mechanisms share a similar working principle: allowing access to only some high-level "features" of the eye gaze streams, possibly with some added noise, instead of the raw gaze streams. While some of them provide formal privacy guarantees [12, 53, 79], they are mostly imprac-

tical to deploy due to multiple limitations. First, they require modification of the eye-tracking application programming interfaces (APIs) since the applications expect to receive a sequence of raw eye gaze positions, not just features. Second, processing eye gaze streams to extract features does not happen in real-time, affecting the user experience. Third, they require the user to control a set of parameters that are hard to understand for most users. In short, the question of how to provide a backward-compatible, easy-to-use privacy-preserving system for real-time eye tracking is still an open one.

In this paper, we design, implement, and evaluate Kalɛido as an affirmative answer to the above question. Kalɛido provides a formal privacy guarantee based on differential privacy (DP) [21], the de-facto standard for achieving data privacy. To the best of our knowledge, Kalɛido is the first system to (1) provide a privacy guarantee on raw eye gazes, (2) seamlessly integrate with the existing eye-tracking ecosystem, and (3) operate in real-time. Kalɛido offers the following advantages:

• **Formal privacy guarantee.** Kalɛido uses a differentially private algorithm to release noisy eye gaze streams to the applications, which *protects the spatial distribution of a gaze trajectory that is formed within any window of a specific duration* (as determined by the users). Kalɛido achieves this objective by bringing the privacy semantics from two distinct contexts, absolute location data and streaming event data, into the domain of eye gaze data (Section 4.3.3). Figure 1(b) shows Kalɛido's privacy protection in action.

• **Seamless integration with the eye-tracking ecosystem.** As Kalɛido operates on raw eye gaze streams, it fits within the existing ecosystem of eye-tracking applications. It is also platform- and application-agnostic; it operates on popular eye-tracking platforms and requires no modification of the applications, making it more practical to deploy.

• **Ease of use.** As the parameters of Kalɛido's privacy guarantee are a function of the visual feed semantics, it reduces the burden of complex privacy configuration on the user.

We integrate Kalɛido as a Unity [26] plugin; it acts as a protection layer between untrusted applications and trusted platforms. Unity is the mainstream engine for gaming and mixed reality applications; it supports various peripherals such as eye-tracking sensors. Kalɛido's architecture comprises four major components: (1) *context processing core*, which extracts scene semantics from keyframes of dynamic visual feed; (2) *configuration manager*, which automatically configures the parameters of the DP guarantee based on scene semantics and user preferences; (3) *noisy gaze generator* which generates noisy gaze streams; and (4) *noisy gaze processor*, which performs local post-processing on the noisy gaze streams. The Kalɛido plugin leverages off-the-shelf APIs and computing blocks, providing backward compatibility across a broad spectrum of applications and platforms.

We conduct a user study and trace-based analysis to evaluate Kalɛido. To understand perceived utility, we investigate the user experience of a real-time eye-tracking game with Kalɛido. The quantitative and qualitative feedback indicates a minor impact on users' game performance and satisfaction. The users show a high incentive to adopt Kalɛido and its control knob for eye-tracking privacy. Furthermore, we validate that Kalɛido can successfully thwart various adversarial analytics, aiming to identify unique traits from users' eye gazes. Even with modest privacy levels, Kalɛido can drive the attacker's accuracy close to random baselines.

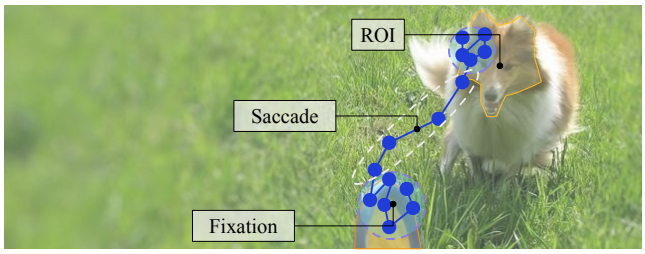

Figure 2: Example of fixations, saccades, and ROIs in a scene [52], where the blue dots represent individual gazes and purple (grey) dashed circles represent fixations (saccades).

2 Background on Eye Tracking

2.1 Properties of Eye Gaze

Eye gaze data, commonly represented as a stream of gaze positions projected onto a visual scene, reflects how people explore and process the visual content. Typically, eye gaze data is abstracted as a *scanpath*, which captures the characteristics of the user's visual attention [68]. A scanpath is a time sequence of *fixations* that are separated by *saccades* [8, 82]. Fixations represent clusters of gazes concentrated around specific regions in the scene (such as an object). Saccades denote gazes traveling rapidly from one fixation to another. A region in the scene space that attracts human attention [58] is referred to as a *region of interest* (ROI). Figure 2 illustrates fixations, saccades, and ROIs in a scene.

2.2 Eye-Tracking Platform

Two of the most popular techniques for acquiring real-time eye gaze [56] are: *vision-based tracking* and *infrared pupil-corneal reflection tracking*. The former estimates gaze positions from the captured images of the eyes; the latter projects infrared light onto the eyes and estimates the point of gaze from the pupil and corneal reflections. The raw measurement data is represented as a stream of tuples $\langle x, y, t \rangle$, where x and y represent the 2D coordinates of its location on the visual scene (corresponding to a pixel of the image), and t is the associated timestamp [47, 83, 86].

Eye-tracking platforms [37, 45] incorporate eye-tracking with development engines, such as Unity. The platform ex-

poses eye gaze streams to user applications through predefined APIs. An application *session* is the duration of user interaction with the platform to perform a task, such as playing a game or browsing a document. Each session is a series of *scenes* where the visual content remains relatively unchanged (e.g., part of the same panoramic view).

Each application defines its interaction semantics based on the eye gaze streams. Examples include eye gaze-based input and selection [84], active event triggering by eye gaze gestures [51], automatic scene switching during browsing [46], foveated rendering [6, 67], and virtual social interaction using digital avatars [80].

2.3 Privacy Threats

Eye gaze patterns inherently reflect human traits and carry sensitive information about the user. While the applications would primarily process eye gaze streams for user interaction purposes, accumulating the data over multiple sessions can result in privacy threats. Below, we discuss some examples of possible psychological and physiological inferences that can be drawn from eye gaze streams.

Absolute gaze distribution on a scene. The spatial distribution of absolute gaze positions on a scene can reveal insights about the individual's cognitive process of exploring specific visual content. Fixations and saccades within and between ROIs reflect how an individual's attention moves within a scene – revealing cues about one's interest. For example, gaze patterns on merchandise can enable precision marketing and personalized recommendations in consumer research [16, 24, 92]. Other researchers have attributed individuals' fixation patterns to their psychological state, such as lying about recognizing a face [60, 78]. Further, individuals with different physiological and cultural backgrounds demonstrate distinguishing characteristics depending on the ROI features such as color, texture, and semantics [3, 70].

Aggregate statistics on gaze distribution over time. The statistical characteristics or features of scanpaths computed over a period of time, such as fixation duration/rate and saccade speed/acceleration, can reveal sensitive information about an individual. For example, the length of saccades can help in categorizing fixations into different functional groups, including "locating," "guiding," "directing," and "checking," which reveal one's behavioral traits while performing daily tasks, such as interpersonal communication [49, 75, 77]. Diseases such as autism spectrum disorder [30] and Alzheimer's [41] can also be diagnosed from fixation features. Additionally, fixation and saccade features can be utilized as biometrics for user identification and authentication [23, 33] because of their uniqueness to individuals. These features can also reveal information about a user's physiological conditions, such as vision correction conditions [63].

3 Related Work

In this section, we provide a summary of the related work. One line of work proposes "recognizer" systems that process a sensor stream, such as a video, to "recognize" predefined objects or features [38, 69, 73]. The principle underlying these systems is to send only abstract features from the data stream (possibly after obfuscation) to the untrusted applications in place of the raw stream. However, this approach suffers from a set of shortcomings when applied in the context of real-time eye tracking. First, APIs of current user applications expect, as inputs, raw eye gaze streams directly or basic gaze events such as fixations. Second, this approach does not provide a formal privacy guarantee and cannot defend against attacks that consume only coarse-grained measurements (that can be computed from the features) [53]. Last, such systems introduce complications for permission control for both users and application developers.

Another line of work uses adversarial machine learning-based approaches to protect the raw eye gaze data [29]. However, such techniques operate on predetermined data streams and require training. Hence, these solutions are not practically feasible for real-time interactions. Additionally, they do not offer any formal privacy guarantee. In another work, Bozkir et al. [13] use randomized encoding to privately train an SVR model for gaze estimation. However, this method would require significant changes, such as communication with a third-party server, to existing eye-tracking ecosystems.

Differential privacy has been proposed in the context of eye tracking [12, 53, 79]. However, the major problem with the existing works is that they release noisy high-level features, such as heatmap [53] and ratio of saccades [12, 79]. Moreover, their workflow involves collecting the dataset of eye gaze streams from a group of users and then performing noisy feature extraction from it – the data release cannot be performed in real-time. Also, the computation of the sensitivity [21] of the features in two of the works [12, 79] is dependent on the dataset, leading to additional privacy leakage [64]. Further, Bozkir et al. [12] adopt the central differential privacy setting that requires the presence of a trusted data aggregator, an infeasible proposition for most eye-tracking applications.

Thus, the solutions above are not directly comparable to Kalɛido, aiming to provide a formal privacy guarantee for raw gaze streams in real-time interactions.

4 Privacy Model

As discussed in Section 2.3, we observe that the privacy threats to eye-tracking data arise either from the analysis of the absolute spatial distribution or the aggregate statistics of gaze positions over time. Thus, the spatial information of the gaze positions is the primary source of sensitive information. Hence, in Kalɛido, we choose to provide our formal guarantee (Definition 4.5) on the spatial information of the gaze

positions. In what follows, we start with some background on differential privacy, followed by the privacy definition for Kalɛido and its implications.

4.1 Differential Privacy Preliminaries

For Kalɛido's formal privacy guarantee, we leverage two variants of differential privacy: geo-indistinguishability [5] and w-event differential privacy [42].

Geo-indistinguishability. Geo-indistinguishability is a specialization of differential privacy that provides privacy guarantees for geographical information in 2D space. It is formally defined as follows:

Definition 4.1 ((ε, r)-geo-indistinguishability). A mechanism $\mathcal{M} : \mathcal{X} \mapsto \mathcal{Z}$ is defined to be (ε, r) - geo-indistinguishable iff for all pairs of inputs $(x, x') \in \mathcal{X} \times \mathcal{X}$ such that $d(x, x') \leq r$,

$$\forall S \subset \mathcal{Z}, Pr[\mathcal{M}(x) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(x') \in S] \quad (1)$$

where $d(\cdot, \cdot)$ denotes the Euclidean metric.

We refer to the pair (x, x') in the above definition as the r-Euclidean neighboring. Intuitively, the above definition protects all pairs of r-Euclidean neighbors[1].

***w*-event differential privacy.** As discussed above, eye gaze data in real-world interaction interfaces is obtained in the form of streaming data. Hence, we also use a variant of the *w-event differential privacy* guarantee [42], which is defined in the context streaming data. In this context, the user's behavior breaks into a set of "events," corresponding to data updates in the stream due to user actions. Intuitively, this privacy guarantee protects all event sequences of length w in a stream.

Let S be a stream of an infinite tuple $S = (D_1, D_2, \cdots)$ where every data point D_i at time stamp i is a database with d columns and arbitrary rows (each row corresponds to an unique user). Let S_t denote a stream prefix of S up till time stamp t, $S_t = (D_1, D_2, \cdots, D_t)$, and $S_t[i], i \in [t]$ denote the i-th element of S_t, D_i.

