EvaluatAR: A Cross-Device Evaluation Framework for Rapid Prototyping of Bystander PETs in AR

Syed Ibrahim Mustafa Shah Bukhari
Virginia Tech
simsb@vt.edu

Matthew Corbett
Army Cyber Institute at West Point
matthew.corbett@westpoint.edu

Bo Ji
Virginia Tech
boji@vt.edu

Brendan David-John
Virginia Tech
bmdj@vt.edu

Abstract

Augmented Reality (AR) headsets continuously sense their surroundings, capturing nearby bystanders and raising privacy risks. Visual bystander privacy-enhancing technologies (PETs) mitigate this risk by detecting bystanders in egocentric scene views and applying privacy transformations (e.g., obfuscation). However, traditional PET evaluation is human-dependent, high-overhead, and device-specific, making it difficult to reproduce across devices. We present EvaluatAR, a cross-device evaluation framework for rapid prototyping at the early stage of PET evaluation. Our framework enables controlled replication of experimental conditions by standardizing PET inputs (sensor data and visual stimuli) and outputs through a record-replay workflow. We validate EvaluatAR through three case studies on HoloLens 2, Magic Leap 2, and Meta Quest 3 across implicit (continuous, context-driven) and explicit (intent-driven) PETs: (1) cross-device replay of inputs to a PET to reveal device-specific privacy-performance trade-offs; (2) generalizability of the same framework workflow across implicit and explicit PET design categories; and (3) replay of privacy-relevant edge cases to diagnose failures and validate PET modifications, yielding an improvement over the state-of-the-art baseline. These results demonstrate EvaluatAR's support for rapid, iterative PET development to advance reproducible cross-device evaluation of bystander PETs at a critical moment in the emergence of ubiquitous AR.

Keywords

Privacy-enhancing technologies (PETs), bystander privacy, evaluation framework

1 Introduction

Augmented Reality (AR) headsets create immersive experiences by integrating multiple sensors, including outward-facing cameras, depth sensors, microphones, and biometric sensors (e.g., eye tracking) [6, 35, 53]. Advances in AR hardware [5, 18, 41] have accelerated mainstream adoption, with market size projected to reach $1.05 trillion by 2030, an estimated 29.7% increase from 2025 [1]. Increasing adoption of AR in industries like entertainment, healthcare, manufacturing, and education [12, 39, 46, 50] exacerbates existing privacy issues associated with its always-on sensing.

Always-on sensing poses a key privacy risk for nearby bystanders, especially when they are unaware of the sensing or do not consent to being recorded. Prior work shows that data captured by AR headsets can reveal sensitive personally identifiable information of bystanders, including facial identity, sex, race, age, and sexual preference [8, 11, 13, 23, 30, 34, 49, 60]. These risks are compounded by AR headset designs that often conceal sensors, making it difficult for bystanders to recognize when and how their privacy might be compromised [10, 38, 51]. This data may be shared without the consent of bystanders, resulting in undesired tracking or profiling [21, 27], and creating unease in bystanders regarding AR technology [22, 42]. This unauthorized disclosure of bystanders' data is known as the bystander privacy protection (BPP) problem [15, 16], and visual sensing is a prominent driver of it in wearable technologies [31].

To address BPP concerns in visual sensing pipelines in AR, researchers have developed privacy-enhancing technologies (PETs) to protect bystanders' privacy (e.g., detecting bystanders and applying privacy transformations such as obfuscation) [15, 28, 29, 45, 57, 61]. These PETs are typically evaluated along three key dimensions: *usability* (bystander perceptions such as comfort and acceptability of PET's outputs), *privacy protection* (scenario-based protection metrics), and *performance* (system responsiveness, often reported as frames per second (FPS)). For example, Corbett et al. [15] evaluate BystandAR through a user study with 16 participants to assess system usability through survey responses, report privacy protection metrics computed from the dataset, and evaluate its performance on the HoloLens 2 with dedicated experiments. This example exemplifies the existing PET evaluation pipeline that has several limitations.

Gap. The current PET evaluation pipeline is *human-dependent*, i.e., it involves multiple experimenters and participants, introducing confounds due to variations in human behavior (bystander movement, viewpoint changes, etc.). It also has *high overheads*, i.e., it requires substantial investment in engineering effort, study design, participant recruitment and compensation, and relies on experimenter availability. Moreover, system performance evaluation in the existing pipeline is *device-specific* or *limited*, i.e., performance is measured only on the device on which the PET is developed. **These limitations imply that the existing PET evaluation pipeline is difficult to reproduce, limiting our understanding of PETs' generalizability, particularly their cross-device performance and privacy-performance trade-offs.**

Proceedings on Privacy Enhancing Technologies 2026(4), 947–963

https://doi.org/10.56553/popets-2026-0153

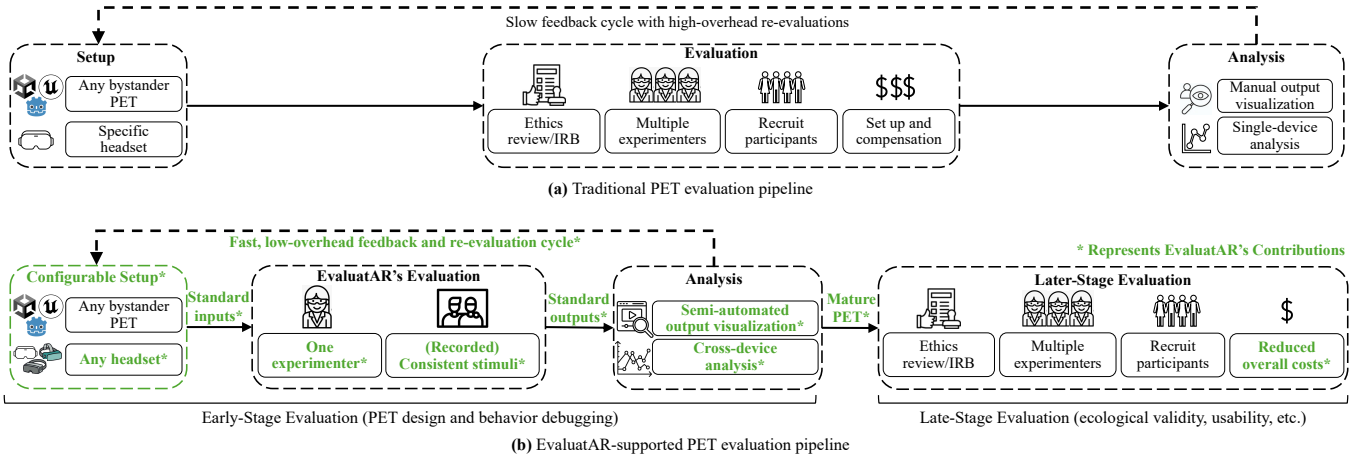


Figure 1: (a) Traditional evaluation is human-dependent and device-specific, coupling PET debugging and validation with live human-subject studies. This incurs substantial setup, recruitment/compensation, and experimenter overheads, while introducing confounds from variations in human behavior. (b) EvaluatAR introduces a low-overhead evaluation stage for controlled replication. This stage supports cross-device design validation, privacy-performance profiling, and rapid iteration before later-stage human-subject studies of mature PETs focused on ecological validity, user experience, and social acceptability.

We address this gap by introducing EvaluatAR, a framework that standardizes PETs' inputs (sensor data and visual stimuli) and outputs, to enable controlled cross-device evaluation of PETs under consistent evaluation conditions. We introduce EvaluatAR at the early stage of the PET evaluation pipeline, focusing on testing PET's design feasibility and robustness. Hence, we focus our evaluation of PETs via EvaluatAR on assessing their generalizability and to profile their performance and privacy-performance trade-offs, enabling rapid prototyping. In doing so, EvaluatAR reduces the overheads associated with the existing PET evaluation pipeline by enabling the development of mature PET designs before later-stage, high-overhead human-subject studies that focus on testing PET's ecological validity, user experience, and social acceptability.

Use Case. Consider a researcher who has developed a novel visual bystander PET and wants to evaluate it. Traditionally, as illustrated in Fig. 1a, this requires live human-subject studies, ethics review/IRB approval, participant recruitment and compensation, and device-specific data collection and analysis. Even with this overhead, the evaluation remains limited by the number of testable conditions, confounds from variations in human behavior, and difficulties assessing how the PET generalizes across headsets.

Using EvaluatAR instead, they would perform a low-overhead early-stage evaluation centered on controlled replication. After selecting videos depicting relevant scenarios, they integrate their PET with EvaluatAR through lightweight input/output hooks, collect scenario-synced sensor data, and replay these inputs across trials and headsets. This enables reproducible, low-overhead, cross-device evaluation for design validation, debugging, privacy-performance profiling, and rapid iteration on failure cases under controlled conditions. As illustrated in Fig. 1b, EvaluatAR-supported pipeline helps researchers identify and address major PET design issues early on to produce mature PETs for later-stage evaluation, focusing on ecological validity, user experience, and social acceptability. Doing so saves researchers costly redesigns and repeated iterations.

Scope. In this work, we present and evaluate EvaluatAR using two PETs representing *implicit* and *explicit* design categories. Implicit PETs infer bystanders' privacy preferences through the sensed context, whereas explicit PETs require bystanders to communicate their privacy preferences. For our case studies, we choose BystandAR [15], a state-of-the-art (SOTA) implicit PET that combines real-time computer vision with eye tracking, and a Cardea-inspired [56] gesture-driven explicit PET that uses active bystander input for privacy protection enforcement. Using these PETs, we present three case studies to validate EvaluatAR's capabilities: (1) controlled cross-headset replay of identical inputs to characterize device-specific privacy-performance trade-offs; (2) generalizability of EvaluatAR's workflow across explicit and implicit PET design categories; and (3) rapid prototyping via a targeted failure-case study to diagnose privacy-relevant failures in PET outputs and validate proposed modifications, yielding an improved PET variant.

Contributions. Our work makes three main contributions.

(1) We present EvaluatAR, a headset-agnostic, modular record-replay framework that enables reproducible evaluation of bystander PETs by standardizing their inputs (sensor data and visual stimuli) and logging comparable outputs, enabling low-overhead, cross-device PET evaluation.

(2) We validate EvaluatAR on three widely used, commercially available AR headsets (Microsoft HoloLens 2, Magic Leap 2, and Meta Quest 3), demonstrating reproducible cross-device evaluation across sensing pipelines and on-device compute constraints while requiring lightweight integration effort.

(3) We show how EvaluatAR supports rapid PET iteration by capturing and replaying privacy-relevant failures to compare PET variants under identical inputs, enabling a verified improvement to BystandAR, a state-of-the-art bystander PET.

Table 1: Summary of evaluation scope across surveyed visual PETs. ✗ : none; ❍ : limited; ● : extensive. Prior work evaluates usability and privacy protection more extensively, but performance evaluation is often device-specific.

System Name	Usability	Privacy	Performance
SnapMe [26]	❍	✗	✗
FaceBlock [43, 61]	✗	✗	✗
BlindSpot [44]	❍	❍	✗
PrivacyCamera [33]	✗	●	❍
I-Pic [4]	●	●	❍
Cardea [56]	●	●	❍
Respectful Cameras [52]	✗	●	❍
Your Privacy is in Your Hand [55]	●	●	❍
PrivateEye [48]	●	●	❍
WaveOff [48]	●	●	❍
BystandAR [15]	●	●	❍
MarkIt [47]	✗	❍	❍
Virtual Curtain [54]	❍	❍	●
PlaceAvoider [57]	✗	●	●
Courteous Glass [29]	✗	●	●

2 Related Work

In this section, we review PETs for visual data (§2.1), their evaluation practices (§2.2), and examine prior efforts to standardize their evaluation (§2.3).

