

EDIVA: Exploiting Human Behavioral Patterns for Differential Egocentric Video Processing in AR

Tianhua Xia
Tandon School of Engineering
New York University
New York, USA
tx856@nyu.edu

Haiyu Wang
Tandon School of Engineering
New York University
New York, USA
hw3689@nyu.edu

Fengze Yu
Tandon School of Engineering
New York University
New York, USA
fy2243@nyu.edu

Jieyu Lin
Reality Labs Research
Meta
Burlingame, USA
jyericlin@meta.com

Ziyun Li
Reality Labs Research
Meta
Redmond, USA
liziyun@meta.com

Barbara De Salvo
Reality Labs Research
Meta
Burlingame, USA
barbarads@meta.com

Sai Qian Zhang
Tandon School of Engineering
New York University
New York, USA
sai.zhang@nyu.edu

Abstract

Augmented reality (AR) systems increasingly rely on real-time image segmentation to interpret continuous egocentric video streams and enable immersive, intention-aware interactions. However, executing neural segmentation models on every high-resolution frame introduces significant computational overhead and latency, exceeding the real-time constraints of AR devices. This paper presents EDIVA, an algorithm–hardware co-optimization framework for differential egocentric video processing that exploits temporal redundancy and multimodal sensory cues in AR environments. EDIVA uses gaze, pose, and inertial signals to infer user intent and reproject previous segmentation masks via perspective transformation, updating only regions with significant visual changes. A lightweight mask refinement module corrects residual distortions to maintain segmentation fidelity. We further design a dedicated EDIVA accelerator that integrates with the AR system-on-chip (SoC), enabling efficient cache management, mask reprojection, and reuse detection. Evaluations on representative AR workloads show that EDIVA reduces segmentation latency by $2.67\times$ and energy consumption by $3.65\times$ while preserving accuracy and temporal consistency, achieving real-time performance on edge AR hardware. Code is available at <https://doi.org/10.5281/zenodo.23225199>.

CCS Concepts

• **Computer systems organization** → **Neural networks**; *System on a chip*; • **Computing methodologies** → *Machine learning*.

Keywords

Video Segmentation; Hardware Accelerator; Augmented Reality

ACM Reference Format:

Tianhua Xia, Haiyu Wang, Fengze Yu, Jieyu Lin, Ziyun Li, Barbara De Salvo, and Sai Qian Zhang. 2027. EDIVA: Exploiting Human Behavioral Patterns for Differential Egocentric Video Processing in AR. In *Proceedings of the 32nd ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1 (ASPLOS '27)*, April 11–15, 2027, Heraklion, Greece. ACM, New York, NY, USA, 18 pages. <https://doi.org/10.1145/3845814.3850080>

1 Introduction

Virtual Reality (VR) and Augmented Reality (AR) are immersive technologies that bridge the digital and physical worlds. VR creates a fully simulated environment that allows users to interact with computer-generated spaces, often through head-mounted displays. AR, in contrast, enhances the real world by overlaying digital content such as visuals, audio, and contextual information onto the user’s surroundings, transforming a wide range of fields [5, 15, 25, 74, 94, 119].

As AR continues to evolve toward richer and more interactive experiences, its effectiveness increasingly depends on how well the system perceives and interprets continuous video streams from the AR user’s environment. Modern AR platforms rely on computer vision techniques to analyze these real-time video inputs, enabling consistent alignment between virtual and physical elements. Within this framework, segmentation has become essential for parsing each video frame into meaningful regions, enabling AR systems to anchor digital content accurately, enhance realism, and maintain seamless interaction with the real world. Figure 1 illustrates two examples. In educational settings (Figure 1 (a)), segmentation identifies the object a user is gazing at, providing the instance of interest (IOI) to downstream modules such as a vision–language model (Figure 1



This work is licensed under a Creative Commons Attribution 4.0 International License. ASPLOS '27, Heraklion, Greece

© 2027 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-3012-2/2027/04

<https://doi.org/10.1145/3845814.3850080>

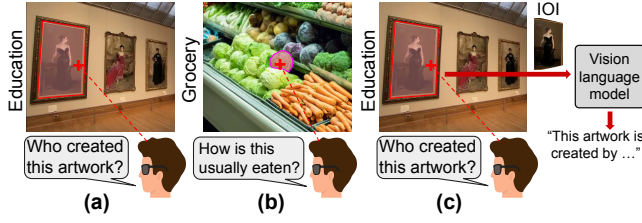


Figure 1: The user wears the AR device and views various scenes, with real-time image segmentation applied.

Platform	HRNet [116]	SegFormer [125]	SAM-B [54]	ESAM-S [127]
Jetson Orin NX [4]	779 ms	1419 ms	5462 ms	1307 ms
Qualcomm XR2 [88]	252 ms	880 ms	3471 ms	464 ms

Table 1: Segmentation latencies of different models across platforms. Latencies on the Qualcomm XR2 Gen2 are measured on the NPU using the Qualcomm AI Hub Toolkit [87].

(c)). In an AR grocery scenario (Figure 1 (b)), real-time segmentation across consecutive frames helps users locate items and enhances the shopping experience. Prior work has explored many additional AR applications built on image segmentation [7, 43, 44, 109].

Segmentation serves as a fundamental building block in AR, supporting downstream tasks such as object recognition [145], scene understanding [107], and multimodal reasoning [16]. Many AR applications rely on accurate frame-level segmentation to localize and interpret user focus in complex scenes. However, segmentation is computationally expensive. Running deep neural network (DNN)-based segmentation continuously on high-resolution video introduces substantial compute overhead and latency, often causing perceptible delays that degrade responsiveness and immersion.

To examine this issue, we benchmark image segmentation workloads using established DNN models, including HRNet-W32 [116], SegFormer B2 [125], SAM-B [54], and EfficientSAM-Small [127] (ESAM-S), at an input resolution of 1408×1408 , matching the video resolution of typical AR devices (e.g., Meta Aria [32] and VIVE Pro 2 [2]). Experiments are conducted on the NVIDIA Jetson Orin NX edge GPU [4], which is widely used in prior studies evaluating edge GPU performance for AR systems [41, 51, 80, 93, 136, 140], and on the Qualcomm XR2 Gen2 platform [88], used in Meta Quest Pro [52] and Meta Aria [32]. As shown in Table 1, average processing latencies reach 2241 ms on Jetson Orin NX and 1267 ms on XR2, orders of magnitude above the latency threshold required for a fluid visual experience, which prior research identifies as approximately 50-70 ms [6]. This issue is further exacerbated during active user movement, where large motion-to-photon latency [92] causes significant misalignment between the rendered mask and the underlying object, leading to severe cybersickness [57].

The usage scenarios in Figure 1 illustrate two key properties of segmentation in AR environments that differ from conventional computer vision settings. First, AR segmentation is inherently intention-driven: users typically attend only to the IOI within their field of view, often indicated by gaze direction. As shown in Figure 1, this means segmentation can be localized around the gazed region rather than performed densely over the entire frame. Second, the

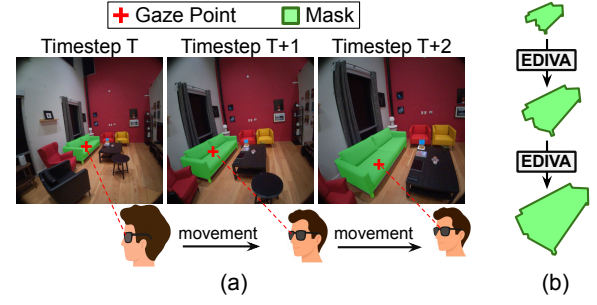


Figure 2: (a) Temporal correlation across the segmentation masks. (b) EDIVA performs segmentation in a differential manner to improve efficiency.

segmentation masks of gaze-focused objects exhibit strong temporal consistency across successive frames. This arises from the smooth, continuous nature of head and eye movements in AR users, where the viewpoint evolves gradually rather than undergoing abrupt changes. As illustrated in Figure 2 (a), at timestep T, assume the user is gazing at a sofa that is segmented by the DNN. At timestep T+1, a small rotation (e.g., a head swing) or translation of the user's head (e.g., walking) slightly shifts the front view, yet IOI remains the same. Consequently, the segmentation masks of the IOI across adjacent frames are highly correlated. Instead of performing independent segmentation for every incoming frame, **leveraging this temporal redundancy offers an opportunity to substantially reduce redundant computation while maintaining accurate and temporally consistent segmentation over time.**