Definition 4.2 (w-Neighboring Stream Prefixes [42])**.** Two stream prefixes S_t, S'_t are defined to be w-neighboring, if

- for each $S_t[i], S'_t[i]$ such that $i \in [t]$ and $D_i = S_t[i] \neq S'_t[i] = D'_i$ it holds that, D'_i can be obtained from D_i by adding or removing a single row, and
- for each $S_t[i_1], S_t[i_2], S'_t[i_1], S'_t[i_2]$ with $i_1 < i_2, S_t[i_1] \neq S'_t[i_1]$ and $S_t[i_2] \neq S'_t[i2]$, it holds that $i_2 - i_1 + 1 \leq w$.

Using the above definition, w-event differential privacy is defined formally as follows:

[1] We introduce some notational change from the original work [5]. Our privacy parameter ε is equivalent to the term $\varepsilon \cdot d(x, x')$ from the original definition (see Section 4.3.3 for details). We adopt this change to improve readability, which does not affect the semantics of the definition.

Definition 4.3 (w-Event Differential Privacy [42])**.** A mechanism $\mathcal{M} : \mathcal{S} \mapsto \mathcal{C}$, where $\mathcal{S}$ is the domain of all stream prefixes, satisfies w-event differential privacy if for all pairs of w-neighboring stream prefixes $\{S_t, S'_t\} \in \mathcal{S} \times \mathcal{S}$, we have

$$\forall O \subseteq \mathcal{C}, \forall t, Pr[\mathcal{M}(S_t) = O] \leq e^{\varepsilon} Pr[\mathcal{M}(S'_t) = O] \quad (2)$$

Note that w refers to the count of distinct "events" in a stream in the above definition. In our definition, w refers to the duration of the event window (as in Definition 4.5).

4.2 Privacy Definitions in Kalɛido

We now discuss how the aforementioned privacy definitions are used for protecting eye gaze streams. We observe that in a 2D scene, the eye gaze data is analogous to geographical information as modeled in the geo-indistinguishability framework [5]. Specifically, we can use the Euclidean distance as a metric for gaze data points. Keeping this in mind, we model the eye gaze time series as a stream of an infinite tuple $S^g = (\langle g_1, t_1 \rangle, \langle g_2, t_2 \rangle, \cdots)$, where each data point $g_i = \langle x_i, y_i \rangle$ gives the corresponding 2D gaze position, and t_i is the associated timestamp. Let S^g_k denote a stream prefix of S^g of length k, i.e., $S^g_k = (\langle g_1, t_1 \rangle, \langle g_2, t_2 \rangle, \cdots, \langle g_k, t_k \rangle)$. Using this model of eye gaze positions, we present our notion of (w, r)-neighboring for gaze stream prefixes.

Definition 4.4 ((w, r)-neighboring gaze stream prefixes). Two gaze stream prefixes $S^g_k = (\langle g_1, t_1 \rangle, \cdots, \langle g_k, t_k \rangle), S^g_{k'} = (\langle g'_1, t'_1 \rangle, \cdots, \langle g'_k, t'_k \rangle)$ are defined to be (w, r)-neighboring, if

- the timestamps of their elements are pairwise identical: for $i \in [k]$, we have $t_i = t'_i$;
- the gaze positions of their elements are r-Euclidean neighboring: for each g_i, g'_i such that $i \in [k]$, it holds that $d(g_i, g'_i) \leq r$; and
- all of the neighboring gaze points can fit in a window of time duration at most w: for each $g_{i_1}, g_{i_2}, g'_{i_1}, g'_{i_2}$, with $i_1 < i_2, g_{i_1} \neq g'_{i_1}$ and $g_{i_2} \neq g'_{i_2}$, it holds that $t_{i_2} - t_{i_1} \leq w$.

Leveraging the notion of neighboring gaze stream prefixes, we present our formal privacy definition as follows. This definition is a variant of the *w-event differential privacy* guarantee [42].

Definition 4.5 ((ε, w, r)-differential privacy for gaze stream prefixes). A mechanism $\mathcal{M} : \mathcal{S}^g \mapsto \mathcal{C}^g$, where $\mathcal{S}^g$ is the domain of all stream prefixes, satisfies (ε, w, r)-differential privacy if for all pairs of (w, r)-neighboring gaze stream prefixes $\{S^g_k, S^{g'}_k\} \in \mathcal{S}^g \times \mathcal{S}^g$, we have

$$\forall O \in \mathcal{C}^g, \forall k, Pr[\mathcal{M}(S^g_k) = O] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(S^{g'}_k) = O] \quad (3)$$

Based on this definition, we present a result that enables a (ε, w, r)-differentially private mechanism to allocate a privacy budget of ε for any sliding window of duration w in a given stream prefix.

Theorem 1. *Let $\mathcal{M} : \mathcal{S}^g \mapsto \mathcal{C}^g$ be a mechanism that takes as input a gaze stream prefix $S_k^g = (\langle g_1, t_1\rangle, \cdots, \langle g_k, t_k\rangle)$ and outputs a transcript $O = (o_1, \cdots, o_k) \in \mathcal{C}$. Additionally, let $\mathcal{M}$ be decomposed into k mechanisms $\mathcal{M}_1, \cdots, \mathcal{M}_k$ such that $\mathcal{M}_i(g_i) = o_i$, and each $\mathcal{M}_i$ generates independent randomness while achieving (ε_i, r)-geo-indistinguishability. Let $l \in [1, i-1]$ represent an index such that $(t_i - t_l) = w$. Then, $\mathcal{M}$ satisfies (ε, w, r)-differential privacy if*

$$\forall i \in [k], \sum_{j=l}^{i} \varepsilon_j \leq \varepsilon \tag{4}$$

The proof of Theorem 1 follows directly from the proof of Theorem 3 in Kellaris et al. [42].

Discussion of privacy semantics. The idea behind (ε', r)-geo-indistinguishability (Definition 4.1), in the context of eye-tracking data, is that given a gaze position g, all points within a circle of radius r centered at g (i.e., all r-neighbors of g) would be "indistinguishable" to an adversary who has access to the corresponding "noisy" location. Thus, this privacy guarantee provides a cloaking region of radius r around g. (ε, w, r)-differential privacy (Definition 4.5) extends this guarantee to gaze stream prefixes. Specifically, an adversary cannot distinguish[2] between any two gaze stream prefixes, which (1) differ in gaze positions that are within a distance of r from each other, and (2) all such differing pairs occur within a window of duration w.

Additionally, from Theorem 1, we observe that a (ε, w, r)-differentially private mechanism can achieve two goals: for every subsequence of duration w in the gaze stream S_k^g, it (1) allocates up to ε privacy budget, and (2) takes budget allocation decisions considering the *entirety* of the subsequence. Thus, this privacy definition *protects the spatial distribution of any gaze trajectory that is formed over any window of a duration w*.

Further, we define and prove another result, which shows that the privacy guarantee degrades gracefully if the r-Euclidean neighbors in both stream prefixes are separated by more than w duration. The proof of the following theorem is in Appendix A.1.

Theorem 2 (Composition over multiple windows theorem). *Let $\mathcal{M} : \mathcal{S}^g \mapsto \mathcal{C}^g$ be a mechanism that takes as input a gaze stream prefix $S_k^g = (\langle g_1, t_1\rangle, \cdots, \langle g_k, t_k\rangle)$, and outputs a transcript $O = (o_1, \cdots, o_k) \in \mathcal{C}$. Additionally, let $\mathcal{M}$ be decomposed into k mechanisms $\mathcal{M}_1, \cdots, \mathcal{M}_k$ such that $\mathcal{M}_i(g_i) = o_i$, and each $\mathcal{M}_i$ generates independent randomness while achieving (ε_i, r)-geo-indistinguishability. Then for two stream prefixes S_k^g and $S_k^{g'}$, such that:*

- *for all $i \in [k]$, $t_i = t_i'$;*
- *for each g_i, g_i' such that $i \in [k]$ and $g_i \neq g_i'$ it holds that $d(g_i, g_i') \leq r$, i.e., $(g_i, g_{i'})$ are r-Euclidean neighboring; and*
- *for each $g_{i_1}, g_{i_2}, g_{i_1}', g_{i_2}'$, with $i_1 < i_2, g_{i_1} \neq g_{i_1}'$ and $g_{i_2} \neq g_{i_2}'$, it holds that $t_{i_2} - t_{i_1} \leq m \cdot w, m \in \mathbb{N}$;*

we have

$$\forall O \in \mathcal{C}^g, \forall k, Pr[\mathcal{M}(S_k^g) = O] \leq e^{m\cdot\varepsilon} \cdot Pr[\mathcal{M}(S_k^{g'}) = O]. \tag{5}$$

Another important result for differential privacy is that any post-processing computation performed on the noisy output does not cause any privacy loss. Thus, once Kalεido releases the noisy gaze streams, all subsequent analyses by the adversary enjoy the same privacy guarantee.

Theorem 3 (Post-processing). *Let the randomized mechanism $\mathcal{M} : \mathcal{S}^g \mapsto C^g$ satisfy (ε, w, r)-differentially privacy. Let $f : C^g \mapsto R$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : \mathcal{S}^g \mapsto R$ is (ε, w, r)- differential private.*

4.3 Privacy Implications of Kalεido

In the following, we discuss the implications of the formal privacy guarantee of Kalεido (Definition 4.5).

4.3.1 Choice of Parameters

The aforementioned privacy guarantee involves three parameters – the privacy budget, the window length, and the radius of location indistinguishability:

Privacy budget ε. ε captures the privacy requirements of the user which can be set at the user's discretion [2, 35, 50].

Window length w. As explained above, the proposed privacy definition protects the spatial distribution of a gaze trajectory that is formed within any window of duration w. In a typical eye-tracking setting, gaze trajectories are formed over individual visual scenes. Thus, a good choice for w could be average scene lengths in a visual feed. Over the whole session, which spans multiple windows, the resulting privacy guarantee degrades gracefully (by Theorem 2).

Radius of location indistinguishability r. Recall that eye gaze streams be abstracted to a series of *fixations* and *saccades* within and between ROIs. Hence, we propose the following two choices for the value of parameter r:

- **Intra-region radius r_{intra}.** This measure captures the radius of a single ROI (approximated by a circular area) and is catered to protect gaze data positions corresponding to fixations.
- **Inter-region radius r_{inter}.** This measures the distance between a pair of ROIs (approximated by circular areas) and protects gaze positions corresponding to inter-ROI saccades.

[2] with probability higher than what is allowed by the privacy parameter ε

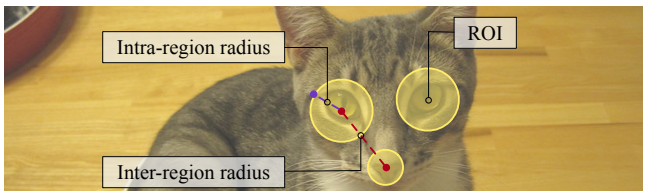

Figure 3: Illustration of the two choices for the radius of location indistinguishability parameter [52].

The two radii are illustrated in Figure 3. As a general rule, the larger the value of r greater is the privacy enjoyed (at the cost of lower utility). Note that we assume that the visual feeds are publicly available (see Section 5.1).

Thus, in a nutshell, Kalɛido's privacy guarantee ensures that an adversary cannot learn about the distinguishing features of a user's spatial distribution. Specifically, if r is chosen as r_{intra}, then an adversary cannot distinguish[3] between two users gazing at the same ROI, within any window of length w. Similarly, if r is chosen as r_{inter}, then the adversary cannot distinguish two users such that (1) user 1's gaze moves from ROI_1 to ROI_2, and (2) user 2's gaze moves from ROI_1 to ROI_3, within any window of length w.