2.1 PETs for Visual Data

Developing PETs to protect sensitive information in visual sensing pipelines has been an active area of research. Corbett et al. [15] distinguish bystander PET designs into two categories:

Implicit PETs. These systems infer privacy intent from sensor data and contextual cues without requiring bystander participation. They rely on social cues (e.g., posing for a picture) or other contextual information (e.g., distance from the camera, eye gaze direction, emotion, head pose, position in the frame) to detect bystanders and obfuscate them [17, 25]. An early example of an implicit PET is Courteous Glass [29] that uses on-device sensing (e.g., thermal imaging) to detect privacy-relevant situations.

The current SOTA implicit PET for AR headsets is BystandAR [15] as it supports AR's always-on sensing requirements without requiring bystander registration or cooperation. It runs continuously on-device and processes egocentric scene captures, performs real-time face detection on them, and distinguishes subjects and bystanders in the scene by leveraging sensor-rich cues such as eye gaze, audio, and spatial mapping. However, the system's reliance on sensor-rich, real-time inference makes it sensitive to device-specific sensing and compute pipelines, motivating the need to evaluate its cross-device feasibility, performance, and privacy-performance trade-offs. Hence, we select BystandAR as our implicit PET case-study as it captures the core properties and challenges of this category in AR.

Explicit PETs. These systems require the user or the bystander to perform some action to express their privacy preferences. While these systems provide stronger user/bystander control over what is protected, they also impose an interaction burden by requiring conscious participation, special equipment, or enrollment in advance. Prior approaches include preference registration and broadcast [4, 33, 57], visible markers or tags [44, 52], and interaction-driven controls such as gestures, tagging, or region specification [48, 54, 55].

A central challenge for explicit PETs is whether they can correctly detect an intent signal, associate it with the intended person, and update the protection state accordingly over time. Cardea [56] is a hybrid design in this space that combines contextual policy enforcement with direct privacy signaling at capture time via bystanders' hand gestures. As it captures the core technical and interaction challenges of explicit PETs in AR, we use a Cardea-inspired gesture-driven PET as our explicit case study.

Together, these categories represent the design space of PETs that our framework must support. Therefore, we instantiate one PET per category in our evaluation: BystandAR as the SOTA implicit PET and a Cardea-inspired gesture-driven explicit PET.

2.2 Evaluation of PETs for Visual Data

We surveyed the evaluations of 15 visual PETs. Our analysis identified three key evaluation dimensions that we use as an organizing lens rather than a strict taxonomy, since PETs differ in sensing pipelines, threat models, and deployment goals.

Usability. Usability refers to the perceptions of bystanders/users regarding their comfort and acceptability of a PET's outputs. Many systems, including Cardea [56], WaveOff [48], and BystandAR [15], conduct in-person user studies to assess social acceptability, user comprehension, and interaction burden. Other systems, such as SnapMe [26], use online surveys to collect perceptions at scale.

Privacy Protection. Privacy protection refers to the quantitative measures of bystander privacy protections. This dimension is commonly evaluated experimentally using annotated datasets or scenario-based tests across diverse environmental conditions. For example, PlaceAvoider [57] and Courteous Glass [29] test privacy logic across settings (e.g., public vs. private spaces) and under different detection criteria, offering broader scenario coverage.

Performance. Performance refers to system responsiveness under resource and latency constraints, often reported as FPS. While many systems report FPS, CPU/GPU usage, or energy consumption [4, 33, 48], these measurements are frequently collected on a single device, often the platform for which the PET was designed. For instance, Android-based systems such as PrivacyCamera [33] and WaveOff [48] were evaluated on a Nexus 5 smartphone, and BystandAR was evaluated solely on HoloLens2 [15]. Moreover, existing evaluations are often live and human-dependent, making it difficult to reproduce the same inputs (visual stimuli and sensor data) across trials. As a result, the literature provides limited support for *reproducible, cross-device* comparison of PET performance and privacy-performance trade-offs.

In Table 1, we code prior work's evaluation scope as none, limited, or extensive for each evaluation dimension. *Extensive* indicates an in-person user study for usability, evaluation across multiple distinct environments/conditions or annotated datasets for privacy protection, and measurements on multiple devices for performance; *limited* otherwise. Our survey suggests a consistent trend: usability and privacy protection are often evaluated with relatively extensive methodologies, whereas performance is often evaluated in a

limited manner, as evaluation pipelines are difficult to replicate. Since real-time performance can directly influence whether privacy transformations are applied reliably in practice, this fragmentation motivates standardized, cross-device evaluation pipelines that can isolate design choices and expose privacy-performance trade-offs.

2.3 Evaluation Frameworks for AR Visual PETs

A small number of research efforts aim to standardize evaluation in AR and visual sensing systems, but these primarily emphasize user-centric usability and do not provide end-to-end support for evaluating bystander PETs in terms of privacy protection and real-time performance. For instance, Choong et al. [14] and Dünser & Billinghurst [19] present structured usability evaluation approaches assessing cognitive load, interaction fidelity, and visual coherence for users. Archie [32] supports in-the-wild usability testing of AR systems, but it does not incorporate bystander privacy perspectives. Collectively, these approaches provide valuable usability insights but do not address evaluation settings where privacy is a dynamic system behavior involving bystanders and real-time sensing.

Erdélyi et al. [20] move closer to our focus by proposing an objective framework to evaluate privacy protection filters for visual data through privacy-utility trade-offs. However, their framework assumes static, offline image datasets and evaluates filters in a post-processing context. This assumption does not align with real-time, multi-sensor AR PETs where latency, tracking stability, and device performance directly affect whether privacy transformations are enforced reliably. In addition, their framework focuses on visual data alone, whereas AR bystander PETs may depend on additional sensor context (e.g., gaze and spatial cues) and must operate under sensor-driven variability.

Commercial AR SDKs and toolkits (e.g., ARCore [24], MRTK [40], Vuforia [59]) provide platform-specific utilities for recording and replaying *application sessions* (e.g., head pose and controller/hand input) to aid application debugging. However, these features do not standardize evaluation of bystander PETs across headsets: they do not support synchronized replay of the full sensor data streams a PET consumes (e.g., camera/depth with pose and gaze), nor do they provide reproducible visual stimuli and logging interfaces that enable cross-headset comparisons under the same inputs.

Overall, existing efforts show limited support for reproducible, cross-device evaluation of bystander PETs using standard stimuli.

3 Framework Design

In this section, we introduce EvaluatAR, a framework that enables low-overhead early stage bystander PET evaluation in a *reproducible, headset-agnostic* manner. We summarize the common components of visual bystander PETs (§3.1), enumerate technical challenges that arise when designing for cross-device evaluation (§3.2), and describe how EvaluatAR operationalizes controlled evaluation through its record-replay workflow (§3.3).

3.1 Components of Visual Bystander PETs

While bystander PETs vary in sensing modalities, detection models, and privacy transformations, most visual PETs follow a common runtime pattern consisting of the following components (Alg. 1):

Sensing. This component involves ingestion of egocentric sensor data streams (Alg. 1, lines 3 and 4), including camera frames and optional sensor data that a PET may consume (e.g., eye gaze, depth, audio cues, or other context).

Detection. This component entails the detection of candidate bystanders (Alg. 1, line 5) in the scene (e.g., faces or full body segmentation). This process is typically performed through detectors that produce candidate regions (e.g., bounding boxes or masks).

Bystander Decision Logic. The goal of this component is to apply PET-specific logic to determine which candidates require protection, based on detections from the Detection stage, and optionally sensor data (Alg. 1, line 7). A possible default logic for a PET is to protect all candidate bystanders.

Privacy Transformation. This step applies PET-specific privacy transformation (e.g., obfuscation through blurring or pixel masking) to protected candidates in the visual output (Alg. 1, line 9).

Algorithm 1 Generic visual bystander PET control loop.

```
1:  Parameters: Detector ∈ FaceDetection, HumanSegmentation
2:  while True do
3:      CameraFrame ← current camera frame
4:      S_t ← sensor(s) data
5:      bboxes ← Detector(CameraFrame)
6:      for each bbox ∈ bboxes do
7:          isBystander ← PET_BystanderLogic(bbox, S_t)
8:          if isBystander then
9:              PET_PrivacyTransformation(CameraFrame, bbox)
10:         end if
11:     end for
12: end while
```

3.2 Main Challenges

EvaluatAR enables low-overhead, reliable replication of experimental conditions across headsets and trials for the early-stage PET evaluation. The goals of this evaluation are to debug PET behavior, validate design choices, and profile PET's performance and privacy-performance trade-offs before investing in high-overhead human-subject studies. To support these goals, our initial design intuition for the framework was to capture scenarios as videos and replay them on a screen. Our rationale behind this design decision was that preserving these scenarios as video stimuli would eliminate the confounds inherent in existing setups requiring live human recreation (inconsistent head motion, bystander movement, or viewpoint changes, etc.). However, we observed that this approach had several technical challenges in practice.

(C1) Viewpoint Inconsistencies The generic PET components identified in §3.1 establish that visual bystander PETs operate on egocentric visual content and sensor inputs. Thus, reliably reproducing the visual scene seen by the PET is critical. If the experimenter's viewpoint changes across trials, PET may process different visual content even when the same video stimulus is replayed. Similarly, small variations in viewpoint can lead to significant differences in replayed sensor data. For example, eye gaze is cast from the wearer's head position, so small differences in the pose of the experimenter wearing a target headset can produce larger angular errors in gaze data for elements in the scene.

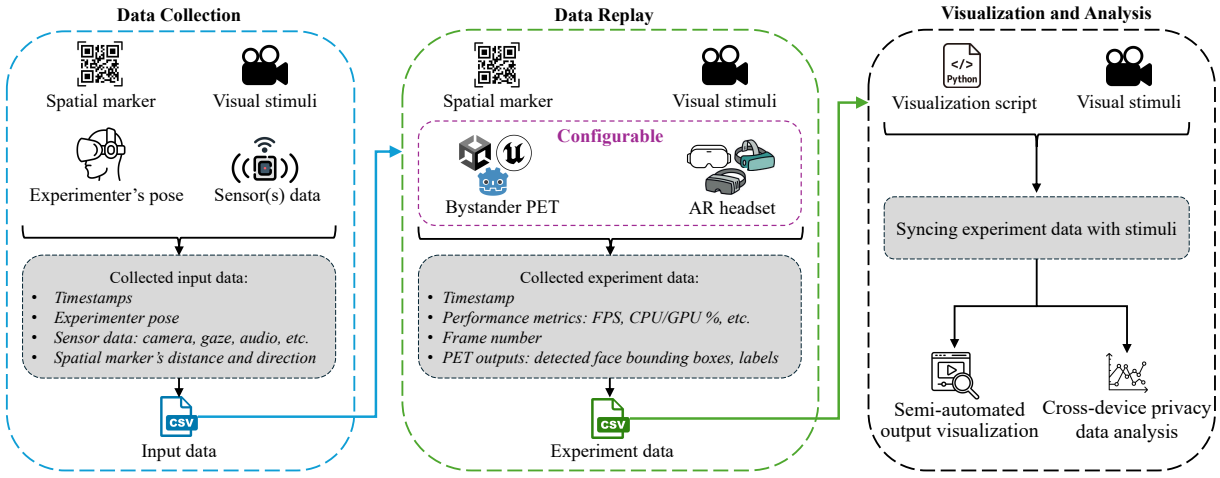


Figure 2: EvaluatAR's three-stage workflow. Data Collection records PET inputs (sensor data and visual stimuli) and the experimenter's pose relative to a spatial marker. Data Replay enables stimulus-synchronized replay of sensor data across headsets and trials, while logging performance metrics and PET outputs. Visualization and Analysis visualizes the logged experiment data, enabling cross-device performance and privacy-performance profiling of PETs under controlled conditions.

(C2) Repeated Device-Integration Effort: APIs available for exposing sensor data to applications vary across AR platforms. This variation incurs non-trivial engineering costs, as implementing sensor data capture and replay requires device-specific implementations across PETs and target headsets.