In this work, we begin by analyzing real-world AR usage traces collected from head-mounted AR devices to quantitatively validate these properties. Additionally, modern AR platforms expose a rich set of sensory signals, including gaze estimation, head and device pose and inertial measurements, that can be exploited to predict IOI motion, constrain the candidate regions, and guide selective updates of segmentation results. Building on these insights, we propose *EDIVA*, a comprehensive algorithm-hardware co-optimization framework that enables differential segmentation for egocentric AR video streams. Rather than recomputing dense segmentation for every frame, the EDIVA algorithm leverages multimodal AR signals to track user motion and IOI dynamics, incrementally updating only the affected regions by selecting and reprojecting appropriate masks to the current point of view while reusing prior results elsewhere. In doing so, it efficiently generates accurate IOI segmentation masks on resource-constrained AR devices with substantially reduced computational cost, while preserving the real-time segmentation accuracy and temporal consistency. Specifically, our contributions can be summarized as follows:

- We propose the EDIVA algorithm, a deep learning-based framework that leverages user motion and gaze location to exploit the temporal correlation for efficient video segmentation in AR environments. The EDIVA framework is designed to be integrated with any existing segmentation network, enhancing its generality.

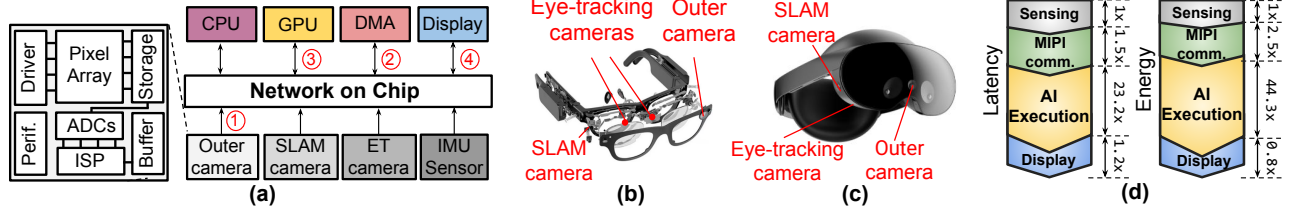


Figure 3: (a) The SoC of the AR device. (b) Meta Orion Glass and (c) Meta Quest Pro. Although the Meta Quest Pro is primarily a VR headset, it can operate as an AR device by showing the real-world scene through its passthrough cameras. (d) Latency breakdown of segmentation process on AR device.

- We propose the EDIVA accelerator as a plug-in for AR SoC that manages video caching, performs differential segmentation, and significantly reduces workload on edge GPUs/NPUs. We also introduce a lightweight enhancement to the image signal processor that lowers the image transmission costs.
- Evaluation results show that EDIVA delivers a $2.67\times$ speedup and $3.65\times$ energy reduction over hardware baselines, with negligible accuracy loss and improved visual quality.

2 Background and Related Work

2.1 AR Device Overview

Modern AR devices are equipped with multiple types of sensors that provide rich sensory information and can continuously track the user's state, such as gaze location and head pose, at very low cost. Figure 3 (b) and (c) illustrate the Meta Orion glasses and the Meta Quest Pro, both of which integrate a rich multi-camera and sensor suite for egocentric perception. Each device incorporates outward-facing RGB cameras that capture high-resolution visual (2880×2880) input. In addition, inward-facing infrared cameras are dedicated to eye tracking (ET). Tracking cameras are used for Simultaneous Localization and Mapping (SLAM). ET and IMU sensors are standard components in modern AR systems, including Microsoft HoloLens 2 [26], Apple Vision Pro [13], and Magic Leap 2 [68]. Together, this sensing stack provides high-quality visual input, gaze information, and precise pose tracking at low cost, making these platforms well-suited for intention-aware AR interaction.

Figure 3 (a) illustrates the primary components of the AR SoC, which comprise the sensing subsystem, alongside the CPU, GPU, DMA controller, and display engine. Some advanced AR devices also integrate a neural processing unit (NPU) to accelerate AI-related workloads. The detailed architecture of the image sensor is shown on the left side of Figure 3 (a), consisting of a pixel array, analog-to-digital converters (ADCs), readout circuits, and both analog and digital buffers. The digitized data are passed to the image signal processor (ISP) before forwarding the processed data to DRAM of SoC via Mobile Industry Processor Interface (MIPI) [3].

To perform conventional image segmentation, the processing flow is illustrated by the numbered steps in Figure 3 (a). At each timestep, the outward-facing camera captures a high-resolution frame and transmits it through the MIPI interface (Step 1), where it is temporarily buffered in DRAM (Step 2). The GPU then executes a DNN-based segmentation model (Step 3) to produce the segmentation mask, which is subsequently rendered on the AR

display (Step 4). However, implementing segmentation over GPU introduces substantial processing latency, resulting in a noticeable degradation of the user's visual experience. Specifically, Figure 3 (d) shows the latency and energy breakdown for processing a single 1408×1408 input frame with ESAM-S [127], revealing that most of the cost is dominated by executing the AI model on the GPU. In addition, the MIPI communication overhead, which scales with the number of transmitted pixels, is also noticeable [42, 63, 137]. In contrast, the latency and energy contributions from sensing and display are relatively minor.

2.2 Saccadic Motion of AR User

Human gaze alternates between stable fixations, where the eyes remain still to gather visual information, and rapid saccades, which redirect gaze to a new target. During saccades, visual sensitivity drops sharply [21, 29, 72], enabling AR systems to temporarily lower shading rates or pause tasks such as super resolution and segmentation [55, 56, 103, 114].

In headset viewing, both eye motion and head motion contribute to gaze shifts, producing saccades with a wide range of speeds and dynamics, from eye-only shifts to coordinated eye and head redirections toward a new target [38, 46, 47]. During head motion, the vestibulo-ocular reflex (VOR) [11, 27, 28] rotates the eyes in the direction opposite to the head in order to stabilize the retinal image. As a result, gaze in the world frame can remain nearly stable even when the eyes rotate substantially in the head frame [19, 105]. Thus, accurate saccade detection must consider both eye motion and head motion and must reason about gaze behavior in the world frame rather than eye rotation alone.

2.3 Image and Video Segmentation

Semantic segmentation [22–24, 59, 73, 110, 115, 126] assigns a semantic label to every pixel in an image, dividing the scene into meaningful regions for structured analysis. Building on this, instance segmentation [18, 49, 60, 75, 123, 135] further separates individual object instances, enabling finer detail and more precise scene understanding. Prior work on video instance segmentation [62, 90, 117, 121, 128, 129] often processes multiple consecutive frames jointly to leverage temporal cues and improve accuracy. Although this improves continuity, it adds substantial latency.

Other prior work uses optical flow [14, 112] to improve video segmentation consistency and efficiency. However, flow estimation models require specialized architectures or training schemes to

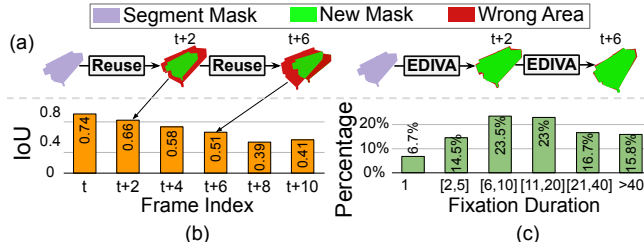


Figure 4: (a) Comparison between mask reuse and EDIVA differential segmentation. (b) IoU of reused masks over time. (c) Distribution of the fixation durations.

be effective [30, 112]. Flow computation itself also adds notable overhead [65, 122, 139]. In contrast, EDIVA uses built-in motion and gaze signals from AR devices to exploit temporal correlation directly in the IOI determined by user behavior and is model-agnostic.

2.4 Temporal Correlation in AR Scene Segmentation

Human visual perception in AR shows strong temporal correlation. Such redundancy offers significant opportunities for computational reuse, especially for tasks like segmentation where object boundaries remain stable across frames. As illustrated in Figure 4 (a), when the user continues looking at the same object, the segmentation mask can often be reused rather than recomputed.