4.3.2 Discussion on Temporal Information of Eye Gaze

Kalɛido's formal privacy guarantee focuses solely on the location information of eye gaze streams. However, as discussed in Section 2.3, some privacy attacks utilize both location and temporal information (aggregate statistics) of gaze streams. In these cases, the location information contained in the aggregate statistics constructed over noisy gaze positions (Definition 4.5) will also be noisy (Theorem 3) – thereby reducing the efficacy of the attacks. Our evaluation results in Section 7.3 provide *empirical evidence* for the above: Kalɛido is able to protect against analyses that exploit such spatio-temporal statistics. Additionally, a formal guarantee on the temporal information would require interfering with the timeliness of the release of gaze data points (noisy or otherwise), which might adversely affect the utility [27]. Nevertheless, Section 8 discusses a possible extension of Kalɛido for providing a formal guarantee on the temporal information of eye gaze streams.

4.3.3 Contributions of Kalɛido's Privacy Definition

Here, we discuss the contributions of Kalɛido's formal privacy definition (Definition 4.5).

First, this definition combines the privacy semantics from two distinct contexts: absolute location data and the streaming of event data. Specifically, Definition 4.5 provides (ε, r)-geo-indistinguishability guarantee for every gaze position within a window of duration w in a gaze stream.

[3] with probability higher than what is allowed by privacy parameter ε

Second, there are certain semantical differences in the use of location perturbation techniques (such as (ε, r)-geo-indistinguishability guarantee) in the contexts of geographical information and eye gaze data. Typically, ROIs (also known as points of interest) for geographical information include physical units such as restaurants, shopping malls, or schools. On the other hand, ROIs in the eye-tracking context are characterized by visual stimuli such as the scene's color and texture. Consider a case where only a single ROI is located within a circle of radius r centered at the true user location (or eye gaze position). In the case of geographical information, the adversary can conclude that the user is visiting the particular ROI. Thus, this completely violates the user's location privacy. However, the above-described scenario corresponds to a fixation event (r_{intra}) in the context of eye-tracking, and eye movements, even within a single ROI are a rich source of sensitive information [70] (as discussed in Section 2.3). Thus, even if the adversary learns the ROI's identity, the perturbation still provides meaningful privacy protection.

Additionally, for the standard geo-indistinguishability guarantee [5], the privacy guarantee enjoyed is parameterized by the multiplicative term $\varepsilon \cdot d(x,x')$, i.e., the privacy guarantee degrades with the distance between the pair of points $\{x,x'\}$. This makes the task of choosing the value of ε tricky for geographical data [65]. The reason behind this is that, for any given value of ε, if the distance $d(x,x')$ becomes too large, then the subsequent privacy guarantee provided ceases to be semantically useful. Hence, deciding on the size of the cloaking region ($d(x,x')$), such that any two points within the region are sufficiently protected, is difficult for geographical data in practice. However, in the context of eye gaze data, sensitive information is captured in the form of fixations and saccades. Thus here, we are primarily concerned about protecting pairs of gaze positions that are bounded by a specific distance (r_{intra} and r_{inter} as discussed in Section 4.3.1). Hence, our formulation (Definition 4.1) explicitly parameterizes the size of the cloaking region, r, and its privacy parameter, ε, is equivalent to the term $\varepsilon \cdot d(x,x')$ (equivalently, $\varepsilon \cdot r$ where $d(x,x') \leq r$) from the original definition. This ensures that all pairs of gaze positions within a distance of r from each other enjoy a privacy guarantee of at least ε, thereby mitigating the aforementioned problem.

5 Kalɛido System Design

We introduce the system design of Kalɛido, starting with the threat model followed by design goals. Next, we present the architectural overview followed by detailed descriptions.

5.1 Threat Model

The software stack of real-time eye tracking comprises two major parties: the *eye-tracking platform* and the *third-party application* (Section 2.2). In our threat model, we assume the

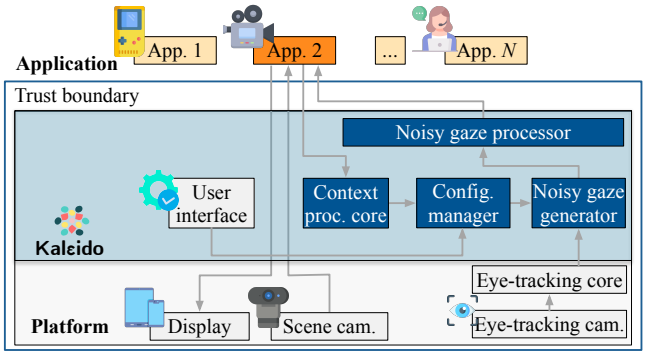

Figure 4: Architectural overview of Kalɛido.

eye-tracking platform to be trusted (a common assumption in prior works [38, 73]) and consider the untrusted third-party application to be the adversary. The application can perform analysis on the gaze streams to learn sensitive information about the user (as described in Section 2.3). Additionally, we assume that the visual feeds (image or video scenes users look at) are publicly available. This assumption holds in most practical eye-tracking applications such as movies and VR games. Thus, attackers (untrusted third-party applications) can access visual feeds and noisy gazes (output of Kalɛido), but not raw gazes.

5.2 Kalɛido Design Principles

Kalɛido relies on the following three design principles.

- **Seamless integration with existing eye-tracking interfaces.** Kalɛido seamlessly integrates with the current eye-tracking ecosystem. Specifically, it interacts with the different components of the eye-tracking framework using their existing interfaces.
- **Real-time system.** Kalɛido is capable of generating noisy gaze streams (satisfying Definition 4.5) in real-time that is suitable for interactive eye-tracking interfaces.
- **Automatic privacy parameter configuration.** Kalɛido automatically configures the privacy parameters, namely w and r, based on the properties of the visual feed.

5.3 Architectural Overview

Figure 4 depicts the high-level architecture of the eye-tracking framework with Kalɛido. It comprises three layers: the eye-tracking platform, Kalɛido, and the applications. Kalɛido is an intermediary layer in this stack that defines the trust boundary.
Eye-tracking platform. The eye-tracking platform includes a display, the eye-tracking camera, the eye-tracking core, and potentially a scene camera. Users consume the visual feed via the platform-specific display, generated either entirely digitally (VR platforms) or from the scene camera (augmented reality platforms). The eye-tracking camera captures eye image frames, from which the eye-tracking core generates raw gaze streams.
Kalɛido. Kalɛido processes the raw gaze stream obtained from the eye-tracking platform in a privacy-preserving manner. Based on the information from the visual feed and user-specified guidelines, it automatically configures the parameters required for the privacy guarantee of Definition 4.5. It then perturbs the raw gaze stream, sanitizes it, and feeds it to the applications. Section 5.4 elaborates the design of Kalɛido.
Applications. The applications use eye gaze streams for their functionalities. They receive gaze streams (albeit noisy) from Kalɛido using the original APIs. Therefore, they need not be modified in any way to be compatible with Kalɛido.

5.4 Kalɛido System Modules

Kalɛido views user interaction with the eye-tracking platform as a set of sessions with dynamic scenes. We elaborate on Kalɛido's modules and how it achieves its privacy guarantee.

5.4.1 Context Processing Core

The context processing module extracts the size and locations of the ROIs from individual frames (still images of a scene) of the visual feed. Kalɛido adopts off-the-shelf region and object detectors [54, 90] for ROI extraction. However, these detectors are computationally heavy, and continuously running them results in a high computational overhead that might hinder real-time operation. Kalɛido solves this challenge by incorporating a threshold-based keyframe detector. As frames remain relatively consistent over short periods, Kalɛido invokes the object detector only at the instances of a scene change.

5.4.2 Configuration Manager

The configuration manager module automatically configures the privacy parameters to satisfy the privacy guarantee of Definition 4.5. It accepts as inputs the processed scene information from the context processing core and the user's privacy preferences, and configures the parameters as follows:
Privacy budget ε. For setting the value of ε, Kalɛido provides the users with a privacy scale ranging from no privacy (releases raw gaze streams) to high privacy (releases noisy gaze streams). Users can adjust this knob during an active session through the configuration manager's UI, and Kalɛido interpolates the corresponding value of ε in the background.
Window length *w*. As discussed in Section 4.2, w is set according to scene lengths. Each scene corresponds to a period during which the visual content, e.g., a video, remains relatively static as defined in Section 2.2. The configuration manager can compute this value either on the fly from the context processing core's scene detectors or offline profiling and video metadata. Small values of w (of the order of a few

seconds) usually work well as most real-world interactive scenes are rapidly changing and spatially heterogeneous.

Radius of location indistinguishability r. The configuration manager module sets the value of r based on either r_{intra} or r_{inter} according to the user's preference. It uses the set of detected ROIs for each scene to compute r as follows. Let $\{\text{ROI}_i\}, i \in [N]$, denote the set of ROIs for a given scene where N is the total number of ROIs. Let a tuple $\langle x_i, y_i, d_i^w, d_i^h \rangle$ represent the output of the object (or region) detector, where (x_i, y_i) is the position of a reference point (for example, the centroid) of the bounding box of ROI_i, and (d_i^w, d_i^h) is its width and height, respectively. Thus, ROI_i can be approximated by a circular area centered at (x_i, y_i) and its radius that is computed from the diagonal of the bounding box:

$$r_{intra}^i = 0.5 \times \sqrt{d_i^{w2} + d_i^{h2}} \tag{6}$$

For any pair of regions of interest (approximated by circular areas) ROI_i and ROI_j $i, j \in [N], i \neq j$, we have

$$r_{inter}^{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{7}$$

After computing the radii of all ROIs, the configuration manager has two default modes for r: r_{small}, which is the median of $\{r_{intra}^i\}$, and r_{large}, which is the median of $\{r_{inter}^{i,j}\}$.

5.4.3 Noisy Gaze Generator

The noisy gaze generator module perturbs the raw gaze streams generated by the eye-tracking core. This perturbation entails allocating a privacy budget for each gaze position and then generating its corresponding noisy position in a (ε, w, r)-differential private manner (Definition 4.5).

The raw measurement frequency is very high ($\sim$ 120 Hz), especially for interactive settings. Even for low values of w, the number of individual gaze positions could be relatively high. Therefore, naive budget allocation strategies such as uniform allocation or fixed-rate sampling are likely to provide poor utility [42]. To this end, we use an adaptive budget allocation strategy that considers the dynamics of the human eye gaze. We observe that the *human gaze is relatively localized during fixations*. Based on this observation, we identify two optimizations for the budget allocation strategy. Let g' denote the last published noisy gaze position.

- Gaze data points generated in quick succession of g' can be skipped over.
- The last released g' can be used as a proxy for data points that lie in its spatial proximity.

These optimizations are akin to (1) performing a simple fixation detection (in a privacy-preserving manner) based on the spatio-temporal gaze data points, and (2) publishing a noisy gaze position only when a new fixation is detected. This requires the privacy budget to be distributed between two tasks: testing the proximity of the gaze positions and the publication of noisy gaze positions. The temporal check (for skipping data points) consumes no privacy budget since our formal guarantee (Definition 4.5) applies to spatial information only.

Kalɛido uses an *adaptive budget allocation* strategy that (1) starts with a total privacy budget ε for every window of duration w, (2) allocates no budget for the gaze data points to be skipped over, (3) allocates a fixed budget for testing all other data points, (4) distributes publication budget in an exponentially decreasing manner to the data points which have been decided to publish, and (5) recycles the budget spent in timestamps falling outside the active window. Algorithm 1, based on the BD algorithm [42], outlines the above method; similar ideas have also been presented in the context of location sequences [18].