(C3) Synchronization Issue of PET Inputs: AR headsets differ in processing speed, compute capabilities, and runtime behavior. These differences impact the synchronization of replayed sensor data with its associated stimulus. For example, a headset with superior hardware may replay sensor data faster than a headset with older hardware, leading to inconsistent data replay across headsets.

(C4) Synchronization Issue of PET Outputs and Stimulus: Variations in headset execution rates and stimulus playback cause PET outputs and visual stimuli to go out of synchronization. Desynchronization complicates the precise alignment required for reliable visualization and analysis for effective early-stage PET evaluation.

3.3 Framework Workflow

The identified challenges motivate EvaluatAR's controlled design. It uses video-based stimuli for standardized scene recreation rather than relying on live participants whose behavior could vary, introducing confounds. Similarly, EvaluatAR uses a fixed-pose design for the experimenter to preserve viewpoint and to enable standardized replay of sensor data across headsets and trials. It also standardizes the PET inputs/outputs and adopts an elapsed time-based replay approach of sensor data through a common workflow to reduce device-specific engineering overheads and synchronization issues. EvaluatAR operationalizes this design in three stages: Data Collection, Data Replay, and Visualization and Analysis.

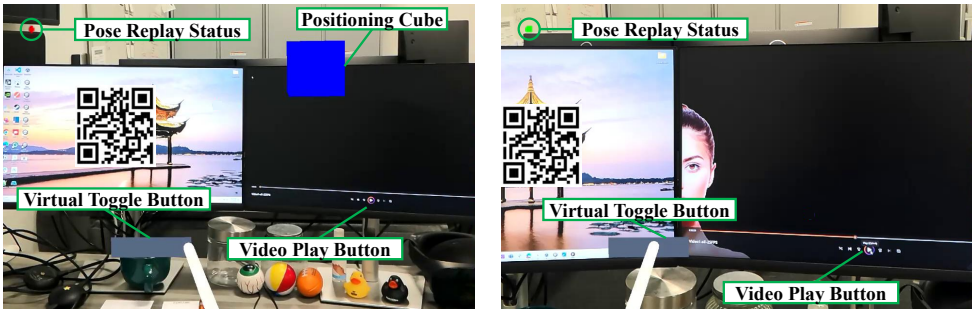
Data Collection. The goal of this stage is to capture input data for consistent experiment replication in a headset-agnostic manner. This includes sensor data and visual stimuli (visual content presented to the wearer, such as video recordings depicting bystanders and scene activity). This stage aims to record these inputs in a device-independent representation so the data captured on one headset can be replayed on another, despite the differences in sensor APIs exposed by headsets.

To achieve this goal, EvaluatAR records the experimenter's pose (Alg. 2, line 7) over time relative to a spatial marker (e.g., a QR code) placed in the environment (Alg. 2, line 6) and any additional input sensor data streams that the target PET consumes (Alg. 2, line 12), such as eye gaze, audio cues, or ultra-wideband and Bluetooth signals used for locating bystanders [3]. The spatial marker allows the experimenter to maintain a consistent spatial reference between Data Collection and Data Replay to preserve the experimenter's viewpoint across trials, addressing **(C1)**.

To avoid confounding runtime performance measurements, the visual stimuli are presented externally rather than rendered by a separate in-headset application alongside the PET. All recorded input data is stored with elapsed timestamps and is written via the unified logging hook (Alg. 2, line 26), addressing **(C2),** in a CSV file that has extensive compatibility across headsets.

Data Replay. The goal of this stage is to replay the input data to evaluate a PET under standardized conditions across headsets and trials. During replay, the headset detects the visual marker to calculate its current marker-relative pose, then uses the recorded marker-relative start pose to guide the experimenter back to the correct starting position. A lightweight alignment mechanism (e.g., visual cues) assists the experimenter in reproducing the viewpoint. Marker detection is disabled once the viewpoints align to avoid additional performance overhead caused by the framework on a PET. The framework also captures a reference FoV image at this point (enabled by storing the current camera frame via the hook in Alg. 2, line 8) to support accurate overlays in the Visualization and Analysis stage.

After alignment, the visual stimulus is played externally while EvaluatAR supplies the PET with input sensor data for replay (Alg. 2, line 10). However, as noted in **(C3)**, different headsets process frames at different rates, desynchronizing replayed input data



(a) Before alignment **(b) After alignment**

Figure 3: EvaluatAR's user interface before and after experimenter's viewpoint alignment across headsets and/or trials. The QR marker, positioning cube, and pose replay status indicator (turns from red to green when aligned) help guide the experimenter to the recorded start pose. Once aligned, the experimenter can start the trial via the virtual toggle button.

and its associated stimulus. EvaluatAR support elapsed time-based replay of inputs: at time t, the framework uses the most recent logged input data point with timestamp $\leq t$, addressing **(C3).** In parallel, EvaluatAR logs per-frame data, including elapsed time, frame number, runtime performance metrics (e.g., FPS), and PET outputs (e.g., detected regions/labels or obfuscation states), to an experiment data CSV file via a unified logging hook (Alg. 2, line 26).

In our design, the experimenter coordinates the start of input data replay and visual stimulus playback to minimize additional instrumentation (e.g., casting/remote control) that could otherwise introduce confounds.

Visualization and Analysis. This stage supports semi-automated inspection and analysis of a PET's outputs by aligning per-frame experiment data logs with the recorded visual stimulus. The experiment data CSV contains per-frame experimental data logs corresponding to the visual stimuli presented during the experiments. EvaluatAR synchronizes logged experimental data (e.g., detected regions/labels and other PET outputs) with the visual stimuli and renders them as an annotated .MP4 video. This visualization enables quantitative analysis (e.g., performance trends over time) and systematic inspection of privacy failure cases. When appropriate, it can also serve as input to downstream perception studies (e.g., bystander-facing evaluations of PET outputs via surveys).

To generate accurate overlays, EvaluatAR uses the reference FoV image captured during Data Replay to identify the on-screen stimulus boundaries (top-left and bottom-right corners). These points define the mapping from camera-space coordinates in the experiment data logs (e.g., bounding boxes) to stimulus coordinate space so overlays align with the visual content presented during evaluation. Since headset execution and stimulus playback may differ in frame rate, as noted in **(C4)**, EvaluatAR aligns experimental data logs to the visual stimulus frames using elapsed time and drops unmatched frames at the beginning of playback, producing a synchronized output suitable for both visualization and analysis, and addressing **(C4)**.

4 Framework Implementation

We implemented EvaluatAR in Unity (2022.3.12f1) and made it available at *https://github.com/SIMSB-99/EvaluatAR.git*. Our implementation uses a QR code as a shared physical marker, a virtual toggle button to start/stop logging, and a pose alignment guiding mechanism using a virtual positioning cube and a pose-status indicator (Fig. 3). The algorithm of our implementation can be found in Appendix A.

Algorithm 2 Generic visual bystander PET control loop with EvaluatAR hooks for standardized record-replay and logging.

1: **Parameters:** QRDetector, Mode $\in \{Collect, Replay\}$, Detector $\in FaceDetection, HumanSegmentation$
2: FrameNum = 0
3: **while** True **do**
4: CameraFrame ← current camera frame
5: FrameNum++
6: $Q_t \leftarrow QRDetector.Pose$
7: $H_t \leftarrow HeadsetPose$
8: $EvaluatAR.setCurrentCameraCapture(CameraFrame)$
9: **if** Mode == *Replay* **then**
10: $S_t \leftarrow EvaluatAR.getCurrentFrameData()$
11: **else**
12: $S_t \leftarrow$ sensor(s) data
13: **end if**
14: bboxes ← $Detector(CameraFrame)$
15: **for** each $bbox \in bboxes$ **do**
16: isBystander ← PET_BystanderLogic(bbox, S_t)
17: **if** isBystander **then**
18: PET_PrivacyTransformation(CameraFrame, bbox)
19: **end if**
20: **end for**
21: **if** Mode == *Replay* **then**
22: LogData ← $(FrameNum, bboxes)$
23: **else**
24: LogData ← $(FrameNum, Q_t, H_t, S_t)$
25: **end if**
26: $EvaluatAR.writeToLogsFile(LogData)$
27: **end while**

Data Collection. EvaluatAR logs one time-synchronized entry per frame in this mode until data recording is stopped by the experimenter. Each entry includes the timestamp, elapsed time since

the start of data recording, frame number, the experimenter's pose, and the headset-to-marker relative pose (encoded as a 3D vector from the headset to the detected QR code).

Data Replay. Before running a PET with this mode of EvaluatAR, the input data CSV created after Data Collection mode is added to the on-device storage of the target headset. When the PET runs, EvaluatAR reconstructs the trial's starting pose from the stored relative pose, and renders a virtual positioning cube at that location. The experimenter aligns their head pose with this cube until the pose-status indicator confirms alignment (Fig. 3), after which EvaluatAR disables QR detection for the remainder of the trial. EvaluatAR also captures an RGB camera frame to support later mapping between camera space and the on-screen stimulus during Visualization and Analysis. The experimenter then begins the trial by pressing the toggle button and playing the visual stimulus video simultaneously.

Visualization and Analysis. We implement this stage as an offline Python script that processes collected experiment data, aligns the recorded headset FoV to the stimulus screen (annotated by the experimenter), and generates time-synchronized overlays of PET outputs on the stimulus for inspection and analysis.

We instantiate this implementation across implicit and explicit PET designs and multiple headsets in our case studies (§5). To integrate a PET, we (1) expose the required inputs through lightweight getter functions in the PET's main Unity script and (2) attach the EvaluatAR script to a Unity GameObject to pass inputs and PET outputs into the framework's logging interface. As our supported headsets already implement the remaining workflow infrastructure (as described above), adding a PET mainly requires integrating EvaluatAR's input/output interfaces and validating the intended functionality. Based on the development time it took to integrate the MQ3 headset and Cardea-inspired explicit PET, we estimate 2-6 hours (one developer) for integration, plus 1-2 hours of validation per supported headset, totaling 3-8 hours to integrate and execute a new PET across all three headsets. These are informed engineering estimates rather than a controlled time study.

5 Case Studies

We conduct a case-study-based evaluation to validate whether EvaluatAR supports representative early-stage PET evaluation goals. These goals include investigating PET's feasibility across headsets, robustness under controlled conditions, profiling its performance and privacy-performance trade-offs across supported devices, and debugging via replayable failure cases to support rapid iteration. This technical validation serves as a precursor to standalone privacy and usability evaluations, and high-overhead human-subject studies centered on ecological validity, user experience, and social acceptability, which are beyond the scope of this work.

We design three case studies with the following goals that demonstrate EvaluatAR's ability to support early-stage PET evaluation:

- **G1:** Enabling reproducible, headset-agnostic evaluation of a bystander PET's performance.
- **G2:** Supporting evaluation of privacy-performance trade-offs across implicit and explicit PET design categories using the same end-to-end workflow.
- **G3:** Enabling PET design validation and iterative debugging by comparing proposed modifications for privacy-relevant failures exhibited by a PET under standardized replay of edge cases.