To quantify this effect, we analyze 10 representative sequences from the Aria Everyday Activities (AEA) [66] and Aria Digital Twin (ADT) datasets [81]. We construct a histogram of *fixation durations*, defined as the number of consecutive frames during which the user continues to gaze at the same object, as shown in Figure 4 (c). The results reveal that 23.5% of all frames belong to video segments where the gaze remains on the same object for 6–10 frames, while 15.8% correspond to extended fixations lasting more than 40 frames. Only 6.7% of frames involve immediate gaze shifts.

However, directly reusing previous segmentation masks without adaptation can lead to poor segmentation performance and degrade the user experience. Even when the user continues to look at the same object, natural head or body movement can change the object’s apparent size, shape, or position within the field of view. As illustrated in Figure 2 (a), when the user walks toward a sofa, the sofa gradually occupies a larger portion of the view, leading to geometric distortion if the earlier mask is reused. To quantify this effect, we evaluate mask reuse accuracy using the ESAM model on a representative sequence from the ADT dataset. As shown in Figure 4 (b), the intersection over union (IoU) between the ground truth mask and the reused mask is 0.74 at timestep t , but it drops to 0.66 after only two timesteps $t + 2$, with accuracy continuing to decline as reuse duration increases. This result highlights the need for **dynamic mask adjustment to account for user motion**.

3 EDIVA Algorithm

Performing segmentation on every individual AR frame introduces significant processing latency and severely degrades the user experience. In this section, we present the *EDIVA algorithm* to refine

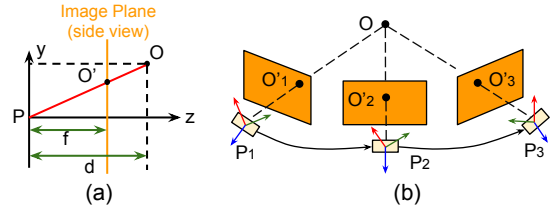


Figure 5: (a) Side view of perspective projection of O to O' in the image plane. (b) Perspective reprojection process.

segmentation masks across time in a differential manner with high efficiency.

3.1 Geometry-Based Mask Reprojection

Motivated by the observations in Figure 2, where segmentation masks across consecutive frames exhibit strong temporal correlation, we note that when a user continuously gazes at the same IOI, the variation in the segmentation mask primarily arises from changes in the egocentric front view, which in turn depend on the user’s pose. Since pose information can be readily obtained from tracking information from the AR headset from IMU sensors, we leverage this geometric information to efficiently estimate the evolving segmentation mask over time with minimal cost.

Image capture in a camera follows the principle of perspective projection [8], where a 3D point O in the scene is projected onto a 2D point O' on the image plane, as shown in Figure 5 (a). This projection can be expressed by a transformation matrix $T_{wc}(f) \in \mathbb{R}^{4 \times 4}$, where $T_{wc}(f)$ depends on the camera focal length f . Conversely, a 2D image coordinate can be unprojected back into 3D space using the principle of similar triangles, also illustrated in Figure 5 (a). This requires the depth d of point O , defined as its distance from the camera optical center P along the z axis. The unprojection operation is represented by another transformation, $T_{cw}(f, d) \in \mathbb{R}^{4 \times 4}$.

Additionally, an example of *reprojection* is illustrated in Figure 5 (b), where the camera moves from position P_1 to P_2 and P_3 . The projections of point O on image plane 1 and image plane 2 are denoted as O'_1 and O'_2 , respectively. The 2D coordinates of the projected points O'_1 and O'_2 are denoted as o'_{f_1} and o'_{f_2} , respectively. Given a point o'_{f_1} within the segmentation mask of the coordinate system P_1 , the goal is to compute its corresponding projection o'_{f_2} in the coordinate system P_2 . With the camera intrinsic model and the depth (d_1) of O at P_1 , we first unproject O'_1 in the 2D image plane to the 3D coordinate of O in P_1 camera coordinate system, denoted as o_{p_1} . Then we compute the 3D coordinate of O in P_2 camera coordinate system, denoted as o_{p_2} , using the tracking information from the IMU sensor, which returns the device position and orientation in real time. With their respective translation vectors and rotation matrices, the transformation matrix from o_{p_1} to o_{p_2} can be calculated and denoted as $T_{p_1 \rightarrow p_2}$. We then project o_{p_2} onto the image plane of camera P_2 to obtain o'_{f_2} using the projection matrix $T_{wc}(f)$. The resulting expression for computing o'_{f_2} from o'_{f_1} is given as:

$$[o'_{f_2}^T, f, 1]^T = T_{wc}(f) T_{p_1 \rightarrow p_2} T_{cw}(f, d_1) [o'_{f_1}^T, f, 1]^T \quad (1)$$

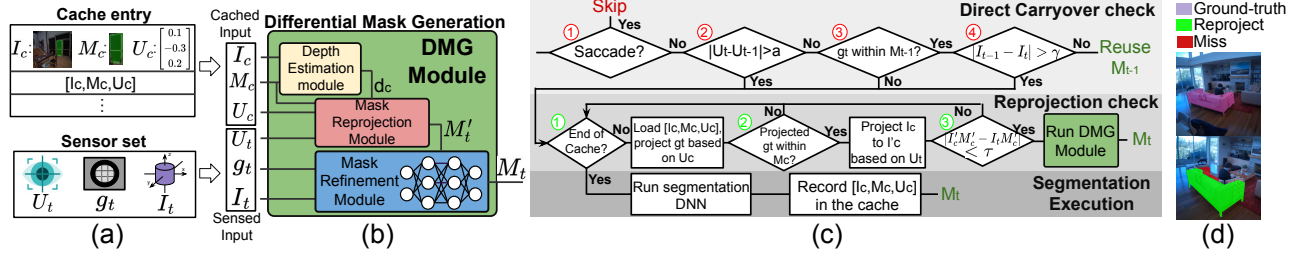


Figure 6: (a) The composition of cache for mask reuse. (b) EDIVA differential mask generation module. (c) EDIVA algorithm to produce mask. The step numbers for Direct Carryover check and reprojection check are highlighted in red and green, respectively. (d) Mask reprojection effect. The mask refinement module covers the missing part.

Equation 1 can be applied to every point in the segmentation mask of o'_{f1} to compute its corresponding location in o'_{f2} . Since $T_{wc}(f)$, $T_{p1 \rightarrow p2}$, and $T_{cw}(f, d_1)$ are all 4×4 matrices, the computation is very lightweight. The same procedure can be applied to compute the reprojection from P_2 to P_3 . The Mask Reprojection Module, shown in red in Figure 6 (b), takes as input the cached original frame I_c , its mask M_c , pose U_c (Figure 6 (a)), the estimated depth map d_c described in Section 3.2, and the current pose U_t . Using these inputs, it produces an estimated mask M'_t from the new viewpoint defined by U_t , following the formulation in Equation 1.

3.2 Depth Estimation Module

As shown in Equation 1, the depth information d_1 for each point within the mask is essential for performing the unprojection step. However, such depth data are typically unavailable from standard AR devices. To address this limitation, we design a lightweight DNN to estimate the required depth values. Unlike prior approaches that predict dense per-pixel depth maps for the entire image [31, 40, 113] at high computational cost [120, 130, 132], EDIVA restricts depth inference to the pixels within the segmentation mask of the IOI. Furthermore, the depth estimation module produces a single average depth value and reuses it across all pixels within the IOI mask. This design choice is motivated by the observation that the refinement mechanism described in Section 3.3 requires only this coarse depth estimate to generate a high-quality refined mask. The mask comparison mechanism introduced in Section 3.4 prevents low-quality reprojection and refined masks from being reused by triggering a new segmentation, thereby further ensuring segmentation quality. As validated by the evaluation results in Section 5, this approach significantly reduces computational overhead while maintaining segmentation accuracy.

As illustrated by the yellow block in Figure 6 (b), the depth estimation module takes the cached input image I_c and its corresponding mask M_c as inputs, and outputs a single average depth prediction d_c , which is subsequently used in the mask reprojection stage. The module employs a compact four-layer neural network, where the mask M_c is concatenated as an additional input channel to I_c for joint processing.