Adaptive budget allocation. The algorithm proceeds in three stages. In the first stage (Steps 1–4), every gaze position that is generated up to duration t_{skip} after i_{test} is skipped, where i_{test} denotes the timestamp of the last tested gaze position. A good choice for t_{skip} can be the minimum duration of fixations $\approx$ 50 ms [48]. Thus, this stage reuses the last published noisy gaze $(g'_{i_{pub}})$ and consumes no privacy budget (Step 3).

The second stage (Steps 5–11) is the testing phase, where all the "not-skipped" gaze positions are tested for their proximity to $g'_{i_{pub}}$. Specifically, it checks whether the current gaze position g_i (not-skipped) is within a certain noisy threshold $(l_{thresh} + \eta)$[4] from $g'_{i_{pub}}$ (Steps 6–8). In case this is satisfied, the algorithm again reuses $g'_{i_{pub}}$. The total privacy budget allocated for testing for any window duration of w is ε/h. Each individual test consumes a budget $\varepsilon_{test} = \varepsilon/(h \cdot n_{test})$, where n_{test} is the number of gaze positions to be tested per window, and h is a parameter with a value greater than 2. The first two stages of the algorithm can be interpreted as a simple $(\varepsilon/h, w, r)$-differentially private fixation detection scheme.

Finally, in the third stage (Steps 12–16), the algorithm publishes a noisy gaze position corresponding to g_i only if it is sufficiently distant from $g'_{i_{pub}}$. For this, it computes the remaining budget for the active window (Step 13) as follows

$$\varepsilon_{rem} = \underbrace{\varepsilon}_{\substack{\text{Total privacy} \\ \text{budget for} \\ \text{each window}}} - \underbrace{\varepsilon/h}_{\substack{\text{Budget consumed} \\ \text{for testing in} \\ \text{the active window}}} - \underbrace{\sum_{k=i-n_{raw}+1}^{i-1} \varepsilon_k^{pub}}_{\substack{\text{Budget consumed} \\ \text{for noisy publication in} \\ \text{the active window}}}$$

Next, the algorithm assigns half of it ($\varepsilon_{rem}/2$) for the noisy publication (Step 14). Thus, the publication budget is allocated in an exponentially decreasing manner. The rationale behind this is that investing a high budget (i.e., injecting low noise) in the current measurement g'_i would result in better approximation (test and reuse) for the future ones. Additionally, note that ε_{rem} considers the budget consumed only in the

[4] The value of l_{thresh} impacts utility and is chosen empirically depending on r.

Algorithm 1 Adaptive Budget Allocation

Parameters: w - Time duration of a single window in seconds (s), ε - Total privacy budget per window of size w
p_{raw} - Rate of raw gaze data generation in $samples/s$, l_{thresh} - Threshold for distance
t_{skip} - Time duration for skipping after every gaze data point testing, r - Radius of indistinguishability
h - Ratio of privacy budget used for testing

Initialization:

$n_{raw} = w \cdot p_{raw}$ ▷ Number of raw gaze data points generated in a single window
$n_{test} = \lceil w/t_{skip} \rceil$ ▷ Number of raw gaze data points tested in a single window
$\varepsilon_{test} = \varepsilon/(h \cdot n_{test})$ ▷ Privacy budget allocated for every test in a single window
$i_{test} = \varnothing$ ▷ Timestamp of the last tested gaze position
$i_{pub} = \varnothing$ ▷ Timestamp of the last published noisy gaze position

Input: g_i - True gaze position for timestamp i
$g'_{i_{pub}}$ - Output for the last timestamp, initialized to $\varnothing$ when $i_{pub} = \varnothing$
$\{\varepsilon^{pub}_{i-n_{raw}+1}, \cdots, \varepsilon^{pub}_{i-1}\}$ - Privacy budget consumed for publication in last n_{raw} timestamps, initialized to 0 if $i < n_{raw}$

Output: g'_i - Noisy gaze position released for timestamp i
ε^{pub}_i - Privacy budget consumed in publications

Stage I: Check whether to skip or test the gaze data point ▷ Fixation detection based on timestamp of data

1: **if** ($i_{test} \neq \varnothing$ and $time(i) - time(i_{test}) < t_{skip}$) **then**
2: $g'_i = g'_{i_{pub}}$ ▷ Reuse last published gaze position
3: $\varepsilon^{pub}_i = 0$
4: **Return** $\{g'_i, \varepsilon_i\}$

Stage II: Test whether current gaze data point should be published ▷ Fixation detection based on location of data

5: $i_{test} = i$
6: $l_{dis} = d(g_i, g'_{i_{pub}})$ ▷ Euclidean distance between last published gaze position and current gaze position with $d(\cdot, \varnothing) = \varnothing$
7: $\eta \sim \mathsf{Lap}(1/\varepsilon_{test})$ ▷ $\mathsf{Lap}(\cdot)$ denotes the Laplace distribution
8: **if** ($l_{dis} \neq \varnothing$ and $l_{dis} \leq l_{thresh} + \eta$) **then** ▷ Test whether current gaze position is in the proximity of the last published gaze position
9: $g'_i = g'_{i_{pub}}$
10: $\varepsilon^{pub}_i = 0$
11: **Return** $\{g'_i, \varepsilon^{pub}_i\}$

Stage III: Publish noisy gaze point

12: $i_{pub} = i$
13: $\varepsilon_{rem} = \varepsilon - \varepsilon/h - \sum_{k=i-n_{raw}+1}^{i-1} \varepsilon^{pub}_k$ ▷ Remaining privacy budget for the active window
14: $\varepsilon^{pub}_i = \varepsilon_{rem}/2$
15: $g'_i = \mathsf{PlanarLap}(g_i, \varepsilon_i/r)$ ▷ $\mathsf{PlanarLap}(\cdot)$ is a geo-indistinguishable mechanism from [5]
16: **Return** $\{g'_i, \varepsilon^{pub}_i\}$

active window $[i - n_{raw} + 1, i]$. Thus, the publication budget of older timestamps (preceding the active window) is recycled for future usage. The generation of the noisy gaze position is done via the PlanarLap() mechanism (Step 15), which satisfies geo-indistinguishability [5] (with the notational difference of using ε^{pub}_i/r as the privacy budget).

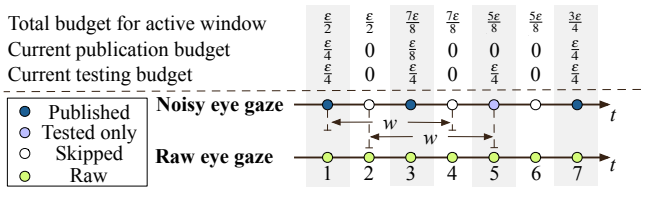

Figure 5: Illustrative example of Kalɛido's budget allocation ($n_{raw} = 4$, $n_{test} = 2$, $h = 2$).

Illustrative example. Figure 5 presents an illustrative example of Algorithm 1. Here we consider $n_{raw} = 4$, $n_{test} = 2$ and $h = 2$. Hence, the budget for testing per gaze position is $\varepsilon/4$. For the first window (timestamps 1-4), the algorithm publishes at timestamps 1 and 3 and skips at timestamps 2 and 4. Hence, timestamps 1 and 3 consume budget $\varepsilon/4$ each for testing. Additionally, the publication budgets are $\varepsilon_1 = (\varepsilon/2 - 0)/2 = \varepsilon/4$, $\varepsilon_3 = (\varepsilon/2 - \varepsilon/4)/2 = \varepsilon/8$ and $\varepsilon_2 = \varepsilon_4 = 0$. Thus, the total privacy budget consumed in this window is $\varepsilon/2$ (budget for testing) $+\varepsilon/4 + \varepsilon/8 = 7\varepsilon/8 \leq \varepsilon$. For the second window (timestamps 2-5), the algorithm reuses g'_3 at timestamp 5. Hence, its total privacy budget is $\varepsilon/2 + \varepsilon/8 = 5\varepsilon/8 \leq \varepsilon$. For the third window (timestamps 3-6), the algorithm skips the gaze position at timestamp 6 and the total privacy budget is $\varepsilon/2 + \varepsilon/8 = 5\varepsilon/8 \leq \varepsilon$. A noisy gaze position is published at timestamp 7 in the fourth window (timestamp 4-7) with $\varepsilon_7 = (\varepsilon/2 - 0)/2 = \varepsilon/4$. Thus, the total privacy budget for this window is $\varepsilon/2 + \varepsilon/4 = 3\varepsilon/4 \leq \varepsilon$.

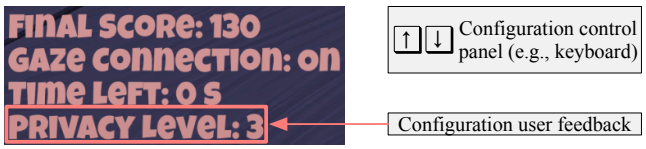

Figure 6: Basic template of Kalɛido's user interface.

Theorem 4. *Algorithm 1 satisfies* (ε, r, w)*-differential privacy.*

Proof. First, note that Stage I (Steps 1–4, Algorithm 1) do not consume any privacy budget. Next, from Fact I in [18], Stage II consumes privacy budget ε_{test} for every test. Specifically, the output of the test mechanism (Step 8) is a binary decision and hence, its sensitivity is 1. Finally, Stage III consumes budget $\varepsilon_i^{pub} = 1/2(\varepsilon - \varepsilon/h - \sum_{k=i-n_{raw}+1}^{i-1} \varepsilon_k^{pub})$ if it publishes, and 0 otherwise. Next, we prove that the total budget consumed in every window is at most ε. For this note that the total budget consumed for testing is ε/h. Hence, it suffices to show that $0 \leq \sum_{k=i-n_{raw}+1}^{i-1} \varepsilon_k^{pub} \leq \varepsilon - \varepsilon/h$ which follows directly from the proof of Theorem 4 in Kellaris et al. [42]. □

5.4.4 Noisy Gaze Processor

The noisy gaze processor takes as input the noisy gaze streams generated in real-time and performs post-processing operations on it before releasing it to the applications. This module is identical to any local post-processing unit existing in current eye-tracking systems, except for noisy inputs. Examples of such post-processing include data sanitization, such as bounding of off-screen points and data smoothing. Moreover, Kalɛido's noisy gaze processor can support local feature extraction similar to that in the "recognizer" framework [38] (Section 3). Kalɛido is thus compatible with applications with APIs expecting specific features as input, such as fixation/saccade statistics. By Theorem 3, this step does not impact the privacy guarantee of Kalɛido.

6 Implementation

We implement Kalɛido as a C# plugin in Unity [26], a cross-platform engine for developing interactive applications, such as games and mixed reality content. Unity allows developers to integrate plugins that generate visual content and communicate with peripherals, including eye trackers. In our implementation, Kalɛido acts as an intermediate protection layer between applications and the platform.

Stream acquisition. Kalɛido acquires real-time eye gaze streams from the eye-tracking core and forwards them to the noisy gaze generator. To synchronize these gaze streams, we implement the eye gaze receiver using the TCP/IP protocol, which is the most common communication channel for off-the-shelf eye-tracking cores, such as Tobii [83], GazePointer [25], and PupilLab [47].