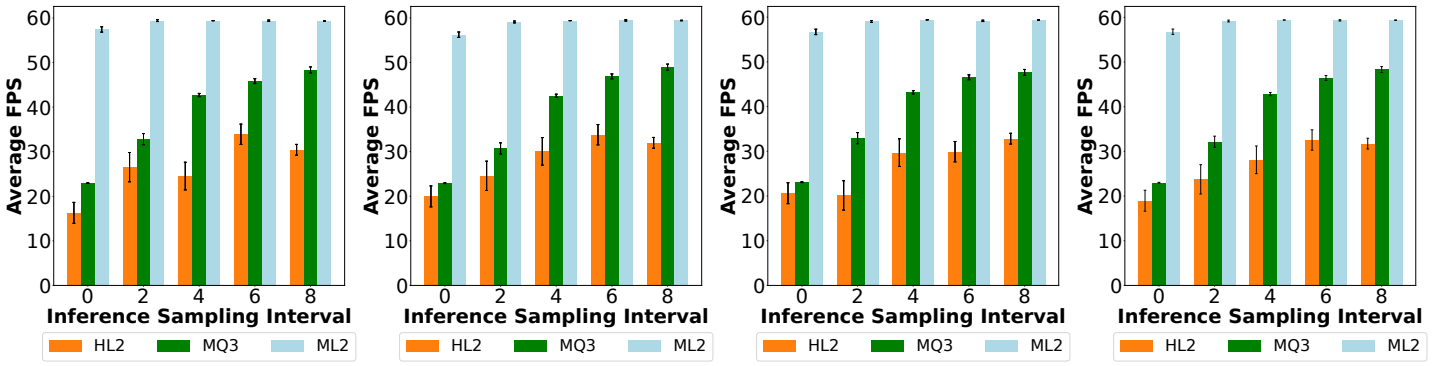
We use three commercially available headsets in our case studies: HoloLens 2 (HL2), Magic Leap 2 (ML2), and Meta Quest 3 (MQ3). These headsets span the two dominant AR display systems: HL2 and ML2 are optical see-through (OST) headsets, where users view the real world directly through transparent displays, whereas MQ3 is a video see-through (VST) headset, where the real world is mediated through a camera passthrough pipeline. These headsets also represent distinct on-device compute profiles: ML2 uses a quad-core Zen2 CPU (4×3.92 GHz, 8 threads), MQ3 uses an octa-core Kryo CPU (1×3.19 GHz, 4×2.8 GHz, 3×2.0 GHz), and HL2 uses an older octa-core Kryo385 CPU (4×2.96 GHz, 4×1.8 GHz) [58]. Together, these headsets represent different sensing pipelines and compute capabilities, allowing our case studies to demonstrate that EvaluatAR supports reproducible early-stage evaluation across a broad range of modern AR headsets.

To reflect the breadth of bystander PET designs in prior work, we evaluate one implicit sensor-rich PET (BystandAR [15]) and one explicit gesture-driven PET (inspired by Cardea [56] and adapted for AR headsets). Using these two PETs lets our case studies demonstrate that the same end-to-end workflow can be applied across the two dominant PET design categories. In addition to the per-frame trial logs (§3.3), we record PET-specific outputs needed for analysis. For BystandAR, we log per-frame face detections and associated PET state (e.g., IDs, regions/predictions, and subject/bystander labels), and for the Cardea-inspired PET, we log per-module processing times, recognized gestures with associated face IDs, and per-face obfuscation state over time. We validate experimental trial integrity using the logged pose alignment status and expected replay progression; invalid trials are discarded and re-run.

Across our case studies, we draw the video-based visual stimuli from three sources: (1) commercially licensed stock footage, (2) author-recorded clips , and (3) synthetic sequences for controlled edge cases. For Case Studys 1 and 2, we use recordings that capture diverse real-world scenarios relevant to bystander privacy, including coworkers in a laboratory, newscasting and press conference settings, friends cooking in a kitchen, people walking on a busy street, and virtual meeting-like scenes. These scenarios vary in motion, crowding, camera-to-subject distance, and occlusion patterns, allowing us to test PET behavior under representative scene diversity while retaining controlled replay. For Case Study 3, we use synthetic videos where precise overlap and crossing trajectories are required to reproduce privacy-relevant failure modes. This use of recorded and synthetic stimuli is an evaluation-scope choice to enable early-stage PET evaluation. Future instantiations can swap in more ecologically valid recordings without changing EvaluatAR's workflow. Additionally, as some stimuli are commercially licensed, we do not redistribute raw stock footage. Instead, we have made EvaluatAR's implementation available so others can reproduce the workflow using their own stimulus set.

5.1 Case Study 1

This case study is designed to achieve **G1** by demonstrating that EvaluatAR can produce reproducible and comparable performance measurements for the same bystander PET when replayed under identical inputs on different headsets.



(a) Minimal head motion (static) (b) Gradual head motion (slow) (c) Rapid head motion (fast) (d) All videos of varying motions

Figure 4: Average FPS across inference sampling intervals (0 represents per-frame inference) for the three headsets across static, slow, fast, and averaged motion conditions in Experiment 1A. Larger sampling intervals increase PET's performance. ML2 consistently achieves the highest performance, followed by MQ3 and HL2.

We instantiate EvaluatAR with BystandAR [15], a SOTA implicit bystander PET for AR headsets, evaluated only on HL2 originally. BystandAR works by continuously processing the egocentric camera stream to detect candidate bystanders (face detection) and combines these detections with additional sensor data (notably eye gaze, audio cues, and depth data) to implement its bystander decision logic. Intuitively, BystandAR uses gaze-derived social cues to distinguish subjects (people the wearer is engaged with) from bystanders, and then applies a privacy transformation (blacking out associated pixel regions) to bystanders in the output stream. To remain real-time on device, BystandAR exposes an *inference sampling interval* that skips a specified number of frames between inferences, directly impacting compute cost and real-time responsiveness. The detailed algorithm with EvaluatAR integration can be found in Appendix B.

We select BystandAR as the target PET for this case study because it represents a realistic, on-device bystander protection workload with two properties that make cross-device evaluation well-defined and interpretable. First, BystandAR has a published configuration knob (i.e., inference sampling interval) that systematically trades inference frequency for compute cost, allowing us to sweep a PET configuration range while holding the PET's core logic constant. Second, BystandAR's per-frame workload increases with the number of candidate bystanders visible in the scene: more people in view increase the number of detections that must be processed and maintained over time (e.g., tracking/association and per-person privacy transformations). These properties allow us to examine two common and practical sources of performance variation under standardized replay: (1) a PET-side configuration knob (§5.1.1) and (2) visual workload variation via candidate load (§5.1.2).

5.1.1 Experiment 1A: PET configuration knob. We design this experiment with inference sampling interval and headset as independent variables (IVs). We evaluate five levels of inference sampling interval IV: $interval \in \{0, 1, 2, 4, 8\}$, where $interval = 0$ runs inference every frame and larger values skip more frames. We cap interval values at 8 to match the configuration range evaluated in the original BystandAR paper so that our evaluation remains within BystandAR's published operating range, as our goal in this case study is to measure cross-device performance under standardized replay, not to redesign BystandAR's logic. The three headsets (HL2, MQ3, and ML2) represent the three levels of headset IV. The dependent variable (DV) is performance, measured via FPS.

Procedure. We use three videos drawn from commercially licensed stock footage as standardized visual stimuli for all trials. Each video contains a single visible person whose motion varies: minimal head motion (static), gradual head motion (slow), and rapid head motion (fast). Each headset is tested for all five *interval* values across all three videos, resulting in a total of 75 trials.

Results. Across all three videos of varying motion (Fig. 4a, 4b, and 4c), increasing the sampling interval yields a clear increase in FPS on every headset. This pattern is consistent across the static, slow, and fast videos, indicating that the impact of the configuration knob is stable across different motions under standardized replay.

The results also expose a stable cross-headset ordering under the same replayed inputs and configurations: ML2 sustains the highest FPS across the tested interval values, MQ3 achieves moderate FPS, and HL2 remains substantially slower even at larger interval values. Since EvaluatAR replays the same inputs and visual stimuli on each headset, these differences can be attributed to device/runtime constraints, showcasing that EvaluatAR enables reproducible cross-headset comparison of the same PET.

Finally, we use these results to select a per-headset operating configuration for subsequent experiments. We define the best interval for a headset as the smallest interval value at which the FPS reaches the performance plateau (i.e., further increases in interval yield only marginal gains), thereby avoiding unnecessary skipping of inference while still operating near the performance knee. Using this criterion, we select $interval = 8$ for HL2, $interval = 4$ for MQ3, and $interval = 2$ for ML2.

5.1.2 Experiment 1B: Candidate bystander load. This experiment evaluates how PET performance scales as the number of candidate bystanders in the visual stimulus increases under standardized replay, as this scaling behavior was not examined in the original

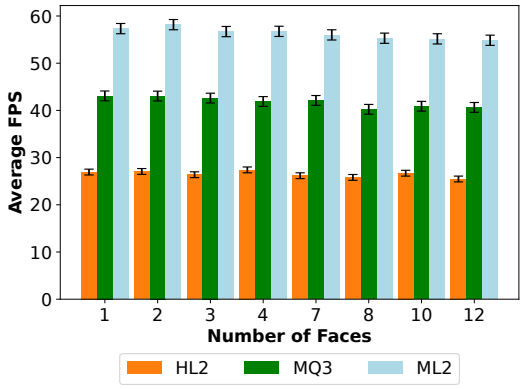


Figure 5: Average FPS across headsets against varying candidate bystander load in Experiment 1B. Headsets differ in baseline performance (ML2 performs the best, followed by MQ3 and HL2) and how PET's performance degrades (MQ3 shows the steepest decline, followed by ML2 and HL2).

BystandAR evaluation. We designed this experiment with the candidate bystander load and headset as IVs. We operationalize candidate load as the number of bystanders visible in the scene and evaluate nine load levels: $load \in \{1, 2, 3, 4, 5, 7, 8, 10, 12\}$. The three headsets represent the three levels of headset IV. The DV is performance, measured via FPS.

Procedure. We use one segmented stimulus video to standardize the visual workload across trials. The video is composed of nine sequential scenes that depict the candidate load levels, separated by blank screens for segmentation. Lower-load scenes are extracted from stock footage, while the highest-load scenes are composited from these licensed clips to preserve facial detail when many bystanders are present. We replay this video for each headset, resulting in a total of three trials.

Results. Fig. 5 further reveals that headsets differ not only in their baseline FPS, but also in how quickly they lose performance as load increases. HL2 starts from the lowest FPS and shows comparatively limited additional decline with increasing load, suggesting it reaches a constrained operating point early. MQ3 exhibits the steepest decline as load increases, indicating tighter compute margins under higher candidate loads.

Taken together, these experiments demonstrate that EvaluatAR enables reproducible, cross-headset performance evaluation of the same PET under standardized replay. Since each experiment replays identical inputs and visual stimuli, the observed trends reflect the intended experimental variations for each experiment rather than being confounded by differences in experimenter behavior, scene content, or experimental procedure across trials. Under these reproducible conditions, EvaluatAR yields consistent within-headset trends when sweeping a PET configuration knob and scaling visual workload, while also revealing stable cross-headset differences under the same conditions. This combination of repeatable trends within a headset and directly comparable measurements across headsets provides empirical support for **G1** and illustrates how EvaluatAR can be used to identify feasible operating configurations for a PET on each device under controlled, replayed inputs.

5.2 Case Study 2

Case Study 1 demonstrated EvaluatAR's standardized workflow on an implicit PET. We design Case Study 2 to advance **G2** by showing that the same end-to-end workflow can also be used to evaluate privacy-performance trade-offs of an explicit PET design while using PET-specific measures appropriate for its trigger mechanism.

We instantiate EvaluatAR with a Cardea-inspired [56] gesture-driven explicit PET adapted for AR headsets. At runtime, the PET processes the egocentric camera stream and runs a multi-stage on-device perception pipeline (face detection, hand detection, and gesture recognition). It then associates a detected hand to a face using spatial proximity and applies a privacy transformation (Gaussian blur) based on the recognized gesture. It also tracks bystanders across frames by matching face detections with the highest overlap. Following Cardea's definition, all people in view are treated as bystanders: by default, bystanders are not obfuscated; a bystander can explicitly request protection (opt-in) via an Open Palm gesture and revoke protection (opt-out) via a Victory (forming a V with index and middle finger) gesture. The detailed algorithm with EvaluatAR integration can be found in Appendix C.