3.3 Mask Refinement Module

The reprojection mechanism outlined in Section 3.1 enables efficient estimation of the current segmentation mask with significantly reduced computational overhead. However, this approach has an inherent limitation: **it fails to account for regions of the object that become newly visible from the updated viewpoint**. As illustrated in Figure 6 (d), the top panel depicts the original segmentation mask (in purple), while the bottom panel shows the reprojection result (in green). The red regions correspond to the incremental portions of the IOI that emerge from the new viewing angle and are therefore absent in the reprojection mask. Furthermore, inaccuracies in pose estimation, depth estimation, and camera distortion can introduce additional degradation in segmentation quality. To mitigate these effects, we propose a DNN-based *Mask Refinement Module* that utilizes the reprojection mask as a strong spatial prior, enabling the generation of a more precise and consistent segmentation result.

The blue block in Figure 6 (b) illustrates the architecture of the mask refinement module. Let g_t and I_t denote the current gaze position and input frame, respectively, and let the input from the mask reprojection module (Section 3.1) serve as its spatial prior. The module refines this input to produce an enhanced segmentation mask M_t for frame I_t , achieving substantially improved quality and boundary accuracy. The mask refinement module employs a lightweight DNN with a U-shaped architecture [91] with a three-layer encoder and decoder. We downsample the input image to further reduce computation overhead.

3.4 Differential Mask Generation and Reprojection Check

Figure 6 (b) shows the architecture of the differential mask generation (DMG) module, which is composed of a depth estimator, a mask reprojection module, and a mask refinement module. The DMG module reprojects cached masks to the current viewpoint according to the user's present pose U_t .

The cache structure is illustrated at the top of Figure 6 (a). Each entry contains three components: the RGB image I_c , its segmentation mask M_c , and the corresponding pose information U_c . The cache stores multiple entries that can later be reprojected to generate the current IOI mask.

The *reprojection check* branch in Figure 6 (c) determines whether a cached mask can be reprojected to produce the current IOI mask,

or whether a new segmentation must be performed. Given the current pose U_t , gaze location g_t , and captured image I_t , the framework proceeds as follows. If the cache is non-empty (step 1), cache entries are examined in temporal order, prioritizing the most recently added items. Before testing whether the current gaze position lies within a cached mask M_c , we first reproject the current gaze coordinate g_t from the current pose U_t into the cached pose U_c . If the reprojected gaze falls inside M_c (step 2), the corresponding cached RGB image I_c together with M_c is reprojected to the current viewpoint, producing I'_c and M'_c . We then compute the pixel-wise difference between the masked regions of I'_c and I_t (step 3). If this difference is below a predefined threshold τ , the cached mask M_c is deemed consistent with the current IOI and is reprojected by the DMG module to generate the new mask M_t . If the condition fails, the system proceeds to the next cache entry. The search terminates once a valid match is found; otherwise, if no entry satisfies the constraints, the segmentation model is invoked to generate new masks for all objects in I_t , and only the masks of the IOI are stored as a new cache entry. If the cache is full, the oldest entries are evicted to make room for incoming ones.

3.5 Saccade Detection

As discussed in Section 2.2, saccadic motion reduces visual sensitivity, enabling the system to briefly pause segmentation and save computation. To enable this, we detect saccades by combining head and eye motion using a gaze angular velocity model. Let $R_{h,w}$ be the rotation matrix from the head frame to the world frame obtained from IMU readings. The gaze angular velocity in the world frame is $\omega_{g,w} \approx \omega_{h,w} + R_{h,w} \omega_{e,h}$, where $\omega_{h,w}$ is the head angular velocity in the world frame and $\omega_{e,h}$ is the eye angular velocity in the head frame. Both terms can be obtained or easily computed from IMU and eye-tracking data. A saccade is detected when the magnitude $\|\omega_{g,w}\|$ exceeds a velocity threshold, typically around 30°/s as used in standard I-VT velocity-threshold methods [17, 78, 79, 95]. This composition-based approach naturally accounts for eye-only saccades, coordinated eye-head movements, and the compensatory effects of the vestibulo-ocular-reflex that keep gaze nearly stable.

3.6 EDIVA Algorithm Summary

Given the detailed descriptions of each component in the previous sections, we now summarize the full EDIVA pipeline by presenting the complete algorithm, as shown in Figure 6 (c), which consists of three core components: *Direct Carryover check*, *Reprojection check* and *Segmentation execution*. The purpose of Direct Carryover check is to determine whether the current segmentation mask M_t can simply reuse the mask from the last timestep M_{t-1} .

Specifically, the algorithm begins with the saccadic check introduced in Section 3.5. At each timestep t , this check is performed to determine whether the user is undergoing a saccadic motion. If a saccade is detected (Step 1), the current frame is skipped to avoid unnecessary computation during rapid eye and head movements. When no saccade is detected, the system next compares the current pose U_t with the previous pose U_{t-1} . If their Euclidean distance exceeds a threshold α , the user is considered to have moved, and the previous mask M_{t-1} cannot be reused (Step 2). Otherwise, we next check whether the current gaze location g_t remains within

the mask M_{t-1} (Step 3). If it does, the pixel-wise difference between the previous RGB image I_{t-1} and the current image I_t is evaluated (Step 4) to determine if the IOI has moved. If this difference is below a threshold γ , the IOI is assumed to remain unchanged, and M_{t-1} can be safely reused.

If M_{t-1} cannot be reused, the reprojection-check branch is executed as described in Section 3.4. If the reprojection-check branch also fails, the segmentation neural network is applied to the current frame I_t , and the corresponding cache entries are refreshed. Although EDIVA greatly reduces the number of segmentation operations during video processing, a single segmentation step still introduces noticeable latency. To mitigate this, we apply average downsampling to the high-resolution input image so that segmentation can meet real-time latency requirements. To compensate for the information loss introduced by downsampling, the mask refinement module is trained to reconstruct a high-resolution mask from the low-resolution output, preserving fine object boundaries.

4 EDIVA Hardware System

In this section, we start with an overview of the EDIVA SoC system (Section 4.1). Then we describe the EDIVA accelerator for efficient EDIVA algorithm execution in Section 4.2. Finally, an image signal processor enhancement is introduced in Section 4.3 to further improve the system performance.

4.1 Motivation and System Overview

The EDIVA SoC design is motivated by the principle that executing the EDIVA algorithm should not repeatedly interrupt other latency-critical, always-on AR tasks that already occupy existing compute units. Frequent interruptions and subsequent recovery introduce unnecessary system latency and energy overheads [146]. This motivates the inclusion of a dedicated hardware unit capable of autonomously executing the EDIVA algorithm, rather than relying on a software implementation that contends for shared compute resources on every frame. Second, as detailed in Section 3, the EDIVA algorithm involves reprojecting multiple cached masks. At our target resolution, a single dense binary mask stored in byte-per-pixel format [76] occupies 410 KB. Storing 20 cached masks without compression requires 8.2 MB, forcing frequent off-chip DRAM accesses. Furthermore, image reprojection in EDIVA depends on the user movement and gaze location, which are non-deterministic, so the memory access in Equation 1 is irregular and data dependent, leading to severe memory divergence. This issue is exacerbated when the movement distance increases. This condition motivates us to design a dedicated memory space with data compression mechanism and structured memory access control. Third, because of the lack of support for preemption at reasonable granularity in edge GPU hardware [12], the context-switch latency is non-deterministic and may reach the millisecond level [48]. Offloading the computationally lightweight but latency-sensitive reprojection and refinement to a dedicated accelerator eliminates this contention, freeing GPU resources for compute-intensive tasks while guaranteeing deterministic reprojection latency.