ROI extraction. Kalɛido identifies the instances of scene change and extracts the ROIs from each scene. For deterministic visual content (such as movies), Kalɛido acquires the timing of keyframes (instances of scene changes) from either the video decoding process or keyframe properties obtained from Unity's Animation feature or content providers [88]. As for online content, Kalɛido identifies the keyframes using an on-the-fly scene change detector [94]. In particular, we implement a threshold-based real-time keyframe detector using the mean absolute frame difference method. First, Kalɛido fetches the current frame from Unity's rendering process. Next, it takes the pixel-wise difference between the current frame and the last keyframe. Kalɛido detects a new keyframe by comparing the pixel values of the binarized difference matrix against a pre-calibrated threshold. We set the default update interval of keyframe detection to 500 ms, which is the typical response latency of human attention to visual stimuli [14].

Kalɛido identifies the spatial information of ROIs for digitally rendered frames using Unity's GameObject API. For all other types of frames, Kalɛido uses YOLOv3-tiny [71], a light-weight neural network. To study the impact of YOLO on real-time performance, we make an exception and use it for digitally rendered frames as well in our user study.

User Interface. Kalɛido offers the users with an interface to adjust their privacy-utility trade-off. Users can control the privacy budget ε on-the-fly through pre-defined triggers, such as keypress, as illustrated in Figure 6. We chose a basic interface for our prototype implementation since UI design is not the focus of this work.

Table 1: Properties of eye gaze traces, with a video dataset highlighted.

Dataset	Num. of stimuli	Num. of users	Sampling rate (Hz)	Avg. duration (s)
Natural [91]	10	19	100	6.0
Web page [91]	10	22	100	16.8
Human [39]	10	60	100	3.7
VR video [4]	12	13	120	64.9

7 Evaluation

We evaluate three aspects of Kalɛido: (1) user-perceived utility, (2) real-time performance, and (3) effectiveness against spatio-temporal attacks. We perform a trace-based evaluation to measure the effectiveness of Kalɛido against attackers using four popular eye-tracking datasets. These datasets, described in Table 1, include the scenarios of natural environment, web pages, human, and virtual reality (VR) videos. In particular, our evaluation answers these questions:

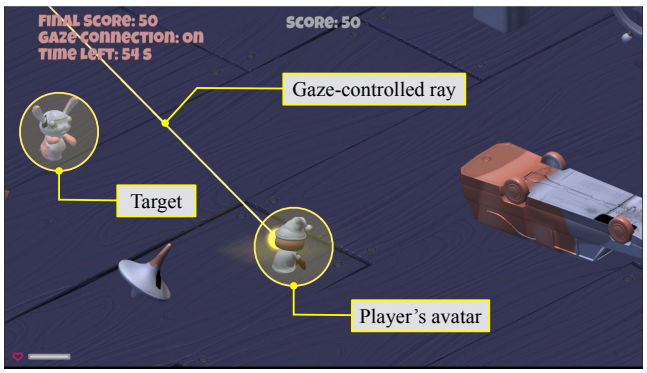

Figure 7: A scene of the "Survival Shooter" game with the player's avatar, target, and gaze-controlled ray annotated.

Q1: *How do users perceive the utility of real-time interactions with Kalɛido?*
We conduct a remote user study with 11 participants to assess the user-perceived utility while playing a real-time PC game with Kalɛido.

Q2: *How much latency overhead does Kalɛido incur?*
We measure the latency overhead of the main modules of Kalɛido to assess its real-time performance.

Q3: *Can Kalɛido thwart attacks that rely on spatio-temporal analysis of eye gaze streams?*
We perform a trace-based evaluation of Kalɛido on popular eye-tracking datasets. We investigate the effectiveness of Kalɛido's formal privacy guarantee against real-world adversarial analytics.

7.1 User Perception in Real-Time Interaction

We conducted a user study to evaluate Kalɛido's impact on utility, as perceived by the users, while playing a real-time PC game. Our objective is to understand the impact of Kalɛido on user experience at different settings of privacy. To this end, we adapted the game "Survival Shooter" [87] from Unity to be eye-tracking compatible. Participants shot targets (Zombie Bunnies) by gazing at the target position on a computer screen, as shown in Figure 7. They used the keyboard to move their digital avatar in the game. We used this PC game because of the requirement to perform the study remotely at the users' places. An in-person lab session with state-of-art eye-tracking or virtual/augmented reality was not possible during the study[5].

Setup. To accommodate a commodity PC setup, we utilize the webcam-based eye-tracking core, GazePointer [25], for detecting the participant's gaze on the screen. The remote user study design was approved by the Institutional Review Board (IRB) of our institution. We recruited 11 individuals from the mailing list of our department. The recruitment email provided no details about the study's privacy objectives and mentioned only user experience with eye-tracking games. Each remote session took 35 minutes on average, and we provided each participant with $15 worth of supplies as a token of appreciation for participating.

Limitations. We acknowledge the following limitations in our study setup resulting from the imposed lockdown. First, the demographic diversity of the participants, as well as the number of participants, might be limited. Hence, one caveat is that the confidence interval of the quantitative analysis is relatively large. Thus, we treat our presented results as a preliminary study. Second, an in-person study using state-of-the-art eye-tracking devices was not possible, which hindered our ability to study diverse scenarios, such as foveated rendering in VR and video watching. We carefully designed our study protocol to reduce the impact of the low accuracy of the webcam-based eye-tracking core; its accuracy is sensitive to posture and lighting conditions. Before starting every new session, the participants were instructed to calibrate the eye tracking using GazePointer's panel. Finally, the constraints of a remote user study also hindered us from conducting a qualitative study via in-person interviews and behavioral observation. An additional caveat is that we did not perform coded analysis for the qualitative study of user responses (via techniques such as open or axial coding [81]) of the free text.

Design. Each study session consisted of five tasks (conducted over a video call using a separate device). The first is a *pre-study survey* to collect the participant's demographic information using a Qualtrics survey. The second is the *calibration* of the webcam-based eye-tracker to map the eye gazes to the computer screen using GazePointer's calibration interface. The participants were asked to familiarize themselves with the game by practicing eye gaze-based shooting until they felt confident. The third covers the *within-subject evaluation* sessions. The fourth task tests the *privacy control knob*. The last task is the *post-session survey*.

To reduce individual differences in gaming behavior and perception, we conducted the within-subject study [17] to test four game settings: (1) *No privacy (NOPV)* — Kalɛido layer disabled; (2) *Low privacy-high utility (LPHU)* — $\varepsilon = 3$, $w = 0.5$ s, r_{small}; (3) *Medium privacy-medium utility (MPMU)* — $\varepsilon = 1.5$, $w = 1.5$ s, r_{small}; and (4) *High privacy-low utility (HPLU)* — $\varepsilon = 0.5$, $w = 2$ s, r_{large}[6]. Each setting lasted for 90 s[7], and we randomized their order for every participant. Additionally, the participants had no knowledge about the setting to which they were exposed. After the completion of each setting, we recorded: the subjective game enjoyment [57]

[5] We conducted this study during the state of Wisconsin's Safer at Home order due to the COVID-19 pandemic.

[6] These values were chosen based on a parameter sweep to represent different points along the privacy-utility spectrum (Appendix A.2.1). In the trace-based analysis of offline datasets, the root mean square error (RMSE) serves as a proxy for measuring application-specific utility loss.

[7] The interval value was chosen during calibration to balance the validity of the session and user fatigue.

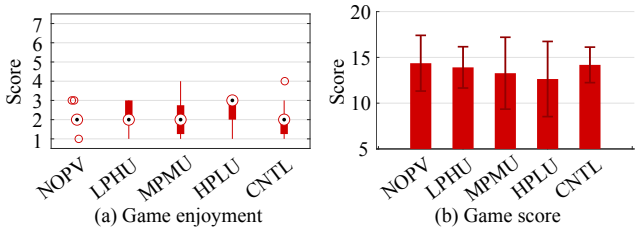

Figure 8: Scores obtained in different conditions.

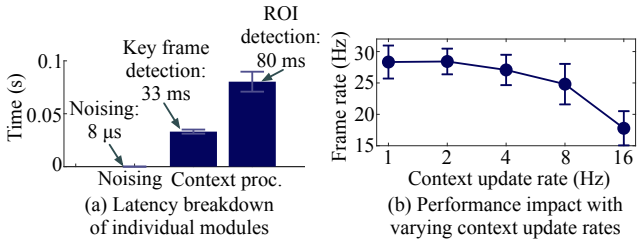

Figure 9: Performance breakdown and trend. ROI detection is the most expensive operation. The frame rate remains relatively steady even for a high context update rate of 8 Hz.

as a 7-item Likert scale, the game score, and the qualitative feedback.

After the four randomized settings, the objective of Kalɛido was revealed, and the participants were offered an adjustable knob to control the tradeoff between privacy and utility. We asked each participant to interact with the control knob; we observed how frequently they adjusted the knob and solicited qualitative feedback about their experience. This part of the study follows a *technology-probe*-based approach [36]. Our objective is to probe the participants to elicit their opinion about the missing design elements that need to be introduced.

Results. We asked the participants to report their subjective experience to evaluate the validity of our game's adaptation. To this end, we asked each participant to report their level of agreement (or disagreement) with this statement: "*You enjoyed the game in this session.*" on a 7-item Likert scale with 1 being "*Strongly Agree*" and 7 being "*Strongly Disagree*". Figure 8(a) shows that for all of the game settings, the participants enjoyed their experience – at least 82% of them reported a score of 3 or lower.

Next, we study the effect of the privacy level on the participants' game scores. Figure 8(b) shows these scores for the different settings. We observe that the game scores decrease with a stronger privacy guarantee. However, the decrease in the score is not significant from the no privacy (NOPV) setting to the low privacy (LPHU) setting (only 3.2%). Even the decrease from the NOPV setting to the high privacy (HPLU) setting is modest (12.0%). These results show that Kalɛido's noise does not adversely affect users' utility in this scenario.

The qualitative feedback that we obtained from the users aligned with our quantitative observations. Some participants were unable to distinguish between the LPHU and NOPV settings – (**P8**: "The second (NOPV) and third (LPHU) configurations are almost the same for me.") The majority of the participants found the highest privacy (HPLU) setting to be the hardest to control. Some participants had a surprisingly different view. For example, **P7** enjoyed the conditions with higher noise because it was more challenging to play.

Finally, we performed a preliminary analysis of the privacy control knob (setting: CNTL). In the last task of the study, we introduced the control knob to the participants and asked them to control the privacy level as per their desired level of utility. Figure 8(b) shows that the adjustment of the control knob does not affect the game scores. However, we find a large variation in the frequency of knob adjustment and the privacy level (ε) across the participants.

The qualitative feedback also indicated that while such a knob might be useful, they had some suggestions for improvement. For example, **P8** and **P11** proposed adding flexibility for an offline calibration of the privacy level for each application. Other participants commented that frequently adjusting the knob during intense gameplay is suboptimal.

7.2 System Performance

We evaluate Kalɛido's real-time performance and measure its processing delay on a commodity PC with an Intel i7-7700 CPU and Nvidia GTX 1080 GPU. Figure 9 shows the latency overheads incurred by the three main operations of Kalɛido: noisy gaze generation (noising), keyframe detection, and ROI detection. We run 100 trials for each of the operations and report the average running time. The latency of the noising operation is only 8 µs, and thus, has no discernible impact on the user's real-time experience.

ROI detection takes 80 ms on average, but it only runs when a new keyframe is detected. Based on our offline game calibration, a new keyframe is detected only every 2.3 s (similar to the timing from the VR videos dataset). Thus, the overall impact of ROI detection in Kalɛido is not significant.