We select this PET because it represents an interaction-driven design of bystander PETs where privacy enforcement depends not only on per-frame performance, but also on whether the system correctly translates observable intent into the intended privacy state over time. This makes it a complementary workload to Case Study 1: under standardized replay, EvaluatAR can capture both performance and intent-to-enforcement behavior using the same workflow, enabling a direct demonstration of generalizability across PET design categories (**G2**).

5.2.1 *Experiment 2: Model stack configuration and candidate load.*

We design this experiment to stress two practical factors that shape explicit PET behavior under real-time constraints: (1) the compute-accuracy trade-off in the PET's perception pipeline and (2) the difficulty of maintaining correct hand-face association when multiple bystanders are present. We therefore use model stack configuration , candidate bystander load, and headset as IVs. Model stack has two levels: $stack \in \{high, low\}$ where *high* represents the high-precision stack that uses the base versions of the face, hand, and gesture models, and *low* represents the low-precision stack that replaces them with INT8 (quantized) variants. Candidate load has two levels, $load \in \{1, 2\}$, corresponding to one-bystander versus two-bystander scenes. We evaluate on two headsets (ML2 and MQ3). We attempted to include HL2, but the gesture pipeline exhibited unstable runtime behavior (very low FPS and frequent crashes), preventing consistent Data Replay runs; we therefore treat HL2 as a feasibility boundary for this particular PET and focus the comparisons across ML2 and MQ3.

We report two classes of DVs. First, we report performance via per-frame processing time (ms) and its module-level breakdown (face, hand, gesture, blur). Second, we measure intent-to-enforcement behavior via (1) reliability: whether each scripted opt-in/opt-out intent event within the recorded visual stimuli yields the correct per-face obfuscation state, and (2) responsiveness via intent-frame pipeline cost proxy by summing the per-module times required to produce and apply the obfuscation decision.

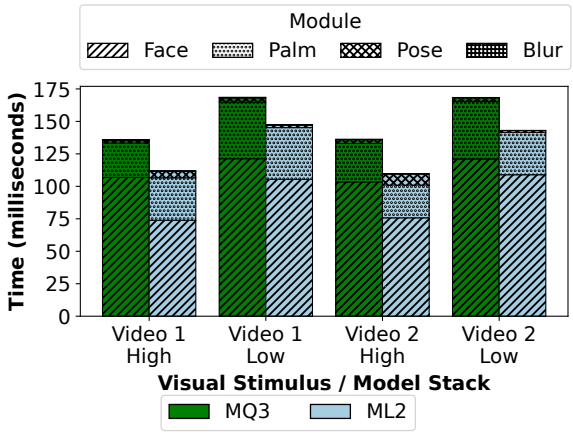


Figure 6: Module-level processing time for the explicit PET pipeline. The face detector takes the most time, dominating the overall pipeline cost. Interestingly, the low-precision stack does not reduce end-to-end processing time.

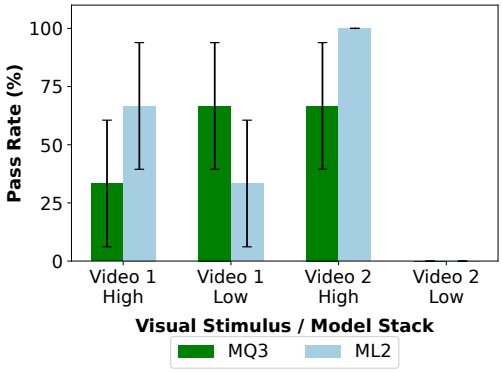


Figure 7: Intent-to-enforcement correctness of explicit PET. Correctness is more stable on ML2 and degrades across headsets under demanding configurations, especially with two bystanders (Video 2) and the low-precision stack (Low), where the Video 2 + Low condition reaches 0% pass rate.

Procedure. The visual stimuli for this experiment were recorded by the authors. We recorded two standardized replay videos with scripted intent events so that bystander intent is controlled and repeatable under identical visual stimuli. Video 1 contains a single bystander performing repeated Open Palm (opt-in) and Victory (opt-out) gestures, interleaved with non-gesture periods to test stability. Video 2 increases candidate load by including two bystanders, while only one performs the scripted gesture sequence to stress hand-face association under higher candidate load. For each headset, we replay each video under both model stack configurations and repeat each condition three times, resulting in 24 total trials.

Results. Under identical replayed inputs, ML2 consistently sustains lower total per-frame processing time (higher FPS) than MQ3 (lower FPS) across both videos and both model stacks (Fig. 6). This mirrors the stable cross-headset ordering observed in Case Study 1 (§5.1), where ML2 maintained the highest FPS across headsets. However, the absolute operating performance remains much lower (ML2: 7 FPS, MQ3: 5.5 FPS) because the explicit PET executes a multi-stage detection pipeline on-device (face, hand, gesture, and obfuscation) on every frame. Moreover, the configurations with higher per-frame cost (MQ3 and the low-precision stack) also incur higher intent-to-enforcement processing cost, meaning the PET reacts more slowly to opt-in/opt-out events under these settings.

Notably, the low-precision stack does *not* improve performance; instead, it increases the per-frame time (FPS reduction) on both headsets for both videos (Fig. 6). EvaluatAR enables us to diagnose this counterintuitive result under standardized replay: since the visual stimulus and sensor inputs are held constant across runs, EvaluatAR's module-level logs let us localize where the slowdown occurs rather than attributing experimental confounds. The module-level timing breakdown (Fig. 6) shows that the face detector dominates the per-frame budget in all conditions and that the low-precision stack increases time in the heaviest stages, yielding a higher end-to-end per-frame cost. This behavior is consistent with practical limitations of quantized execution in general-purpose inference backends (including OpenCV DNN): when optimized INT8 kernels and fusion are unavailable for the target CPU/operator set, quantized inference can incur additional conversion and data-movement overhead and may negate expected speedups [2, 9]. Since our explicit PET uses OpenCV's DNN (via OpenCV-for-Unity) on-device, we attribute the observed performance regressions to backend/kernel support.

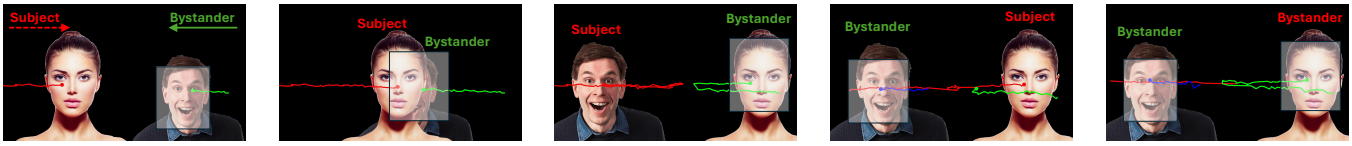
Additionally, we find that the candidate load affects this explicit PET differently than the implicit BystandAR (§5.1.2). While moving from one to two bystanders does not strongly separate FPS within a headset, it instead stresses intent-to-enforcement behavior. Fig. 7 shows that correctness becomes less stable under higher load and under the low-precision stack, including a configuration where intended protection toggles fail in the two-bystander video.

These results demonstrate **G2**: using EvaluatAR's same end-to-end workflow as Case Study 1, EvaluatAR supports a standardized evaluation of PETs across implicit and explicit designs by producing repeatable cross-headset performance measurements and design-appropriate PET outputs.

5.3 Case Study 3

Case Studies 1 and 2 established that EvaluatAR's standardized workflow can produce repeatable measurements across headsets for both implicit and explicit PET designs under identical replayed inputs. In Case Study 3, we advance **G3** by demonstrating how EvaluatAR enables PET design validation and iterative debugging by comparing proposed modifications exhibited by a PET under standardized replay of privacy-relevant edge cases. Such cases present a key barrier in PET development: many failures occur only under brief occlusions, rapid motion, or tight spatial interactions between candidates. These conditions are difficult to recreate reliably in live trials and, therefore, difficult to debug or iterate on systematically.

We return to BystandAR for this case study because it provides a suitable baseline for demonstrating **G3**. From Case Study 1, BystandAR's pipeline executes stably and near real-time on modern headsets (especially ML2), which makes it feasible to run many controlled replays and attribute differences in outcomes to the intended algorithmic change rather than to unstable execution. In contrast,



(a) Before overlap correct tracking **(b) During overlap occluded face lost** **(c) After overlap (F_s) detections swapped** **(d) After overlap (F_l) occluded face re-detected** **(e) After overlap (F_d) one reassigned, one misassigned**

Figure 8: Overlap outcomes for crossing faces: one "Before" and "During" panel, followed by three representative "After" cases for the observed failures. In F_s, detections are swapped, causing loss of privacy and utility. In F_l, the occluded face is re-detected as a new identity, breaking bystander protection. In F_d, one face is reassigned while the other is misassigned, resulting in the loss of privacy, utility, or both, depending on the incorrect association. These cases show why overlap and occlusion are privacy-relevant edge cases for association logic. The bystander's face is occluded by the translucent box.

Case Study 2 showed that our OpenCV DNN-based gesture pipeline operates at single-digit FPS and is unstable on HL2; under such performance constraints, iteration is dominated by end-to-end feasibility and backend/runtime bottlenecks rather than by logic-level robustness. Therefore, we focus on BystandAR in this case study, which represents a mature implicit SOTA PET where standardized replay can be used to isolate and improve privacy-relevant failures.

5.3.1 Experiment 3: Face association failures and proposed modifications. A privacy-preserving bystander PET must maintain a consistent mapping between each physical person and the PET's internal identity/state over time; otherwise, the PET can obfuscate the wrong person (utility loss) or expose the person who should be protected (privacy loss). During preliminary runs of BystandAR under dynamic multi-person scenes (§5.1.2), we observed these failure cases when faces overlap or cross under occlusion within the visual stimuli. These scenarios stress BystandAR's face association logic, which links new detections to previously tracked candidates. The baseline BystandAR's association logic uses Unity's `Physics.OverlapBox` to find overlapping facial detections across frames and select the first overlapping match to represent the same face across frames. This logic misassociates identities when faces overlap or when a face disappears briefly during occlusion.

We design this experiment around two IVs that together capture the iteration problem in **G3**. The first is the edge-case visual stimulus with three levels: *failure* ∈ {overlapping faces, slow crossing faces, fast crossing faces}. The second IV is the association logic with five levels: *association* ∈ {baseline, Naive Predicted Position (NPP), Closest Depth (CD), Kalman Predicted Position (KPP), Hybrid (KPP+CD)}. Apart from the baseline association logic, the other four logics represent our proposed modifications for the identified privacy-relevant association failures. NPP and KPP predict each detected face's next location and match new detections to predicted positions using 3D Cartesian distance; KPP uses a Kalman filter to improve temporal stability and reduce sensitivity to jitter over NPP's naive approach of predicting position using the current and previous known locations. CD resolves overlaps by using depth estimates from BystandAR's 2D-3D mapping to prefer the closest consistent candidate when bounding volumes overlap. The hybrid (KPP+CD) combines KPP and CD via a weighted score to handle both motion continuity and depth ordering. The algorithms for each modification can be found in Appendix B.

The DV is the robustness of identity/state continuity under the replayed stimulus, reported as per-trial pass/fail and summarized across repetitions. We count a trial as a pass if the PET preserves a correct identity-to-privacy mapping across the scenario (including cases where tracking recovers without identity swap), and as a fail otherwise. We further use EvaluatAR's visual overlays to categorize failures into the three observable modes shown in Fig. 8: swapped identities (F_s), lost-and-recreated identity (F_l), and drift/misassignment after occlusion (F_d).

Procedure. We use three standardized stimulus videos, one for each level of the edge case IV, constructed from commercially licensed stock footage to produce controlled overlap/crossing occlusions. We run all conditions on ML2 to minimize hardware confounds and because Case Study 1 showed ML2 exhibits the most

Table 2: Case Study 3 results for the baseline and four BystandAR modifications across crossing and overlapping faces scenarios. KPP is the only modification that eliminates all observed privacy-relevant association failures across all scenarios.