Figure 7 illustrates the overall EDIVA AR SoC architecture. To speed up key EDIVA operations such as mask reprojection, cache search, and cache storage, we introduce the EDIVA accelerator,

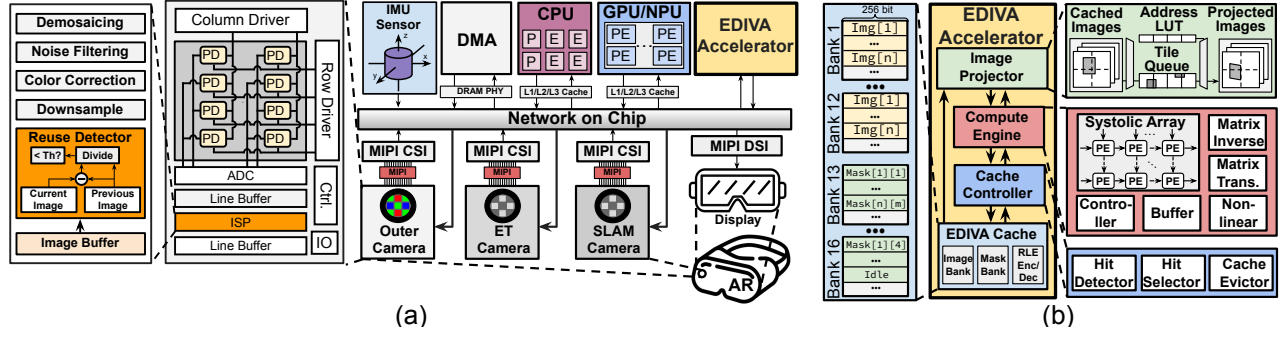


Figure 7: (a) An overview of the EDIVA AR SoC. The EDIVA accelerator is designed to integrate with the AR SoC. The outer camera ISP is enhanced with a reuse detector to accelerate the Direct Carryover check. (b) EDIVA accelerator design.

a plug-and-play hardware unit designed for seamless integration into AR SoCs. The SoC integrates an RGB outward-facing camera, a grayscale eye-tracking camera, a grayscale SLAM camera, and an IMU. Its compute resources comprise a CPU and GPU, along with an optional NPU. The display subsystem renders the scene and overlays segmentation masks onto the user's view. The image signal processor (ISP) handles early-stage image processing.

Figure 8 (a) illustrates the EDIVA SoC workflow. The system operates asynchronously, with different components running at their own frequencies. The mask is displayed following the video frame rate. The SLAM camera and IMU continuously supply data to the mono-inertial SLAM module, which estimates the user's position and orientation; the resulting pose estimates are stored in a pose information buffer. In parallel, the ET camera captures eye images, and the gaze-detection module continuously infers the user's gaze direction, storing the outputs asynchronously in a gaze buffer. These are standard features on modern AR devices.

Meanwhile, the RGB camera streams frames into the RGB input buffer, which are then consumed by the EDIVA accelerator, where the algorithm in Figure 6 (c) executes to generate the mask for display. When EDIVA determines that a new segmentation is needed, it triggers the segmentation DNN, which runs on either the GPU or the NPU. Otherwise, the mask for the current frame is generated directly within the EDIVA accelerator as described in Section 4.2. For every frame, the accelerator fetches the most recent position, RGB image, and gaze information from their buffers to ensure that processing always reflects the latest sensor inputs.

EDIVA supports binocular AR displays by computing the segmentation mask once and reprojecting it to both eye viewpoints during the display stage.

4.2 EDIVA Accelerator

EDIVA accelerator is composed of four major components: the *image projector*, *compute engine*, *cache controller*, and *EDIVA cache*, as shown in Figure 7 (b).

4.2.1 EDIVA Cache Design. EDIVA accelerator features its own scratchpad to store the EDIVA cache entries thus avoiding the shared system cache and DRAM access contention in the SoC. As illustrated in Section 3.4, each cache entry consists of an RGB image I_c , a segmentation mask M_c , and pose information U_c . M_c is a binary

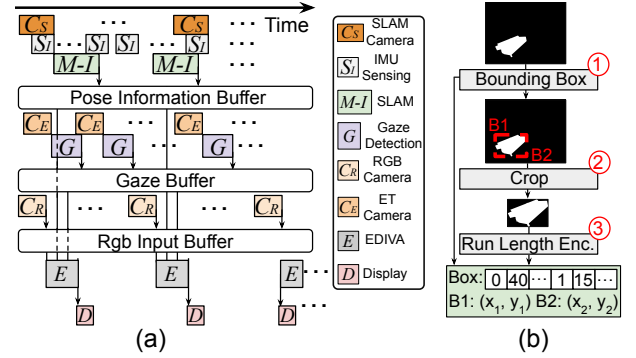


Figure 8: (a) EDIVA workflow in the SoC. (b) EDIVA cache mask compression steps.

map whose values are either 0 or 1. To reduce its storage footprint, we compress each mask using the steps illustrated in Figure 8 (b). In Step 1 of Figure 8 (b), a bounding-box check is applied to the binary mask to obtain the corner coordinates B1 and B2. In Step 2, the mask is cropped to the bounding-box region. We then further compress the cropped region using line-by-line run-length encoding (RLE), leveraging the fact that mask regions are typically spatially continuous. Finally, the RLE-encoded box together with B1 and B2 is stored as a mask entry in the EDIVA cache. On average, RLE provides a 17× reduction within the bounding box region, resulting in an overall 66× decrease in mask memory footprint, based on profiling across five sequences from the Aria Digital Twin and Aria Everyday Activities datasets. A lightweight hardware RLE encoder and decoder are implemented to perform compression and decompression efficiently.

The cache is organized into 16 banks, with 12 banks dedicated to RGB image storage and the remaining 4 banks used for masks, pose information, and associated metadata. The cache controller manages read/write operations, performs cache-entry hit detection, entry selection, and cache eviction.

4.2.2 Computation Engine and Image Projector. The computation engine, highlighted in the red block of Figure 7 (b), contains the hardware modules that accelerate geometry-based mask reprojection, depth estimation, and mask refinement. Specifically, a transposer

Dataset	Model	Img Size	FV	AD	AS	EDIVA
ADT	ESAM	640 ²	0.522/155	0.486/51	0.468/51	0.519/45
	PSPNet	1408 ²	0.489/185	0.426/61	0.375/61	0.478/54
	SegFormer	640 ²	0.524/178	0.483/59	0.441/59	0.517/49
	HRNet	704 ²	0.594/178	0.559/59	0.523/59	0.582/51
AEA	ESAM	640 ²	0.502/155	0.478/78	0.414/78	0.495/60
	PSPNet	1408 ²	0.464/185	0.418/93	0.385/93	0.458/74
	SegFormer	640 ²	0.476/178	0.447/89	0.370/89	0.465/83
	HRNet	704 ²	0.527/178	0.501/89	0.452/89	0.516/66
HD	ESAM	640 ²	0.455/155	0.427/78	0.384/78	0.444/62
	PSPNet	1408 ²	0.436/185	0.388/93	0.353/93	0.419/74
	SegFormer	640 ²	0.488/178	0.439/89	0.402/89	0.471/67
	HRNet	704 ²	0.493/178	0.462/89	0.407/89	0.488/69
Ego-exo4D	ESAM	640 ²	0.562/155	0.524/78	0.497/78	0.553/58
	HRNet	704 ²	0.544/178	0.518/89	0.483/89	0.537/65

Table 2: Accuracy results on video segmentation. The two numbers before and after "/" show the IoU and GFLOPs.

module supports operand transposition, and the Gauss–Jordan method [9] is employed to compute matrix inverses. We implement the transposer module and matrix inversion module from scratch in RTL. A systolic array executes the neural networks for the depth estimation module and mask refinement module, as well as the transformation-matrix computations. The non-linear unit handles operations including activation functions and normalization. Following prior work [82, 86, 101, 124], we implement Taylor approximation to accelerate computation.

The image projector is responsible for pixel-wise value assignment according to the reprojection results. An address look-up table (LUT) is configured based on the coordinate transformation derived from Equation 1. Each mask is spatially partitioned into tiles that are processed sequentially within the accelerator. Multiple virtual channels share the address LUT and the tile queue, enabling concurrent processing of tiles with varying and irregular memory accesses. The pixels are prefetched in a finer granularity based on the coordinates computed from the compute engine, thus improving the memory bandwidth efficiency.

4.3 ISP Enhancement

The image signal processor (ISP) is an integral component of the image sensor subsystem in AR SoCs [104, 131]. With the growing interest in in-sensor and near-sensor computation, the ISP has become a promising candidate for customization to accelerate future AR/VR workloads [10, 63, 104, 111].

In EDIVA, part of the Direct Carryover check (Step 4 in Figure 6 (c)) is offloaded to ISP, which computes the pixel-wise difference between the previous image I_{t-1} and the current image I_t . This design choice is motivated by two factors. First, performing this operation on the SoC would require transferring both high-resolution frames over MIPI interface; as shown in Figure 3 (d), MIPI communication incurs large latency and energy overhead. Executing this step within the ISP eliminates the need for these large data transfers. Second, pixel difference computation is lightweight and integrates naturally into the ISP’s lightly loaded processing pipeline, making it well suited for efficient near-sensor execution.