Keyframe detection takes 33 ms on average. The frequency of keyframe detection (context update rate) is comparatively higher (2 Hz in our implementation). Figure 9(b) shows its performance impact on effective frame rates of the game used in the study. We observe that, even with a high context update rate of 8 Hz, the frame rate degrades only slightly to 25 Hz.

In this paper, we evaluate a research prototype of Kalɛido, which shows its real-world potential. Nevertheless, to deploy in scale, Kalɛido can leverage various performance optimizations, such as GPU offloading, model compression, and resource sharing. These optimizations would enable fast context processing even on resource-constrained platforms.

7.3 Effectiveness Against Attacks

Recall that post-processing operations on the outputs of a DP algorithm do not result in additional privacy loss (Theorem 3). Thus, Kalɛido's formal DP guarantee for the spatial information of gaze streams holds for every attacker (even for one with full knowledge of Kalɛido's protocols). However, Kalɛido does not provide a formal guarantee on the temporal information of gaze streams (Section 4.3.2). Hence, we perform a trace-based evaluation to study the effectiveness of Kalɛido against spatio-temporal attacks using the datasets in Table 1. These attacks exploit the spatio-temporal features of gaze streams, such as fixation durations and saccade velocity [31, 74]. We select two representative analyses of gaze streams: (1) similarity and outlier analysis of a scanpath for an individual, and (2) biometric inferences. We use (1) MultiMatch [20] for computing the scanpath similarity scores, and (2) F1 score, which considers both precision and recall, to measure attackers' classification accuracy.

Note that the attackers considered in this section are knowledgeable; they have complete knowledge of the target visual scenes and Kalɛido's noise generation protocols. Further, they use a noise-robust fixation detection [31]. Additionally, all the classifiers used in this section are trained on noisy gaze streams from Kalɛido (for the same privacy configurations).

7.3.1 Similarity and Outlier Analysis of Scanpath

Given a dataset of gaze streams for single scenes, this attack constructs a feature vector of the scanpath for each individual in the dataset. Since the visual stimulus is the same, the hypothesis is that the differences in the scanpath features arise from distinguishing psychophysiological traits. Thus, this type of analysis aims at distinguishing individuals based on their scanpath features [9].

Setup. We use the image datasets (the first three rows of Table 1: natural, web page, and human) to evaluate the distinguishability of the scanpath features on static image frames. This evaluation assesses the accuracy of the analysis of raw and noisy gaze streams. For each stream, we extract the scanpaths using an offline algorithm [31]. Next, we perform similarity analysis and outlier identification as follows.

Similarity analysis. The adversary here has a priori knowledge of a user's scanpath on a certain image. It attempts to re-identify the user by measuring the similarity between this scanpath and a newly observed one formed on the same image. For each dataset, we compute the similarity between the scanpaths of the same user, before and after adding noise. We use the standardized similarity metric, MultiMatch [20], which ranges from 0 to 1. This score measures scanpath similarity by considering features about the shape (the length, shape, and direction of saccade vectors) and the spatial distribution (position and duration of aligned fixations) of gaze data.

Outlier identification. In this attack, the adversary tries to identify the outlier users whose scanpath features are significantly different from that of the rest. This attack utilizes a density-based clustering model DBSCAN [28], where inter-scanpath distances are computed via dynamic time warping (DTW) over the scanpaths on a single image. We use the F1 score to report the attacker's success in identifying the outlier users from the dataset containing noisy gaze streams. We show the F1 scores of outlier identification compared to random guessing as a baseline ("Random guess").

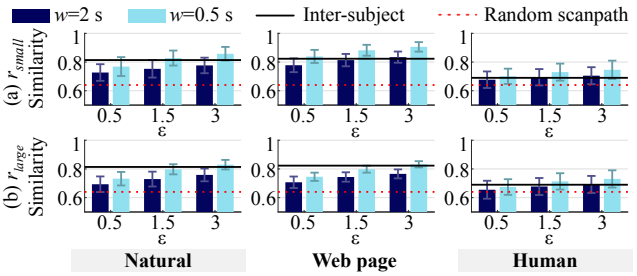

Figure 10: Similarity scores between noisy and raw scanpaths. Kalɛido reduces the similarity scores to be close to the inter-subject threshold (black lines) even at low privacy configurations (r_{small}). The scores are reduced further to be close to the random scanpath baseline (red dash lines) at high privacy configurations ($\varepsilon = 0.5$, r_{large}, and $w = 2$ s).

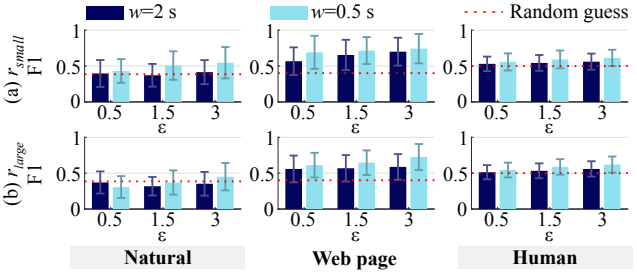

Figure 11: F1 scores of outlier identification among scanpaths. At high privacy configurations (low values of ε, r_{large}, and $w = 2$ s), Kalɛido thwarts outlier identification attacks in all three datasets by reducing F1 scores to be close to the random guess baseline (red dash lines).

Results. *Similarity analysis.* In Figure 10, we compare the measured similarity with two thresholds: (1) mean inter-subject similarity score ("Inter-subject") in each dataset, and (2) the similarity of two randomly synthesized scanpaths presented in [20] ("Random scanpath"). Figure 10 shows a consistent trend in all three image datasets: the scanpath similarity decreases with higher privacy level (i.e., smaller ε, larger w, and larger r). Kalɛido degrades the similarity score below the inter-subject threshold, even though it perturbs the spatial data only; at $\varepsilon = 0.5$, Kalɛido brings the similarity score close to the random scanpath baseline.

Outlier identification. As observed from Figure 11,

Kalɛido degrades the effectiveness of outlier identification for all of the privacy settings. For the natural and human image datasets, Kalɛido reduces the attacker's F1 scores to the random guess using r_{large} with ε as high as 3. Although the attacker's F1 score remains relatively high in the web page dataset, it is reduced significantly for $\varepsilon = 0.5$.

7.3.2 Biometric Inferences

Setup. We construct attacks that attempt to predict (1) users' identities and (2) whether the users wore contact lenses for vision correction (use of contact lenses leads to distinguishing eye gaze patterns [63]).

For this experiment, we use the VR video dataset (last row in Table 1). The associated classification labels are provided in the dataset. This attack uses aggregate statistics of fixation/saccade features over several VR video sessions as training data and predicts users' identities and vision conditions for an unseen session. Specifically, each video session uses a different VR context for the same user. Hence, the evaluation of biometric inferences here assesses Kalɛido's effectiveness against linkability attacks across different contexts (this has been exploited in prior work [22]). We adopt the features suggested by the Cluster Fix toolbox [44], which are then used to train a discriminant analysis classifier [19]. This evaluation includes 11 users from the VR video dataset who comfortably completed all 12 video sessions. Additionally, the training and test sets correspond to the same privacy configuration, i.e., either raw gaze streams or noisy gaze streams. We report the F1 scores for leave-one-out cross-validation.

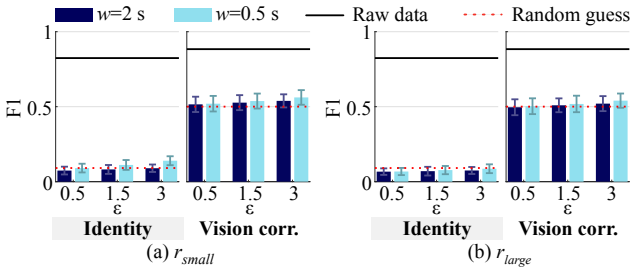

Figure 12: F1 scores of predicting user identity and vision correction. Kalɛido reduces the F1 scores of biometric inferences to be close to random guess baselines (red dash lines) even for low privacy configurations (high values of ε or r_{small}).

Results. Figure 12 shows the F1 scores obtained from both the raw and noisy gaze streams. For both classifiers (identity and vision correction), the raw gaze streams enable accurate classification – the F1 score is close to 1 ("Raw data" in Figure 12) and is much higher than that of random guess. This indicates that the attacker can successfully predict users' identities and vision correction conditions, even across different contexts. On the other hand, we observe that Kalɛido significantly degrades the attacker's classification accuracy to be close to the random baseline even for low privacy configurations (high values of ε or r_{small}).

8 Discussion

Kalɛido is a first step toward designing real-time eye-tracking systems that provide a formal privacy guarantee. Here, we discuss several possible avenues for future research:

Support for more data formats and types. An eye-tracking platform may offer eye-tracking data in various formats such as 2D gaze positions and 3D gaze positions. Currently, Kalɛido is designed for 2D gaze streams and supports head-and-eye gaze streams as well (discussed in Appendix A.2.2). Extension to 3D gaze streams is possible and would involve extending the PlanarLap mechanism (Algorithm 1 to 3D positions. Additionally, some eye-tracking cores collect data including blink timing and pupil dilation. Kalɛido's scope of privacy can be further broadened to address these data types.

Privacy guarantee for temporal information. Kalɛido can be extended to protect the temporal information of eye gaze streams by interfering with the timeliness of gaze releases. For example, for fixation duration (a popular aggregate statistic), Kalɛido can decide on a predefined threshold T based on standard human gaze fixations [34]. Next, stage I and II from Algorithm 1 can be replaced by a sophisticated fixation detection approach such as online differentially private clustering [43, 55], which (1) releases a single noisy position in the first T duration of a fixation and (2) stops any further data release for the given fixation. This ensures that the duration for all fixation events in the noisy gaze stream is fixed to T.

Optimization for long scenes. Although visual content in an eye-tracking application is typically dynamic, it might remain relatively static for long periods in some cases. Such long scenes that span multiple windows may lead to a large privacy budget consumption. Techniques including noisy data caching can be used to help address this issue. Specifically, Kalɛido can check online if the current ROI has been visited previously, and it can reuse the corresponding noisy gazes from recent history. Additionally, for applications where interactions are sporadic, Kalɛido can skip releasing new gazes for scenes when the user is inactive to save the privacy budget.

Optimizations for context processing. One interesting future direction can be optimizing Kalɛido's context processing core. The overhead of Kalɛido's context processing can be reduced by sharing the detection module with other applications. Kalɛido can leverage other models for ROI detection, including Selective Search [85] and Faster R-CNN [72], which may be implemented by the platform already. For instance, eye-tracking platforms, such as Hololens [59], provide certain context information that Kalɛido can use directly for performance optimization. Additionally, smart calibration of the frequency of key frame detection can also reduce the overhead of context processing.

Optimizations for privacy budget allocation. In this paper,

the presented composition theorem (Theorem 2) is based on the simple k-fold composition of the DP guarantee [21]. However, a tighter analysis might be possible via advanced composition [21] and moment-based accounting [1].

Evaluation of other utility metrics. In this paper, we primarily focus on qualitatively evaluating Kalɛido's utility for the use case of a real-time game (as demonstrated in Section 7). However, as mentioned in Section 2.2, eye-tracking data is used for diverse purposes. Hence, an important future direction is to investigate user perception for other online applications and quantitatively evaluate Kalɛido's utility for offline gaze data analysis (Kalɛido's impact on fixation saliency maps is presented in Appendix A.2.3). Another direction could be exploring application-specific utility optimizations. For instance, data-smoothing techniques can be used to improve the accuracy of the noisy gaze streams.