BystandAR Variants	Overlapping Faces		Crossing Faces (Slow)		Crossing Faces (Fast)	
	Pass	Fail	Pass	Fail	Pass	Fail
Baseline (unmodified)	4 (3 P_s, 1 P_r)	6 (6 F_s)	6 (1 P_s, 5 P_r)	4 (4 F_s)	5 (5 P_r)	5 (4 F_s, 1 F_l)
Naive Predicted Position (NPP)	2 (1 P_s, 1 P_r)	8 (8 F_s)	3 (3 P_s)	7 (7 F_s)	6 (5 P_s, 1 P_r)	4 (3 F_s, 1 F_d)
Closest Depth (CD)	7 (6 P_s, 1 P_r)	3 (2 F_s, 1 F_l)	1 (1 P_s)	9 (9 F_s)	2 (2 P_s)	8 (6 F_s, 2 F_d)
Kalman Predicted Position (KPP)	**10** (9 P_s, 1 P_r)	**0**	**10** (9 P_s, 1 P_r)	**0**	**10** (9 P_s, 1 P_r)	**0**
Hybrid (KPP + CD)	**10** (10 P_s)	**0**	2 (2 P_s)	8 (8 F_s)	**10** (10 P_s)	**0**

stable high-performance execution for BystandAR, making it suitable for repeated iteration runs. For each video, we replay the same recorded inputs and visual stimulus while swapping only the association logic inside the PET, repeating each logic ten times per video, resulting in a total of 150 trials.

Results. Table 2 summarizes outcomes across all association logics and edge cases. The baseline association fails frequently under overlapping and crossing faces, with failures dominated by identity swaps under overlap and by loss/reassignment under occlusion in crossing scenes. CD improves overlap handling (more passes under overlap than the baseline), but performs poorly under crossing faces where depth ordering is insufficient to preserve identity through occlusion. NPP does not consistently improve robustness, suggesting that naive motion continuation is too unstable under these occlusions. In contrast, KPP achieves perfect robustness across all three scenarios (10/10 passes in overlap, slow crossing, and fast crossing), indicating that stabilizing predicted motion with a Kalman filter is sufficient to resolve both overlap and occlusion cases. The hybrid strategy matches KPP under overlap and fast crossing but degrades under slow crossing, illustrating why controlled replay is valuable: two strategies that appear similarly effective in one edge case can diverge sharply under a slightly different motion regime.

Beyond aggregate pass rates, EvaluatAR's synchronized overlays make the failure mechanisms directly inspectable and comparable across proposed modifications (Fig. 8). This is the practical iteration benefit enabled by standardized replay: once an edge case is captured in a stimulus, we can repeatedly reproduce it, localize the failure mode, and evaluate proposed modifications under identical conditions without re-running live trials or introducing variability from repeated enactments.

Taken together, these results demonstrate **G3**. EvaluatAR makes privacy-relevant edge cases replayable and therefore testable as controlled inputs, enabling PET design validation and rapid iteration by swapping a single PET component (association logic) and comparing fixes under identical replayed stimuli. As the stimulus and replayed inputs are held constant across repetitions and algorithm variants, improvements in robustness can be attributed to the candidate fix itself rather than to uncontrolled differences in experimental execution.

6 Discussion

In this section, we discuss EvaluatAR capabilities demonstrated by our case studies (§5) and what their results imply for evaluating bystander PETs across headsets and PET designs. We also identify limitations that inform future work.

6.1 Case Study 1

This case study demonstrates that EvaluatAR can evaluate the same bystander PET under the same replayed inputs across different headsets. This capability is important because headsets differ in sensing pipelines and compute resources; without controlled replay, cross-device comparisons can be confounded by variation in how experimental trials are executed.

Beyond showing feasibility, the results highlight what cross-device reporting should capture when evaluating bystander PETs as real-time systems. In our setup, we vary candidate load and quantify how performance changes under standardized inputs. This enables a consistent characterization of whether a PET stays near real-time on each headset and how sensitive it is to increasing candidate load, using a shared logging format and analysis pipeline.

This case study also clarifies the practical value of EvaluatAR's workflow, fulfilling **G1**. The Data Collection and Data Replay stages provide a repeatable procedure with experimenter alignment, while the Visualization and Analysis stage produces synchronized experiment data and its overlays that make differences across headsets inspectable rather than anecdotal. To reduce the risk that EvaluatAR inflates measured performance, our replay process disables any additional processing costs (visual marker detection once alignment is achieved), so the reported measurements better reflect the PET pipeline without being confounded by the framework.

6.2 Case Study 2

This case study demonstrates EvaluatAR's ability to evaluate an explicit, gesture-driven bystander PET via the same workflow used to evaluate an implicit PET (§5.1). Unlike implicit PETs, where protection decisions are derived continuously from sensed context, explicit PETs introduce discrete intent events (e.g., opt-in/opt-out gestures) that command state changes. Therefore, evaluation must capture not only system performance, but also whether intent events are (i) recognized, (ii) associated with the intended face candidate, and (iii) reflected as the correct protection state transition over time, including the intent-to-enforcement delay.

Hence, evaluation of explicit PETs additionally requires intent-to-state correctness and intent-to-enforcement responsiveness. This case study shows that EvaluatAR supports these measurements within a single end-to-end pipeline. During replay, EvaluatAR provides standardized stimuli and repeatable inputs, allowing the explicit PET to run consistently across trials. During analysis, we use logged outputs (recognized gestures, gesture–face associations, and each candidate's protection state over time) to compute correctness and responsiveness. Together, these measures quantify whether intent is applied to the intended candidate and how quickly the system reacts.

A key result from this case study is that the low-precision (INT8) model stack does not improve performance; instead, it reduces FPS on each headset in our experiment (§5.2.1). This outcome is counter-intuitive if one assumes quantization reliably reduces inference cost. EvaluatAR helps explain this effect because it logs module-level timings in addition to end-to-end FPS. Under standardized replay, the runtime breakdown shows that the face detector dominates the per-frame budget, and the low-precision stack increases (rather than decreases) time in the heaviest stages, yielding a higher end-to-end per-frame PET cost. This behavior is consistent with practical limitations of quantized execution in general-purpose inference backends (including OpenCV's DNN module): if the backend cannot fuse quantization/dequantization patterns or lacks optimized INT8 kernels for the target CPU and operator set, quantized inference can incur extra conversion and data-movement overhead and may even fall back to higher-precision implementations, eliminating the expected speedup. Hence, the high-precision model stack in our experiment for this case study results in better performance.

Taken together with Case Study 1, this case study completes **G2**: EvaluatAR standardizes the evaluation workflow across PET designs, while allowing the dependent variables to differ based on whether the PET is implicit or explicit.

6.3 Case Study 3

This case study fulfills **G3** by demonstrating how EvaluatAR supports PET design validation and iterative debugging by capturing challenging situations once and replaying them to compare PET variants under identical inputs. Many privacy-relevant failures occur briefly (e.g., when candidates overlap or move quickly) and are difficult to reproduce consistently with ad-hoc live experimental trials. By converting such a failure into a replayable trial, EvaluatAR enables more grounded diagnosis of privacy-relevant failures and more reliable comparison of their fixes.

In this case study, Data Replay allows us to test alternative association logic on the same captured inputs and to attribute changes in outcomes to the PET modification rather than to differences in trial execution. The synchronized overlays further help interpret why a failure occurred by making state continuity and identity association issues visible during replay, connecting quantitative outcomes to observable failure modes.

This case study shows that EvaluatAR is not only an evaluation harness, but also a practical mechanism for improving SOTA PETs. Using EvaluatAR, we were able to implement and validate a stronger PET variant of the current SOTA implicit PET (BystandAR) by demonstrating clearer state continuity under the same captured failure case, supported by the framework's synchronized logs and overlays. This ability to move from observing a failure to shipping a verified improvement is a central advantage of EvaluatAR, and it is difficult to achieve with prior ad-hoc evaluation approaches that lack reproducability.

6.4 Ethical Considerations

EvaluatAR has several ethical implications for PET research. It supports studying privacy-sensitive scenarios in an ethically responsible manner via video recording-based replay instead of repeatedly exposing participants to such environments in early-stage evaluations. Moreover, as the framework supports PET design validation, debugging, and privacy-performance profiling under controlled conditions, researchers can identify and fix flaws in their PETs before moving to later-stage human-subject studies. Hence, EvaluatAR protects participants' privacy by limiting their exposure to immature PET designs. In addition, EvaluatAR's responsible usage depends on working with consented, licensed, or synthetic stimuli, and on careful collection, storage, and sharing of replayable visual data that may contain identifiable bystanders or sensitive contextual information. Hence, we do not redistribute raw scenario videos. Instead, we provide the framework code, analysis scripts, and share the visual stimuli sources for obtaining or recreating the stimuli under the appropriate consent and licensing terms.

6.5 Limitations and Future Work

Our evaluation has several limitations that motivate future work.

EvaluatAR's pipeline is not restricted to visual stimuli and can log and replay additional modalities (including audio). However, our evaluation of the framework focuses only on vision-based bystander PETs. Future work can broaden the scope of this evaluation by instantiating and benchmarking audio-driven (or multimodal) PETs within the same record-replay workflow using realistic acoustic conditions (e.g., crowd noise) alongside the visual stimulus.

EvaluatAR's record-replay workflow prioritizes controlled replication of evaluation conditions to meet the goals of early-stage PET evaluation. The trade-off of this controlled evaluation is that replayed stimuli do not capture social dynamics present with live bystanders (e.g., conversational flow, or behavior changes in response to privacy cues). EvaluatAR therefore complements later-stage human-subject evaluations focusing on ecological validity, user experience, and social acceptability.

This work focuses on performance and privacy-performance trade-offs in early-stage PET evaluation, instead of standalone privacy-protection or usability evaluation. However, EvaluatAR's design can support both in future work. The framework's outputs can log arbitrary PET output data in a standardized format, including obfuscation states and bystander labels for each frame. In our case studies, we already log privacy-relevant PET outputs, including per-frame obfuscation states and bystander labels for each detection, to support the privacy-related analyses in Case Studies 2 and 3. Future studies can extend these logged outputs to conduct in-depth privacy-protection analysis using more comprehensive or task-specific protection metrics. Similarly, the synchronized overlays of PET outputs on the visual stimuli generated by the framework can support usability evaluation of PET behavior. For example, large-scale crowd-sourced surveys of participants' perceptions of the PET outputs could be enabled using the framework's visualizations, similar to I-Pic's usability evaluation via an online survey [4].

Lastly, our experiments did not include AR content rendering. In real-world conditions, headsets render AR content, such as ubiquitous glanceable interfaces [36, 37], alongside the PET, which may affect performance. Our framework is capable of analyzing performance alongside standard rendering workloads. Future studies can validate PET behavior under rendering workloads (e.g., peripheral interfaces and gaming).

7 Conclusion

Reproducible cross-device evaluation of behavior, performance, and privacy-performance trade-offs of visual bystander PETs for AR headsets is challenging due to limitations of the existing PET evaluation pipeline. We presented EvaluatAR, a framework that enables low-overhead early-stage evaluation of bystander PETs across devices by standardizing (1) PET input/output, (2) visual stimuli, and (3) the record-replay of PET-consumed sensor data. Across three case studies on HL2, ML2, and MQ3, we demonstrated that EvaluatAR produces comparable cross-device performance measurements for a PET under controlled conditions, supports both implicit and explicit PET designs, and enables rapid prototyping by replaying privacy-relevant edge cases to validate proposed modifications, demonstrating an improved variant over a SOTA PET. These results show how EvaluatAR supports low-overhead, repeatable design validation, debugging, and privacy-performance

profiling across devices and PET designs, helping move PET development beyond ad-hoc, device-specific evaluation toward a more reproducible workflow for emerging AR systems.