Following the in-sensor compute design in [35], we add an on-chip image buffer in the ISP to store the previous RGB image I_{t-1} ,

as shown in the left part of Figure 7 (a). As new RGB values of I_t are read out from the photodiode and ADC, each pixel is compared against its counterpart through a simple subtraction and thresholding operation. If the pixel-wise difference stays below a predefined threshold γ , the RGB image is not transferred to the SoC; instead, the ISP sends a lightweight acknowledgment signal indicating that no mask update is required.

5 EDIVA Accuracy Evaluation

In this section, we assess EDIVA’s instance segmentation accuracy across three datasets: Aria Digital Twin (ADT) [81], Aria Everyday Activities (AEA) [66], and HD-EPIC (HD) [83]. Their video frame rates are 30 Hz, 20 Hz, and 30 Hz, respectively. The ADT, AEA, and HD datasets are collected with Aria Glasses [32] which reflects real AR use cases. We also conduct experiments on the Ego-Exo4D dataset [45], which contains dynamic scenes in both indoor and outdoor environments. The video frame rate is 30 Hz.

For the segmentation module in Figure 6 (c), we evaluate EDIVA using several pretrained networks, including EfficientSAM-Small (ESAM-S) [127], PSPNet-ResNet101 [141], SegFormer-B2 [125], and HRNet-W32 [116]. The image size for each model is set to 640×640 , 1408×1408 , 640×640 , and 704×704 , respectively. The position distance threshold α , pixel-difference threshold γ for reuse, and pixel-difference threshold τ for reprojection in Section 3.6 are set to 0.05 m, 5%, and 10%, respectively. The cache size in EDIVA algorithm is set to 20. The DNNs within Mask Refinement Module and Depth Estimation Module are trained on 20 video sequences in each dataset for 10 epochs. The evaluation is conducted on 10 test video sequences covering different scenarios.

To evaluate EDIVA, we compare EDIVA against the original ESAM-S, HRNet, PSPNet, and SegFormer models, which perform segmentation on every frame; we refer to this baseline as Full Video (FV). Additionally, we develop two baseline algorithms. The first, Average Downsampling (AD), reduces computation by applying average downsampling to the input image. The second, Average Skipping (AS), periodically skips segmentation and reuses the most recent mask for the skipped frames. We adopt *Intersection over Union (IoU)*, a metric used to measure the overlap between the predicted segmentation map and the ground truth map.

5.1 Segmentation Accuracy

We assess EDIVA’s segmentation performance while keeping the FLOP budget roughly consistent across all baselines for each dataset model pair. For the AD baseline, images are downsampled by 3×, 2×, 2×, and 2× for the ADT, AEA, HD-EPIC, and Ego-Exo4D datasets, respectively. For the AS baseline, segmentation is performed every 3, 2, 2, and 2 frames for ADT, AEA, HD-EPIC, and Ego-Exo4D, respectively. As shown in Table 2, EDIVA consistently achieves higher IoU than AD and AS baseline methods while using significantly less computation. On average, compared with FV, EDIVA reduces the computational cost by 71%, 59%, 61%, and 63% on the ADT, AEA, HD-EPIC, and Ego-Exo4D, respectively. This efficiency gain comes from EDIVA’s ability to exploit strong temporal correlations between consecutive masks, substantially reducing redundant computation across frames.

Model	W/O Refine	W/ Refine
ESAM	0.511/42	0.519/45
PSPNet	0.456/50	0.478/54
SegFormer	0.481/46	0.517/49
HRNet	0.534/47	0.582/51

Table 3: Impact of Mask Refinement on ADT.

#Entries	ADT/HRNet	AEA/PSPNet
1	0.591/102	0.463/113
10	0.588/70	0.461/86
20	0.582/51	0.458/74
50	0.580/49	0.453/68

Table 4: Impact of the number of EDIVA cache entries.

Dataset	Model	EDIVA-F	EDIVA-D	EDIVA
Aria Digital Twin	ESAM-S	0.492/46	0.505/95	0.519/45
	HRNet-W32	0.556/53	0.571/101	0.582/51

Table 5: Depth estimation module design choice.

Method	Boundary F1 ↑			IoU Change ↓			LPIPS ↓			SSIM ↑		
	ADT	AEA	HD	ADT	AEA	HD	ADT	AEA	HD	ADT	AEA	HD
FV	0.36	0.26	0.31	0.02	0.04	0.02	0.026	0.058	0.042	0.981	0.973	0.979
AD	0.29	0.18	0.23	0.05	0.10	0.09	0.054	0.073	0.069	0.964	0.955	0.962
AS	0.23	0.16	0.19	0.08	0.12	0.12	0.076	0.085	0.072	0.951	0.947	0.956
EDIVA	0.34	0.22	0.28	0.03	0.05	0.04	0.034	0.065	0.053	0.976	0.968	0.972

Table 6: Video segmentation perceptual quality results.

5.2 Comparison with Temporal-reuse Baselines

Prior works [64, 144] explore the temporal redundancy in computer vision tasks. Here we provide a comparison against Deep Feature Flow (DFF) [144], which reuses intermediate features to improve segmentation efficiency. We evaluate DFF with the ResNet-101 [50] as segmentation backbone and FlowNet [37] for feature reuse on the ADT dataset, finetuned under the same setting as EDIVA. ResNet-101 and FlowNet require 220 and 17 GFLOPs respectively. We also use the ResNet-101 as the segmentation backbone in the EDIVA algorithm as a baseline. We set the DFF keyframe interval to 6 frames so that its computation cost is similar to EDIVA with the ResNet-101 backbone. Under this matched budget, DFF reaches 0.446 IoU at 52.9 GFLOPs, whereas EDIVA reaches 0.487 IoU at 48.8 GFLOPs. EDIVA achieves higher accuracy at lower cost for two reasons. First, DFF’s fixed-keyframe feature propagation reuses from a temporally fixed source with no reuse quality check, so the propagation error accumulates; EDIVA instead reprojects the best-matching cached result and verifies each reuse, avoiding this degradation. Second, DFF’s FlowNet adds a non-trivial 17.1 GFLOPs for flow estimation, whereas EDIVA obtains motion directly from device pose and IMU at negligible cost with a low cost DMG module.

5.3 Impact of Mask Refinement

In Section 3.3, we introduce a lightweight mask-refinement module to enhance the quality of reprojected masks. We evaluate EDIVA with and without this module on the ADT dataset using different segmentation models, with results shown in Table 3. The refinement step adds only 3.5 GFLOPs regardless of image size and improves IoU by an average of 5.7% across all models, which demonstrates the importance of the refinement module in EDIVA.

5.4 Reprojection Error Analysis

In Section 3.2, we introduce a lightweight depth estimation module which predicts a single average depth value and reuses it across all pixels within the IOI mask for reprojection. To quantify the impact of using a single average depth to reproject the masks of objects with significant depth variation and to evaluate whether the refinement module can improve the reprojected mask quality, we develop three baselines. The first baseline conducts mask reprojection using the object’s ground-truth depth map without using the refinement module. The second baseline conducts reprojection using predicted average depth without using the refinement module. The third baseline is EDIVA. We evaluate these three baselines on ADT and AEA dataset using ESAM. The IoU of baseline 1, baseline 2, and EDIVA are 0.522, 0.511, and 0.519 on ADT, and 0.493, 0.462, and 0.495 on AEA, respectively. The results show that using ground truth depth improves mask quality even without a refinement module. However, after introducing the refinement module, the mask quality

is largely recovered even when using predicted average depth. In the AEA dataset, EDIVA can outperform the ground truth depth baseline. This is because objects may move between timesteps, and simply reprojecting cached masks, even with ground truth depth, can result in misalignment. With a refinement module, EDIVA improves the mask quality and achieves the best performance.

5.5 Impact of the Number of Cache Entries

As described in Section 3.4, EDIVA maintains a cache storing images, positions, and masks from previous frames. Increasing the number of EDIVA cache entries raises the likelihood of successful mask reprojection but also incurs higher search and storage overhead. We evaluate IoU and GFLOPs under varying cache sizes using HRNet on the ADT dataset and PSPNet on the AEA dataset, with results shown in Table 4. We find that a cache entry number of 20 achieves the best balance between computational cost and segmentation accuracy. For the evaluation, we fix the cache entry number and report performance averaged across each dataset.