9 Conclusion

We have designed and implemented Kalɛido, an eye gaze processing system that (1) provides a formal privacy guarantee on the spatial distribution of raw gaze positions, (2) seamlessly integrates with existing eye-tracking ecosystems, and (3) is capable of operating in real-time. Kalɛido acts as an intermediary protection layer between the eye-tracking platform and the applications. Our evaluation results show that users enjoy a satisfactory level of utility while deploying Kalɛido for an interactive eye-tracking game. Additionally, it is successful in thwarting real-world spatio-temporal attacks on gaze streams.

Acknowledgments

We thank our user study participants, the anonymous reviewers, and the shepherd, Apu Kapadia, for their contributions and valuable suggestions. This project is supported in part by NSF under grants 1719336, 1845469, 1838733, 1942014, 2003129, and 1931364.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *ACM CCS*, 2016.

[2] J. M. Abowd and I. M. Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *American Economic Review*, 109(1):171–202, 2019.

[3] A. Açık, A. Sarwary, R. Schultze-Kraft, S. Onat, and P. König. Developmental changes in natural viewing behavior: bottom-up and top-down differences between children, young adults and older adults. *Frontiers in Psychology*, 1:207, 2010.

[4] I. Agtzidis, M. Startsev, and M. Dorr. 360-degree video gaze behaviour: A ground-truth data set and a classification algorithm for eye movements. In *ACM MM*, 2019.

[5] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *ACM CCS*, 2013.

[6] E. Arabadzhiyska, O. T. Tursun, K. Myszkowski, H. Seidel, and P. Didyk. Saccade landing position prediction for gaze-contingent rendering. *ACM TOG*, 36(4):1–12, 2017.

[7] K. Bannier, E. Jain, and O. Le Meur. Deepcomics: Saliency estimation for comics. In *ACM ETRA*, 2018.

[8] W. Becker and A. F. Fuchs. Further properties of the human saccadic system: eye movements and correction saccades with and without visual fixation points. *Vision Research*, 9(10):1247–1258, 1969.

[9] S. A. Beedie, D. M. St. Clair, and P. J. Benson. Atypical scanpaths in schizophrenia: evidence of a trait-or state-dependent phenomenon? *Journal of Psychiatry & Neuroscience*, 36(3):150, 2011.

[10] S. Berkovsky, R. Taib, I. Koprinska, E. Wang, Y. Zeng, J. Li, and S. Kleitman. Detecting personality traits using eye-tracking data. In *ACM CHI*, 2019.

[11] A. Borji, D. N. Sihite, and L. Itti. Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE TIP*, 22(1):55–69, 2012.

[12] E. Bozkir, O. Günlü, W. Fuhl, R. F. Schaefer, and E. Kasneci. Differential privacy for eye tracking with temporal correlations. *arXiv:2002.08972*, 2020.

[13] E. Bozkir, A. B. Ünal, M. Akgün, E. Kasneci, and N. Pfeifer. Privacy preserving gaze estimation using synthetic images via a randomized encoding based framework. *arXiv:1911.07936*, 2019.

[14] F. Broz, H. Lehmann, B. Mutlu, and Y. Nakano. *Gaze in Human-Robot Communication*, volume 81. John Benjamins Publishing Company, 2015.

[15] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand. What do different evaluation metrics tell us about saliency models? *IEEE TPAMI*, 41(3):740–757, 2018.

[16] S. Castagnos, N. Jones, and P. Pu. Eye-tracking product recommenders' usage. In *ACM RecSys*, 2010.

[17] G. Charness, U. Gneezy, and M. A. Kuhn. Experimental methods: Between-subject and within-subject design. *Journal of Economic Behavior & Organization*, 81(1):1–8, 2012.

[18] K. Chatzikokolakis, C. Palamidessi, and M. Stronati. A predictive differentially-private mechanism for mobility traces. In *PETS*, 2014.

[19] A. Coutrot, J. H. Hsiao, and A. B. Chan. Scanpath modeling and classification with hidden markov models. *Behavior Research Methods*, 50(1):362–379, 2018.

[20] R. Dewhurst, M. Nyström, H. Jarodzka, T. Foulsham, R. Johansson, and K. Holmqvist. It depends on how you look at it: Scanpath comparison in multiple dimensions with multimatch, a vector-based approach. *Behavior Research Methods*, 44(4):1079–1100, 2012.

[21] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(Nos. 3-4):211–407, 2014.

[22] S. Eberz, G. Lovisotto, A. Patane, M. Kwiatkowska, V. Lenders, and I. Martinovic. When your fitness tracker betrays you: Quantifying the predictability of biometric features across contexts. In *IEEE S&P*, 2018.

[23] S. Eberz, G. Lovisotto, K. B. Rasmussen, V. Lenders, and I. Martinovic. 28 blinks later: Tackling practical challenges of eye movement biometrics. In *ACM CCS*, 2019.

[24] S. Eraslan, Y. Yesilada, and S. Harper. Scanpath trend analysis on web pages: Clustering eye tracking scanpaths. *ACM TWEB*, 10(4):1–35, 2016.

[25] GazeRecorder. *GazeRecorder–webcam eye tracking*, 2020. https://gazerecorder.com.

[26] A. Gibaldi, M. Vanegas, P. J. Bex, and G. Maiello. Evaluation of the tobii eyex eye tracking controller and matlab toolkit for research. *Behavior Research Methods*, 49(3):923–946, 2017.

[27] S. R. Gulliver and G. Ghinea. The perceptual and attentive impact of delay and jitter in multimedia delivery. *IEEE Transactions on Broadcasting*, 53(2):449–458, 2007.

[28] M. J. Haass, L. E. Matzen, K. M. Butler, and M. Armenta. A new method for categorizing scanpaths from eye tracking data. In *ACM ETRA*, 2016.

[29] I. Hagestedt, M. Backes, and A. Bulling. Adversarial attacks on classifiers for eye-based user modelling. In *ACM ETRA*, 2020.

[30] R. S. Hessels, C. Kemner, C. van den Boomen, and I. T. C. Hooge. The area-of-interest problem in eyetracking research: A noise-robust solution for face and sparse stimuli. *Behavior Research Methods*, 48(4):1694–1712, 2016.

[31] R. S. Hessels, D. C. Niehorster, C. Kemner, and I. T. C. Hooge. Noise-robust fixation detection in eye movement data: Identification by two-means clustering (i2mc). *Behavior Research Methods*, 49(5):1802–1823, 2017.

[32] C. Holland, A. Garza, E. Kurtova, J. Cruz, and O. Komogortsev. Usability evaluation of eye tracking on an unmodified common tablet. In *ACM CHI EA*, 2013.

[33] C. Holland and O. V. Komogortsev. Biometric identification via eye movement scanpaths in reading. In *IEEE IJCB*, 2011.

[34] I. TH. C. Hooge and C. J. Erkelens. Adjustment of fixation duration in visual search. *Vision Research*, 38(9):1295–IN4, 1998.

[35] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth. Differential privacy: An economic method for choosing epsilon. In *IEEE CSF*, 2014.

[36] H. Hutchinson and et al. Technology probes: Inspiring design for and with families. In *ACM CHI*, 2003.

[37] S. Jalaliniya, D. Mardanbegi, I. Sintos, and D. G. Garcia. Eyedroid: an open source mobile gaze tracker on android for eyewear computers. In *ACM UbiComp*, 2015.

[38] S. Jana, A. Narayanan, and V. Shmatikov. A scanner darkly: Protecting user privacy from perceptual applications. In *IEEE S&P*, 2013.

[39] M. Jiang, S. Huang, J. Duan, and Q. Zhao. Salicon: Saliency in context. In *IEEE CVPR*, 2015.

[40] B. John, P. Raiturkar, O. Le Meur, and E. Jain. A benchmark of four methods for generating 360° saliency maps from eye tracking data. *IJSC*, 13(03):329–341, 2019.

[41] Z. Kapoula, Q. Yang, J. Otero-Millan, S. Xiao, S. L. Macknik, A. Lang, M. Verny, and S. Martinez-Conde. Distinctive features of microsaccades in alzheimer's disease and in mild cognitive impairment. *Age*, 36(2):535–543, 2014.

[42] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. In *VLDB*, 2014.

[43] M. Khavkin and M. Last. Preserving differential privacy and utility of non-stationary data streams. In *IEEE ICDMW*, 2018.

[44] S. D. König and E. A. Buffalo. A nonparametric method for detecting fixations and saccades using cluster analysis: Removing the need for arbitrary thresholds. *Journal of Neuroscience Methods*, 227:121–131, 2014.

[45] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *IEEE CVPR*, 2016.

[46] M. Kumar, T. Winograd, and A. Paepcke. Gaze-enhanced scrolling techniques. In *ACM CHI EA*, 2007.

[47] Pupil Labs. *Gaze Datum Format*, 2020. https://docs.pupil-labs.com/developer/core/overview/#gaze-datum-format.

[48] D. Lamas, F. Loizides, L. Nacke, H. Petrie, M. Winckler, and P. Zaphiris. *Human-Computer Interaction–INTERACT 2019*, volume 11748. Springer, 2019.

[49] M. F. Land and M. Hayhoe. In what ways do eye movements contribute to everyday activities? *Vision Research*, 41(25-26):3559–3565, 2001.

[50] J. Lee and C. Clifton. How much is enough? choosing ε for differential privacy. In *ISC*, 2011.

[51] Y. Li, Z. Cao, and J. Wang. Gazture: Design and implementation of a gaze based gesture control system on tablets. *ACM IMWUT*, 1(3):1–17, 2017.

[52] TY. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[53] A. Liu, L. Xia, A. Duchowski, R. Bailey, K. Holmqvist, and E. Jain. Differential privacy for eye-tracking data. In *ACM ETRA*, 2019.

[54] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *ACM MobiCom*, 2019.

[55] Z. Lu and H. Shen. Differentially private k-means clustering with guaranteed convergence. *arXiv:2002.01043*, 2020.

[56] P. Majaranta and A. Bulling. Eye tracking and eye-based human–computer interaction. In *Advances in Physiological Computing*, pages 39–65. Springer, 2014.

[57] S. Marwecki, A. D. Wilson, E. Ofek, M. Gonzalez Franco, and C. Holz. Mise-unseen: Using eye tracking to hide virtual reality scene changes in plain sight. In *ACM UIST*, 2019.

[58] S. A. McMains and S. Kastner. Visual attention. *Encyclopedia of Neuroscience*, 1:4296–4302, 2009.

[59] Microsoft. *Scene understanding SDK overview*, 2020. https://docs.microsoft.com/en-us/windows/mixed-reality/develop/platform-capabilities-and-apis/scene-understanding-SDK.

[60] A. E. Millen and P. J. B. Hancock. Eye see through you! eye tracking unmasks concealed face recognition despite countermeasures. *Cognitive Research: Principles and Implications*, 4(1):23, 2019.

[61] E. Miluzzo, T. Wang, and A. T. Campbell. Eyephone: activating mobile phones with your eyes. In *ACM MobiHeld*, 2010.

[62] C. H. Morimoto and M. R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1):4–24, 2005.

[63] C. Müller, W. Stoll, and F. Schmäl. The effect of optical devices and repeated trials on the velocity of saccadic eye movements. *Acta Oto-Laryngologica*, 123(4):471–476, 2003.

[64] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *ACM SOTC*, 2007.

[65] S. Oya, C. Troncoso, and F. Pérez-González. Is geo-indistinguishability what you are looking for? In *ACM WPES*, 2017.

[66] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *IJCAI*, 2016.

[67] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM TOG*, 35(6):179, 2016.

[68] R. Pieters, E. Rosbergen, and M. Wedel. Visual attention to repeated print advertising: A test of scanpath theory. *Journal of Marketing Research*, 36(4):424–438, 1999.