Acknowledgments

The authors acknowledge support from the Commonwealth Cyber Initiative Southwest Virginia Node, the National Science Foundation under Award No. CNS-2350116, and the Center for Human-Computer Interaction at Virginia Tech. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Commonwealth Cyber Initiative, the National Science Foundation, or VT's Center for Human-Computer Interaction.

Moreover, the authors would like to acknowledge the use of ChatGPT to revise the text throughout all sections of the paper to correct typos, grammatical errors, and awkward phrasing.

References

[1] 2026. Augmented Reality Market Size, Share & Trends Analysis Report By Component, By Display Type, By Application, By Region, And Segment Forecasts. https://www.grandviewresearch.com/industry-analysis/augmented-reality-market

[2] 2026. ONNX Runtime Documentation: Quantization. https://onnxruntime.ai/docs/performance/model-optimizations/quantization.html. Accessed: 2026-02-19.

[3] Jad Al Aaraj and Athina Markopoulou. 2025. BLINDSPOT: Enabling Bystander-Controlled Privacy Signaling for Camera-Enabled Devices. *arXiv preprint arXiv:2512.14746* (2025).

[4] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. 2016. I-pic: A platform for privacy-compliant image capture. In *Proceedings of the 14th annual international conference on mobile systems, applications, and services*. 235–248.

[5] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. 2001. Recent advances in augmented reality. *IEEE computer graphics and applications* 21, 6 (2001), 34–47.

[6] Arif Bacchus. 2023. Microsoft HoloLens 2 hands-on review: The future on your face. https://www.digitaltrends.com/computing/microsoft-hololens-2-ar-hands-on-features-price-photos-video-release-date/#dt-heading-digital-meets-physical

[7] Gary Bishop, Greg Welch, et al. 2001. An introduction to the kalman filter. *Proc of SIGGRAPH, Course* 8, 27599-23175 (2001), 41.

[8] Ricard Borràs, Àgata Lapedriza, and Laura Igual. 2012. Depth information in human gait analysis: an experimental study on gender recognition. In *Image Analysis and Recognition: 9th International Conference, ICIAR 2012, Aveiro, Portugal, June 25-27, 2012. Proceedings, Part II 9*. Springer, 98–105.

[9] Gary Bradski. 2023. OpenCV GSoC 2023: FP16 and INT8 for DNN (Idea). https://github.com/opencv/opencv/wiki/GSoC_2023#idea-FP16-and-INT8-for-DNN. Accessed: 2026-02-19.

[10] Syed Ibrahim Mustafa Shah Bukhari, Maha Sajid, Bo Ji, and Brendan David-John. 2025. Rethinking privacy indicators in extended reality: Multimodal design for situationally impaired bystanders. In *2025 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 265–272.

[11] Kent Bye, Diane Hosfelt, Sam Chase, Matt Miesnieks, and Taylor Beck. 2019. The ethical and privacy implications of mixed reality. In *ACM SIGGRAPH 2019 Panels*. 1–2.

[12] Dimitris Chatzopoulos, Carlos Bermejo, Zhanpeng Huang, and Pan Hui. 2017. Mobile augmented reality survey: From where we are to where we go. *Ieee Access* 5 (2017), 6917–6950.

[13] Zijun Cheng, Tianwei Shi, Wenhua Cui, Yunqi Dong, and Xuehan Fang. 2017. 3D face recognition based on kinect depth data. In *2017 4th International Conference on Systems and Informatics (ICSAI)*. IEEE, 555–559.

[14] Yee-Yin Choong, Kurtis Goad, and Kevin C Mangold. 2022. Augmented Reality (AR) Usability Evaluation Framework.

[15] Matthew Corbett, Brendan David-John, Jiacheng Shang, Y Charlie Hu, and Bo Ji. 2023. BystandAR: Protecting bystander visual data in augmented reality systems. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*. 370–382.

[16] Matthew Corbett, Brendan David-John, Jiacheng Shang, Y Charlie Hu, and Bo Ji. 2023. Securing bystander privacy in mixed reality while protecting the user experience. *IEEE Security & Privacy* 22, 1 (2023), 33–42.

[17] David Darling, Ang Li, and Qinghua Li. 2019. Identification of subjects and bystanders in photos with feature-based machine learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 1–6.

[18] Shannon Davis. 2024. How Semiconductor Technologies are Enabling the Next Wave of Augmented Reality. https://www.semiconductor-digest.com/how-semiconductor-technologies-are-enabling-the-next-wave-of-augmented-reality/

[19] Andreas Dünser and Mark Billinghurst. 2011. Evaluating augmented reality systems. *Handbook of augmented reality* (2011), 289–307.

[20] Adám Erdélyi, Thomas Winkler, and Bernhard Rinner. 2018. Privacy protection vs. utility in visual data: An objective evaluation framework. *Multimedia tools and applications* 77 (2018), 2285–2312.

[21] Brennan Center for Justice. 2021. Federal Agencies Are Secretly Buying Consumer Data. (2021). https://www.brennancenter.org/our-work/analysis-opinion/federal-agencies-are-secretly-buying-consumer-data

[22] Andrea Gallardo, Chris Choy, Jaideep Juneja, Efe Bozkir, Camille Cobb, Lujo Bauer, and Lorrie Cranor. 2023. Speculative privacy concerns about ar glasses data collection. *Proceedings on Privacy Enhancing Technologies* (2023).

[23] Sarwesh Giri, Gurchetan Singh, Babul Kumar, Mehakpreet Singh, Deepanker Vashisht, Sonu Sharma, and Prince Jain. 2022. Emotion detection with facial feature recognition using CNN & OpenCV. In *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 230–232.

[24] Google. 2026. ARCore Documentation. https://developers.google.com/ar. https://developers.google.com/ar Accessed: 2026-02-24.

[25] Rakibul Hasan, David Crandall, Mario Fritz, and Apu Kapadia. 2020. Automatically detecting bystanders in photos to reduce privacy risks. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 318–335.

[26] Benjamin Henne, Christian Szongott, and Matthew Smith. 2013. SnapMe if you can: Privacy threats of other peoples' geo-tagged media and what we can do about it. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. 95–106.

[27] Kashmir Hill. 2022. The secretive company that might end privacy as we know it. In *Ethics of Data and Analytics*. Auerbach Publications, 170–177.

[28] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. 2013. A scanner darkly: Protecting user privacy from perceptual applications. In *2013 IEEE symposium on security and privacy*. IEEE, 349–363.

[29] Jaeyeon Jung and Matthai Philipose. 2014. Courteous glass. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*. 1307–1312.

[30] Jacob Leon Kröger, Otto Hans-Martin Lutz, and Florian Müller. 2020. What does your gaze reveal about you? On the privacy implications of eye tracking. In *IFIP International Summer School on Privacy and Identity Management*. Springer, 226–241.

[31] Linda Lee, J Lee, Serge Egelman, and David Wagner. 2016. Information disclosure concerns in the age of wearable computing. In *NDSS Workshop on Usable Security (USEC)*, Vol. 1. 1–10.

[32] Sarah M Lehman, Haibin Ling, and Chiu C Tan. 2020. Archie: A user-focused framework for testing augmented reality applications in the wild. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 903–912.

[33] Ang Li, Qinghua Li, and Wei Gao. 2016. Privacycamera: Cooperative privacy-aware photographing with mobile phones. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[34] Daniel J Liebling and Sören Preibusch. 2014. Privacy considerations for a pervasive eye tracking world. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. 1169–1177.

[35] Lolambean. 2023. HoloLens 2 hardware. https://learn.microsoft.com/en-us/hololens/hololens2-hardware

[36] Feiyu Lu, Leonardo Pavanatto, and Doug A Bowman. 2023. In-the-Wild Experiences with an Interactive Glanceable AR System for Everyday Use. In *Proceedings of the 2023 ACM Symposium on Spatial User Interaction*. 1–9.

[37] Feiyu Lu, Leonardo Pavanatto, Shakiba Davari, Lei Zhang, Lee Lisle, and Doug A Bowman. 2024. "where Did My Apps Go?" Supporting Scalable and Transition-Aware Access to Everyday Applications in Head-Worn Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[38] Angus Main and Dylan Yamada-Rice. 2022. Evading Big Brother: Using visual methods to understand children's perception of sensors and interest in subverting digital surveillance. *Visual Communication* 21, 3 (2022), 384–417.

[39] Mehdi Mekni and Andre Lemieux. 2014. Augmented reality: Applications, challenges and future trends. *Applied computational science* 20 (2014), 205–214.

[40] Microsoft. 2026. Mixed Reality Toolkit 3 (MRTK3) Overview. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-overview/ Accessed: 2026-02-24.

[41] Athanasios Nikolaidis. 2022. What is significant in modern augmented reality: a systematic analysis of existing reviews. *Journal of Imaging* 8, 5 (2022), 145.

[42] Joseph O'Hagan, Pejman Saeghe, Jan Gugenheimer, Daniel Medeiros, Karola Marky, Mohamed Khamis, and Mark McGill. 2023. Privacy-enhancing technology and everyday augmented reality: Understanding bystanders' varying needs for awareness and consent. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–35.
[43] Primal Pappachan, Roberto Yus, Prajit Kumar Das, Tim Finin, Eduardo Mena, Anupam Joshi, et al. 2014. A semantic context-aware privacy model for faceblock. In *Second International Workshop on Society, Privacy and the Semantic Web-Policy and Technology (PrivOn 2014), Riva del Garda (Italy)*.
[44] Shwetak N Patel, Jay W Summet, and Khai N Truong. 2009. Blindspot: Creating capture-resistant spaces. *Protecting Privacy in Video Surveillance* (2009), 185–201.
[45] Alfredo J Perez, Sherali Zeadally, Luis Y Matos Garcia, Jaouad A Mouloud, and Scott Griffith. 2018. FacePET: Enhancing bystanders' facial privacy with smart wearables/internet of things. *Electronics* 7, 12 (2018), 379.
[46] Michael E. Porter and James E. Heppelmann. 2025. Why every organization needs an augmented reality strategy. https://hbr.org/2017/11/why-every-organization-needs-an-augmented-reality-strategy
[47] Nisarg Raval, Animesh Srivastava, Kiron Lebeck, Landon Cox, and Ashwin Machanavajjhala. 2014. Markit: Privacy markers for protecting visual secrets. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication*. 1289–1295.
[48] Nisarg Raval, Animesh Srivastava, Ali Razeen, Kiron Lebeck, Ashwin Machanavajjhala, and Lanodn P Cox. 2016. What you mark is what apps see. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 249–261.
[49] Patrice Renaud, Joanne L Rouleau, Luc Granger, Ian Barsetti, and Stéphane Bouchard. 2002. Measuring sexual preferences in virtual reality: A pilot study. *CyberPsychology & Behavior* 5, 1 (2002), 1–9.
[50] Joe Saballa and Joe Saballa. 2022. US Army OKs acquisition of 5,000 IVAS goggles after Year-Long Delay. https://thedefensepost.com/2022/09/05/us-army-ivas-goggles-2/
[51] Maha Sajid, Syed Ibrahim Mustafa Shah Bukhari, Bo Ji, and Brendan David-John. 2025. " Just stop doing everything for now!": Understanding security attacks in remote collaborative mixed reality. In *2025 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 623–633.
[52] Jeremy Schiff, Marci Meingast, Deirdre K Mulligan, Shankar Sastry, and Ken Goldberg. 2009. Respectful cameras: Detecting visual markers in real-time to address privacy concerns. *Protecting privacy in video surveillance* (2009), 65–89.
[53] Jeff Shepard. 2022. What sensors are used in AR/VR systems? https://www.sensortips.com/featured/what-sensors-are-used-in-ar-vr-systems-faq/
[54] Aakash Shrestha, Yantian Hou, Min Long, and Jiawei Yuan. 2023. Virtual Curtain: A Communicative Fine-grained Privacy Control Framework for Augmented Reality. In *2023 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 188–194.
[55] Jiayu Shu, Rui Zheng, and Pan Hui. 2017. Your privacy is in your hand: Interactive visual privacy control with tags and gestures. In *Communication Systems and Networks: 9th International Conference, COMSNETS 2017, Bengaluru, India, January 4–8, 2017, Revised Selected Papers and Invited Papers 9*. Springer, 24–43.
[56] Jiayu Shu, Rui Zheng, and Pan Hui. 2018. Cardea: Context-aware visual privacy protection for photo taking and sharing. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 304–315.
[57] Robert Templeman, Mohammed Korayem, David J Crandall, and Apu Kapadia. 2014. PlaceAvoider: Steering First-Person Cameras away from Sensitive Spaces.. In *NDSS*, Vol. 14. Citeseer, 23–26.
[58] VRcompare. 2025. VRcompare – Compare VR Headsets. https://vr-compare.com/compare?h1=0q3goALzg&h2=EkSDYv0cW&h3=mt3AEYJu5. Accessed: 2025-07-02.
[59] Vuforia. 2026. Vuforia Developer Portal. https://developer.vuforia.com/home. https://developer.vuforia.com/home Accessed: 2026-02-24.
[60] Frederike Wenzlaff, Peer Briken, and Arne Dekker. 2016. Video-based eye tracking in sex research: a systematic literature review. *The Journal of Sex Research* 53, 8 (2016), 1008–1019.
[61] Roberto Yus, Primal Pappachan, Prajit Kumar Das, Eduardo Mena, Anupam Joshi, and Tim Finin. 2014. Faceblock: privacy-aware pictures for google glass. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. 366–366.