5.6 Impact of Depth Estimation and Mask Refinement Module

In Section 3.2, we introduce the lightweight design of the depth estimator to reproject a cached mask and a mask refinement module to improve the reprojected mask quality. To justify this choice, we replace the EDIVA depth estimator and the refinement modules with FastDepth [120] and Depth Anything V2 Small [130] within the EDIVA pipeline, denoted as EDIVA-F and EDIVA-D, and evaluate them on the ADT dataset using both ESAM-S and HRNet segmentation models. As shown in Table 5, EDIVA’s depth estimator achieves a better accuracy-efficiency tradeoff in IoU and computational cost. These results also highlight the limited impact of depth-prediction errors on EDIVA’s segmentation accuracy, because the refinement module improves the reprojected masks, while the image-difference comparison mechanism filters out low-quality masks and triggers a new segmentation when necessary.

5.7 Perceptual Quality Evaluation of Generated Masks

To evaluate the perceptual quality of EDIVA-generated masks, we measure boundary accuracy, temporal consistency, and visual fidelity. For boundary accuracy, we report the Boundary F1-score [84], which evaluates segmentation quality along object boundaries. For

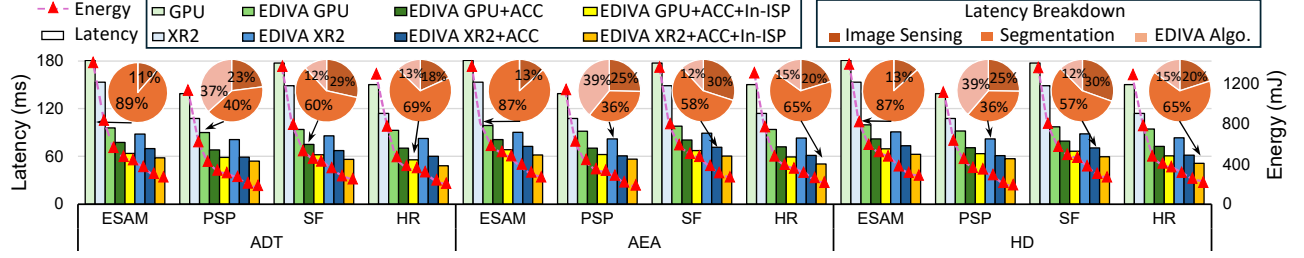


Figure 9: Latency (bar) and energy (red lines) evaluation across different methods. Pie charts show the latency breakdown.

temporal consistency, we measure IoU change [84], a DAVIS temporal stability metric that quantifies the change in IoU from the first to the last temporal quartile of a video segment. We define a fixation duration as a video segment and report the average IoU change across all segments in each video. For visual fidelity evaluation, we evaluate Learned Perceptual Image Patch Similarity (LPIPS) [138], and Structural Similarity Index Measure (SSIM) [118].

Table 6 shows the results on different baselines across datasets and models. EDIVA consistently achieves higher Boundary F1 scores and lower IoU change than both the AD and AS baselines, demonstrating superior boundary accuracy and stronger temporal consistency across frames. EDIVA also achieves lower LPIPS and higher SSIM than AD and AS, confirming that the masks produced by EDIVA visually align to the objects better than AD and AS.

6 Hardware Performance Evaluation

The proposed EDIVA accelerator described in Section 4.2 consists of a $2D\ 16 \times 16$ systolic array, non-linear units, control logic, and essential interfaces, all implemented in SystemVerilog, with frequency set to 1GHz. The components are synthesized using Synopsys Design Compiler [106] and evaluated through full RTL cycle-level simulation to measure area, timing, and power, based on 45nm CMOS technology [1]. A 2MB unified, double-buffered SRAM array holds the EDIVA cache entries, model weights, and activations. The on-chip SRAM bandwidth is set to 128GB/s and is modeled using Destiny [85]. Synthesis results were scaled to 22 nm using DeepScaleTool [96]. The total area is $5.9mm^2$, where on-chip buffers dominate (64%), followed by the computation engine (31%), and control logic (5%). The EDIVA accelerator power is 464mW and the power breakdown of the computation engine, SRAM, and control logic is 22%, 74%, and 4%.

We evaluate segmentation DNN execution on two platforms: GPU performance on NVIDIA Jetson Orin NX [4] using the Tegra System Profiler [77], and NPU performance on the Qualcomm XR2 Gen2 [88] using the Qualcomm AI Hub toolkit [87]. Qualcomm AI Hub profile jobs evaluate models by running them on physical devices in the cloud instead of simulation or analytical modeling. XR2 power is estimated by scaling GPU power according to the NPU–GPU power ratio reported in prior studies [89, 108]. The image sensor power is estimated at 385 mW, with the ISP contributing 164 mW, based on the measurements from previous studies [67, 69, 146] and commercial sensor data [97–99]. For the ISP enhancement, an additional 480 KB image buffer is added to store the previous input frame I_{t-1} . This buffer consumes 98 mW

of power, consistent with prior in-ISP computation designs [10, 53]. The comparison logic takes 2.4mW power. The latency of the image sensor pipeline is set to 5 ms, reflecting typical image sensor designs under normal lighting conditions [61, 71, 100]. The MIPI interface [3] operates at 10 Gbps (four lanes).

We evaluate the system latency and energy of the EDIVA using HRNet, PSPNet, SegFormer, and ESAM-S as segmentation models to execute the instance segmentation tasks on the ADT, AEA, and HD-EPIC datasets with identical algorithmic settings in Section 5. The reported latency and energy values are averaged over all frames in the videos. We disable the saccade detection, ensuring no frame skip from saccade. The hardware/software interface of EDIVA is narrow and confined to the driver layer.

6.1 System Evaluation Results

EDIVA is a joint optimization framework that includes innovative algorithmic design (EDIVA algorithm) and novel hardware design (EDIVA accelerator and ISP enhancement). To evaluate the individual contributions of each part to the overall system performance, we use several baseline approaches. The first baseline, **GPU**, transmits every frame from the image sensor to the SoC and executes the segmentation model to process every frame on the Jetson Orin NX GPU. The second baseline, **EDIVA GPU**, executes the EDIVA algorithm to skip redundant segmentation. Both the segmentation model and the EDIVA algorithm are executed on the Jetson Orin NX GPU, with every frame transmitted from the image sensor to the SoC. The third baseline, **EDIVA GPU+ACC**, executes the EDIVA algorithm in the EDIVA accelerator, with other settings the same as EDIVA GPU baseline. The fourth baseline, **EDIVA GPU+ACC+In-ISP**, additionally enables the ISP enhancement. Similarly, we also implement all segmentation models on the Qualcomm XR2 Gen2 NPU instead of the GPU, resulting in four additional baseline algorithms: **XR2**, **EDIVA XR2**, **EDIVA XR2+ACC**, and **EDIVA XR2+ACC+In-ISP**.

Figure 9 compares the above methods in terms of average latency and energy consumption to process video stream on multiple segmentation models and datasets. On average, EDIVA GPU+ACC+In-ISP and EDIVA XR2+ACC+In-ISP achieve $2.67\times$ speedup and $3.65\times$ energy saving compared to the baseline algorithm (**XR2** and **GPU**) where every frame is segmented across different configurations, while achieving better segmentation accuracy on IOI as shown in Table 2 compared to other baselines. Among all configurations, EDIVA GPU+ACC+In-ISP and EDIVA XR2+ACC+In-ISP achieve the lowest latency, staying within the 50 to 70 ms range required

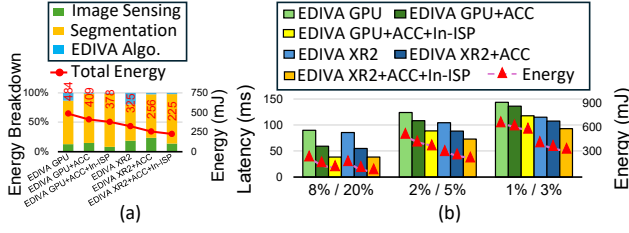


Figure 10: (a) Energy consumption and breakdown across different baselines. (b) Impact of image difference thresholds. The two numbers before and after "/" show the γ and τ .

for a fluid visual experience. EDIVA's strong performance comes from its ability to skip temporally redundant frames and from the specialized AR SoC hardware that incorporates the EDIVA accelerator and ISP enhancement. In addition, as shown in Figure 8 (a), the accelerator accesses the most recent pose and gaze data each frame, allowing SLAM and gaze detection latency to be hidden.