[69] N. Raval, A. Srivastava, K. Lebeck, L. Cox, and A. Machanavajjhala. Markit: Privacy markers for protecting visual secrets. In *ACM UbiComp*, 2014.

[70] K. Rayner, M. S. Castelhano, and J. Yang. Eye movements when looking at unusual/weird scenes: Are there cultural differences? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(1):254, 2009.

[71] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[72] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[73] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang. World-driven access control for continuous sensing. In *ACM CCS*, 2014.

[74] D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *ACM ETRA*, 2000.

[75] A. Sanchez, C. Vazquez, C. Marker, J. LeMoult, and J. Joormann. Attentional disengagement predicts stress recovery in depression: An eye-tracking study. *Journal of Abnormal Psychology*, 122(2):303, 2013.

[76] J. S. Silk, L. R. Stroud, G. J. Siegle, R. E. Dahl, K. H. Lee, and E. E. Nelson. Peer acceptance and rejection through the eyes of youth: pupillary, eyetracking and ecological data from the chatroom interact task. *Social Cognitive and Affective Neuroscience*, 7(1):93–105, 2012.

[77] M. Siqueiros Sanchez, E. Pettersson, D. P. Kennedy, S. Bölte, P. Lichtenstein, B. M. D'Onofrio, and T. Falck-Ytter. Visual disengagement: Genetic architecture and relation to autistic traits in the general population. *Journal of Autism and Developmental Disorders*, 2019.

[78] P. C. Stacey, S. Walker, and J. D. M. Underwood. Face processing and familiarity: Evidence from eye-movement data. *British Journal of Psychology*, 96(4):407–422, 2005.

[79] J. Steil, I. Hagestedt, M. X. Huang, and A. Bulling. Privacy-aware eye tracking using differential privacy. In *ACM ETRA*, 2019.

[80] W. Steptoe, R. Wolff, A. Murgia, E. Guimaraes, J. Rae, P. Sharkey, D. Roberts, and A. Steed. Eye-tracking for avatar eye-gaze and interactional analysis in immersive collaborative virtual environments. In *ACM CSCW*, 2008.

[81] A. Strauss and J. Corbin. *Basics of Qualitative Research Techniques*. Sage, 1998.

[82] P. Termsarasab, T. Thammongkolchai, J. C. Rucker, and S. J. Frucht. The diagnostic value of saccades in movement disorder patients: a practical guide and review. *Journal of Clinical Movement Disorders*, 2(1):14, 2015.

[83] Tobii. *Scripting API of Tobii Unity SDK*, 2020. https://developer.tobii.com/pc-gaming/unity-sdk/scripting-api/.

[84] T. Toyama, D. Sonntag, A. Dengel, T. Matsuda, M. Iwamura, and K. Kise. A mixed reality head-mounted text translation system using eye gaze input. In *ACM IUI*, 2014.

[85] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[86] Unity. *Scripting Reference of XR.Eyes*, 2020. https://docs.unity3d.com/ScriptReference/XR.Eyes.html.

[87] Unity. *Survival shooter tutorial*, 2020. https://learn.unity.com/project/survival-shooter-tutorial/?tab=overview.

[88] Unity. *Unity Scripting API: Keyframe*, 2020. https://docs.unity3d.com/ScriptReference/Keyframe.html.

[89] J. Varona, C. Manresa-Yee, and F. J. Perales. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, 31(4):357–374, 2008.

[90] R. J. Wang, X. Li, and C. X. Ling. Pelee: A real-time object detection system on mobile devices. In *NIPS*, 2018.

[91] N. Wilming, S. Onat, J. P. Ossandón, A. Açık, T. C. Kietzmann, K. Kaspar, R. R. Gameiro, A. Vormberg, and P. König. An extensive dataset of eye movements during viewing of complex images. *Scientific Data*, 4(1):1–11, 2017.

[92] S. Xu, H. Jiang, and F. C. M. Lau. Personalized online document, image and video recommendation via commodity eye-tracking. In *ACM RecSys*, 2008.

[93] Z. Ye, Y. Li, A. Fathi, Y. Han, A. Rozga, G. D. Abowd, and J. M. Rehg. Detecting eye contact using wearable eye-tracking glasses. In *ACM Ubicomp*, 2012.

[94] X. Yi and N. Ling. Fast pixel-based video scene change detection. In *IEEE ISCAS*, 2005.

A Appendix

A.1 Proof of Theorem 2

Theorem 2 (Composition over multiple windows theorem). *Let $\mathcal{M} : \mathcal{S}^g \mapsto \mathcal{C}^g$ be a mechanism that takes as input a gaze stream prefix $S_k^g = (\langle g_1, t_1\rangle, \cdots, \langle g_k, t_k\rangle)$, and outputs a transcript $O = (o_1, \cdots, o_k) \in \mathcal{C}$. Additionally, let $\mathcal{M}$ be decomposed into k mechanisms $\mathcal{M}_1, \cdots, \mathcal{M}_k$ such that $\mathcal{M}_i(g_i) = o_i$, and each $\mathcal{M}_i$ generates independent randomness while achieving (ε_i, r)-geo-indistinguishability. Then for two stream prefixes S_k^g and $S_k^{g'}$ such that*

- *for all $i \in [k]$, $t_i = t'_i$*
- *for each g_i, g'_i such that $i \in [k]$ and $g_i \neq g'_i$ it holds that $d(g_i, g'_i) \leq r$, i.e., $(g_i, g_{i'})$ are r-Euclidean neighboring, and*

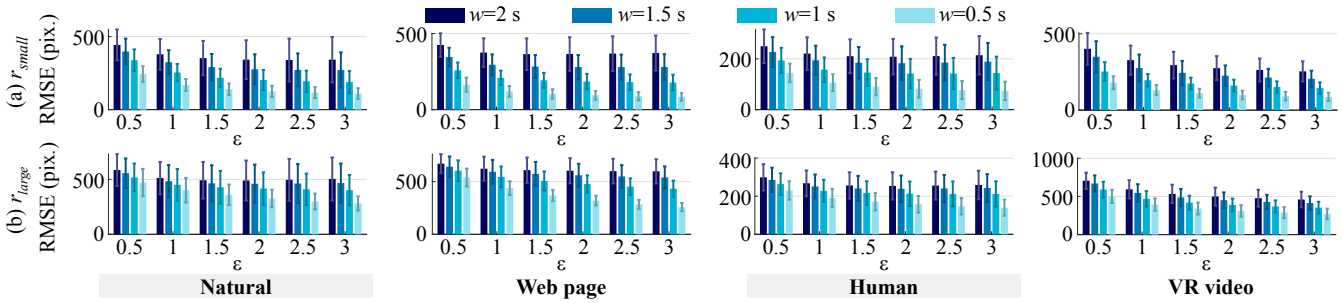

Figure 13: Privacy-accuracy trade-off of Kalɛido.

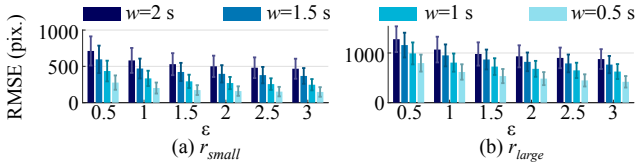

Figure 14: Privacy-accuracy trade-off of Kalɛido on head-and-eye gaze data.

- *for each* $g_{i_1}, g_{i_2}, g'_{i_1}, g'_{i_2}$, *with* $i_1 < i_2, g_{i_1} \neq g'_{i_1}$ *and* $g_{i_2} \neq g'_{i_2}$, *it holds that* $t_{i_2} - t_{i_1} \leq m \cdot w, m \in \mathbb{N}$

$$\forall O \in \mathcal{C}^g, \forall k, Pr[\mathcal{M}(S^g_k) = O] \leq e^{m \cdot \varepsilon} \cdot Pr[\mathcal{M}(S^{g'}_k) = O] \quad (8)$$

Proof. Let $m = 2$ and i_1 be the least index such that $g_{i_1} \neq g'_{i_1}$ and i_2 be the highest index such that $g_{i_2} \neq g'_{i_2}$. Additionally, let $i* \in [i_1, i_2]$ such that $time(i*) - time(i_1) = w$. Let $S^g_{i*} = (\langle g_1, t_1 \rangle \cdots \langle g_{i*}, t_{i*} \rangle)$,$S^g_{k*} = (\langle g_{i*+1}, t_{i*+1} \rangle \cdots \langle g_k, t_k \rangle)$ and $O = O_1 || O_2, |O_1| = |S^g_{i*}|, |O_2| = |S^g_{k*}|, O \in \mathcal{C}^g$. Now using the independence of noise generation for each gaze position,

$$\begin{aligned} Pr[\mathcal{M}(S^g_k) = O] &= Pr[\mathcal{M}(S^g_{i*}) = O_1] \cdot Pr[\mathcal{M}(S^g_{k*}) = O_2] \\ &\leq e^{\varepsilon} \cdot Pr[\mathcal{M}(S^{g'}_{i*}) = O_1] \cdot e^{\varepsilon} \cdot Pr[\mathcal{M}(S^{g'}_{k*}) = O_2] \\ &= e^{2\varepsilon} \cdot Pr[\mathcal{M}(S^{g'}_k) = O] \end{aligned}$$

The rest of the proof follows trivially using induction using the above case as the base. □

A.2 Additional Experimental Results

A.2.1 Privacy-Accuracy Trade-off

In Figure 13, we study the privacy-accuracy trade-off for varying configurations of Kalɛido. The utility is measured by the root mean square error (RMSE) in pixel. We vary the parameters as follows: $\varepsilon \in \{0.5, 1, 1.5, 2, 2.5, 3\}$, $w \in \{0.5, 1, 1.5, 2\}$ and $r \in \{r_{small}, r_{large}\}$. We generate 100 random trials for each combination and report the mean observation. In all the datasets, we observe a clear trend of accuracy improvement (lower RMSE) with increasing privacy budget ε or decreasing window duration w. At the same value of ε and w, using r_{large} gives lower accuracy than r_{small}.

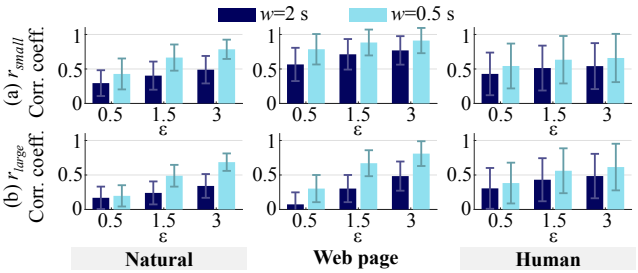

Figure 15: Kalɛido's impact on saliency map at varying privacy configurations.

A.2.2 Kalɛido's Effect on Head-and-Eye Gaze Data

We show the privacy-accuracy trade-off for Kalɛido for head-and-eye gaze data for the VR video dataset in Figure 14. The observations are consistent with Figure 13 of just eye gazes.

A.2.3 Kalɛido's Effect on Fixation Saliency Map

In some cases, the application utility might require extracting the saliency maps [7, 40] from users' fixations. Figure 15 shows Kalɛido's impact on the saliency maps. We compute the correlation coefficient, a standard metric for saliency map similarity [15], between each user's clean and noisy maps. For all the datasets, Kalɛido's accuracy (higher correlation coefficient) [11] increases with increase in the privacy budget ε or decrease in window duration w. At the same value of ε and w, using r_{large} gives lower accuracy than r_{small}. These results are consistent was Kalɛido's premise: it attempts to hide the spatial patterns of the user's fixations. A lower value of ε would result in less accurate extraction of the saliency maps.