A EvaluatAR's Algorithm

Alg. 3 visualizes the control loop of EvaluatAR. In `Collect` mode, EvaluatAR records time-synchronized sensor inputs and runtime measurements. In `Replay` mode, it (1) establishes alignment using the reference marker, (2) disables alignment components once positioned to reduce overhead, and (3) replays logged inputs by selecting the appropriate log entry as a function of elapsed replay time. This time-based replay mechanism supports reproducible alignment between replayed inputs and the stimulus across headsets with different processing rates.

Algorithm 3 EvaluatAR control loop.

Parameters: PositioningCube, AlignmentVisualizer, QRDetector, H_t, Q_t, ReplayLogsPath, LogData, CameraFrame, Mode $\in \{Collect, Replay\}$
logBuffers, elapsedT = 0
isTogglePressed, isRefImgCaptured = false
if Mode == *Replay* **then**
    $logBuffers \leftarrow Load(ReplayLogsPath)$
else
    Disable PositioningCube, AlignmentVisualizer
end if
while True **do**
    Calculate FPS
    if isTogglePressed **then**
        elapsedT $\leftarrow StopWatch.Start$
        isTogglePressed $\leftarrow false$
    end if
    if Mode == *Collect* **then**
        $logBuffers \leftarrow (time, elapsedT, FPS, LogData)$
    else
        if $H_t \in PositioningCube.Bounds$ **then**
            if isRefImgCaptured **then**
                Disable QRDetector
            else
                Save CameraFrame
                isRefImgCaptured $\leftarrow true$
            end if
        else
            UpdatePositioningCubePose($Q_t, logBuffers.QRPose$)
        end if
        if isTogglePressed **then**
            for j $\leftarrow$ 1 **to** j $\leq LogData.Count$ **do**
                if elapsedT == logBuffers[j].elapsedT **then**
                    Replay logBuffers[j].elapsedT
                end if
                if elapsedT < logBuffers[j].elapsedT **then**
                    Replay logBuffers[j−1].elapsedT
                end if
            end for
        end if
    end if
end while

B BystandAR

As shown in Alg. 4, BystandAR maps each detected face to a 3D coordinate and spawns a corresponding virtual GameObject to maintain identity across frames. Tracking is achieved by checking whether the projected positions in subsequent frames overlap with existing GameObjects, using Unity's `Physics.OverlapBox`. The first overlapping match is taken as the continuing identity. While this approach works in simple scenarios, it fails in crowded or complex settings with multiple overlapping or crossing faces.

Proposed Modifications. We implemented and evaluated four modifications to BystandAR's tracking pipeline. *Naive Predicted Positioning (NPP)*, *Kalman Predicted Positioning (KPP)*, *Closest Depth (CD)*, and *Hybrid method*.

NPP extends the BystandAR algorithm by naively calculating and storing the predicted position of each detection in the next frame by assuming consistent motion (Algorithm 5). The prediction is computed based on the difference in 3D position between the past frame and the current frame and assumes the same translation will occur. Whenever there is an overlap between two faces, this approach uses the Cartesian distance between the 3D position of the new detection with the predicted positions of the overlapping faces and chooses the one with the smallest distance. The KPP method instead uses a Kalman filter [7] to calculate and store the predicted position of each bounding box (Algorithm 5).

The CD approach relies solely on the depth of the detections to resolve overlapping bounding boxes (Algorithm 6). This approach relies on the accuracy of face detection and the mapping from 2D camera space to 3D world space provided by BystandAR [15]. Our implementation compares the Z-coordinate of overlapping bounding boxes to select the closest match.

The Hybrid method fuses the KPP and CD predictions when computing the distance between the new inference and overlapping bounding boxes (Algorithm 7). The distance value is the weighted sum of the measures used in KPP and CD (the weights used for KPP and CD are 0.2 and 0.8, respectively).

C Cardea-Inspired PET

As shown in Alg. 8, the Cardea-inspired explicit PET enforces per-face visual obfuscation based on bystander intent gestures. Each frame, the PET runs face, hand, and gesture detection on the current camera frame. When at least one face is eligible for interaction, the PET associates detected gestures to faces using a proximity-based face–hand mapping in image space.

Algorithm 4 BystandAR control loop with EvaluatAR's hooks. The highlighted block is replaced by our proposed modifications.

Parameters: sampling interval N, Mode $\in$ {Baseline, EvaluatAR-Collect, EvaluatAR-Replay}, QRDetector
FrameNum = 0
Detector $\leftarrow$ FaceDetection
while True **do**
  CameraFrame $\leftarrow$ current camera frame; DepthFrame $\leftarrow$ retrieve raw depth
  FrameNum++; $S_t \leftarrow$ current eye gaze, voice input
  if Mode $\neq$ Baseline **then**
    $H_t \leftarrow$ HeadsetPose; $Q_t \leftarrow$ QRDetector.Pose
    EvaluatAR.setCurrentCameraCapture(CameraFrame)
    if Mode == EvaluatAR-Replay **then**
      $S_t \leftarrow$ EvaluatAR.getCurrentFrameData()
    end if
  end if
  if S_t intersects with a face **then**
    Increment eye/voice tracker for face
    if Eye-gaze/Voice history > Threshold **then**
      Label face a subject
    else
      Label face becomes/remains bystander
    end if
  end if
  if FrameNum $\geq N$ **then**
    FrameNum = 0
    faces $\leftarrow$ Detector(CameraFrame)
    for each face $\in faces$ **do**
      Transform 2D detection to 3D world space
      if face overlaps with an existing face **then**
        Replace current detection; reset TTL
      else
        Create new detection
      end if
      if Application requesting sensor data **then**
        Obscure bystander faces in frame
      end if
    end for
    else
      if Application requesting sensor data **then**
        Obscure bystander faces in frame
      end if
    end if
    if Mode $\neq$ Baseline **then**
      if Mode == EvaluatAR-Replay **then**
        LogData $\leftarrow$ $(FrameNum, faces)$
      else
        LogData $\leftarrow$ $(FrameNum, Q_t, H_t, S_t)$
      end if
      EvaluatAR.writeToLogsFile(LogData)
    end if
    Release frames to the application
  end while

Algorithm 5 Control loop for NPP and KPP modifications

```
if face overlaps with an existing face then
    Compute smallest distance using predicted positions
    Replace current detection; reset TTL
else
    Create new detection
end if
```

Algorithm 6 Control loop for CD modification

```
if face overlaps with an existing face then
    Compute smallest distance in Z dimension
    Replace current detection; reset TTL
else
    Create new detection
end if
```

Algorithm 7 Control loop for Hybrid (KPP and CD) modification

```
if face overlaps with an existing face then
    Compute weighted sum of KPP and CD distances
    Replace current detection; reset TTL
else
    Create new detection
end if
```

Algorithm 8 Cardea-inspired PET's control loop with EvaluatAR's hooks.

```
Parameters: Gesture ∈ {OpenPalm, victory}, Mode ∈ {Baseline, EvaluatAR-Collect, EvaluatAR-Replay}, QRDetector
FaceDetector ← FaceDetection
HandPoseDetector ← HandPoseDetection
while True do
    CameraFrame ← current camera frame
    if Mode ≠ Baseline then
        H_t ← HeadsetPose;  Q_t ← QRDetector.Pose
        EvaluatAR.setCurrentCameraCapture(CameraFrame)
    end if
    faces ← FaceDetector(CameraFrame)
    handGestures ← HandPoseDetector(CameraFrame)
    handFaceMap ← (face, handGesture) proximity-based pairs
    for each mapping ∈ handFaceMap do
        if mapping[handGesture] == openPalm then
            Apply blur for obfuscation
        end if
        if mapping[handGesture] == victory then
            Remove obfuscation
        end if
    end for
    if Mode ≠ Baseline then
        if Mode == EvaluatAR-Replay then
            LogData ← data of faces and their hand pose
        else
            LogData ← (FrameNum, Q_t, H_t)
        end if
        EvaluatAR.writeToLogsFile(LogData)
    end if
    Release frames to the application
end while
```

D Case Study 2

This appendix section consists of additional graphs for Experiment 2 (§5.2.1) for completeness. These plots represent the results of Cardea-inspired explicit PET under standardized replay of inputs on MQ3 and ML2 across two visual stimuli (Video 1: one bystander; Video 2: two bystanders) and two model stacks (high vs. low)

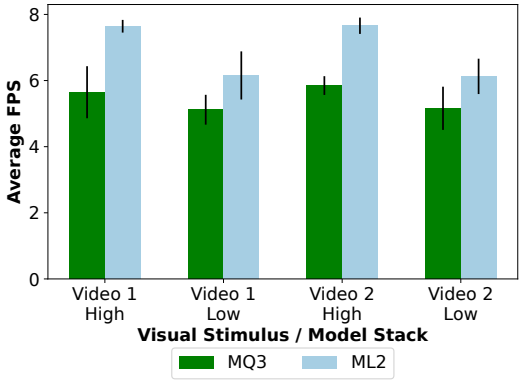


Figure 9: Average FPS of the explicit PET across headsets, visual stimuli, and model stacks. ML2 consistently achieves higher FPS. Interestingly, the low-precision stack does not improve runtime performance.

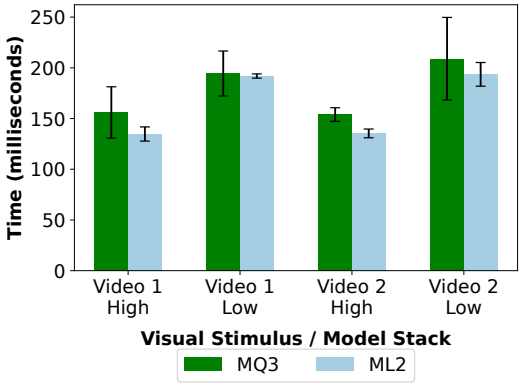


Figure 10: Intent-to-enforcement processing time for the explicit PET across trials. Processing time varies with stimulus and model stack, but the low-precision stack does not offer much end-to-end latency benefit.