Compared to the GPU and XR2 baselines, EDIVA GPU and EDIVA XR2 reduce latency by 1.75 $\times$ and 1.59 $\times$, and energy consumption by 2.72 $\times$ and 2.34 $\times$, respectively, demonstrating that the EDIVA algorithm effectively alleviates the latency and energy bottlenecks of segmentation by carrying forward cached masks. Moreover, EDIVA GPU+ACC and EDIVA XR2+ACC reduce latency by 1.27 $\times$ and 1.31 $\times$, and energy by 1.21 $\times$ and 1.31 $\times$, respectively, compared with the EDIVA GPU and EDIVA XR2 baselines, demonstrating that the EDIVA accelerator substantially lowers system latency and energy. Lastly, during Direct Carryover check, an average of 18% of frames can be directly reused, eliminating projection checks and segmentation. With ISP enhancement, these frames also avoid MIPI transfer, DRAM access, and image-difference computation. As a result, ISP enhancement provides an additional 1.26 $\times$ and 1.22 $\times$ latency reduction and 1.12 $\times$ and 1.20 $\times$ energy reduction over the EDIVA GPU+ACC and EDIVA XR2+ACC baselines, respectively.

Figure 10 (a) shows the average energy breakdown and total energy consumption of the ESAM model on the ADT dataset. The results show that introducing the EDIVA accelerator significantly lowers energy consumption compared with executing the algorithm solely on the GPU or on XR2. Overall, EDIVA GPU+ACC+In-ISP and EDIVA XR2+ACC+In-ISP achieve the lowest energy consumption, largely due to the significant reduction in input transfer power.

6.2 Comparison with Existing Hardware Accelerator

Euphrates [146] is a hardware accelerator that exploits the motion vectors in the SoC's vision pipeline to extrapolate results from a key frame to subsequent frames, thereby skipping full neural network execution on non-key frames. We reimplement this mechanism within our evaluation framework: the segmentation backbone (ESAM) is executed on key frames, and the resulting segmentation mask is warped to subsequent frames using motion vectors, with the motion-vector computation latency and power taken from the values reported in the original paper. All other settings, dataset (ADT), backbone, and hardware platform, are identical to EDIVA, so that the comparison isolates the reuse mechanism. Following

Configuration	Lat.	Eng.	IoU
Euphrates [146]	112.4	458.3	0.483
EDIVA XR2+ACC+In-ISP	61.4	225.4	0.519

Table 7: Comparison against Euphrates. Units of latency (Lat.) and energy (Eng.) are millisecond and millijoule.

ADT ESAM-S		ADT HRNet	
OF GPU	EDIVA GPU	OF GPU	EDIVA GPU
127ms / 615mJ	105ms / 484mJ	121ms / 453mJ	102ms / 369mJ

Table 8: Comparison with optical flow based method.

Euphrates, the warped mask is accepted without any validity check. We set the propagation window so that Euphrates attains a segmentation accuracy comparable to EDIVA, allowing a like-for-like comparison of latency and energy at matched accuracy.

Table 7 reports the latency, energy, and accuracy of Euphrates and EDIVA XR2+ACC+In-ISP. Euphrates incurs 1.83 $\times$ higher latency and 2.03 $\times$ higher energy than EDIVA. Because its reuse is confined to adjacent key frames, Euphrates cannot exploit the long-span temporal redundancy that arises when the user revisits a previously observed region, and therefore triggers full segmentation more frequently than EDIVA. It also lacks gaze information and thus cannot skip frames during saccadic motion. In addition, Euphrates attains lower accuracy, as motion-vector-based warping is applied without any validity check or refinement, so propagated masks degrade under larger viewpoint change; EDIVA instead verifies and refines each reuse, and re-segments when the check fails.

6.3 Comparison to Optical Flow Method

For comparison, we implement an optical flow baseline (OF) by replacing EDIVA's pose based reprojection with dense optical flow warping within the same EDIVA framework. We use the Farneback method [33], a dense optical flow algorithm widely used in real time vision pipelines, while keeping all other components identical to EDIVA. We evaluate the OF baseline under the same settings as the EDIVA GPU baseline, denoted as OF GPU. Table 8 reports the latency and energy of OF and EDIVA baselines on the ADT dataset using ESAM and HRNet. Due to the overhead of the Farneback method in reprojection checks, the OF GPU baseline increases latency and energy by 1.20 $\times$ and 1.25 $\times$ compared to EDIVA GPU.

6.4 Impact of EDIVA Accelerator

Current AR SoCs include neural processing units (NPUs) to accelerate neural network execution [88]. The DNN portions of EDIVA (depth estimation, mask refinement) can also run on an NPU. However, the operations the EDIVA accelerator targets are not only the regular DNN layers but also the irregular mask reprojection, RLE (de)compression, and cache hit-detection/eviction, whose data-dependent, scattered access patterns map poorly onto an NPU's regular systolic dataflow. To evaluate this directly, we add an intermediate baseline, EDIVA XR2+Reproj Acc, which runs the DNN modules on the XR2 NPU but offloads only the irregular reprojection, RLE, and cache operations to the accelerator, and compare it against EDIVA XR2 and EDIVA XR2+ACC baselines.

Configuration	ESAM		PSP		SF		HR	
	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.
EDIVA XR2	98.1	324.8	81.3	240.0	103.2	344.6	92.4	245.1
EDIVA XR2+Reproj Acc	82.6	277.9	68.1	189.4	86.4	286.3	73.5	194.7
EDIVA XR2+ACC	77.4	255.8	60.8	171.6	80.2	267.9	67.3	175.9

Table 9: Impact of EDIVA accelerator. Units of latency (Lat.) and energy (Eng.) are millisecond and millijoule.

Baseline	Saccade	ADT		AEA	
		Lat.	Eng.	Lat.	Eng.
EDIVA GPU+ACC+In-ISP	W/O	57	255	61	271
	W/	51	227	45	227
EDIVA XR2+ACC+In-ISP	W/O	51	137	53	146
	W/	46	122	44	123

Table 10: Impact of Saccade Detection. Units of latency (Lat.) and energy (Eng.) are millisecond and millijoule.

As shown in Table 9, the comparison between EDIVA XR2 and EDIVA XR2+Reproj Acc shows that moving only the irregular operations off the NPU and onto the accelerator already reduces latency by $1.21\times$ and energy by $1.22\times$ on average across ESAM, PSP, SF, and HR, even though the DNN layers still run on the XR2 NPU. Offloading the complete EDIVA algorithm to the accelerator yields a $1.32\times$ latency and $1.34\times$ energy reduction over the NPU-only baseline. These results confirm that the accelerator’s benefit comes specifically from handling the irregular, memory-divergent operations that the NPU executes inefficiently, not from the DNN computation, which the NPU already supports.

6.5 Impact of Saccade Occurrence

The previous evaluation does not include the hardware benefits from saccadic motion. With saccade detection enabled, frames during rapid eye or head movements can be skipped. We evaluate this effect for the EDIVA GPU+ACC+In-ISP and EDIVA XR2+ACC+In-ISP baselines on the ADT and AEA datasets using HRNet. Based on the criteria presented in Section 3.5, 11% and 16% of frames are classified as saccades and skipped, leading to additional reductions in latency and energy as shown in Table 10. We note that even without saccadic motion, EDIVA already delivers strong performance.

6.6 Impact of Thresholds

In this section, we analyze how the image difference thresholds τ and γ used in the Direct Carryover and projection checks (Section 3.6) affect system performance. We measure the latency and energy consumption of EDIVA using HRNet on the ADT dataset under different γ - τ settings, with results shown in Figure 10 (b). As the threshold values decrease, more frames require segmentation, leading to higher latency and energy consumption. This trend is consistent across all baseline methods. EDIVA adopts the threshold settings that optimize the balance of hardware performance and accuracy.

Configuration	ESAM		PSP		SF		HR	
	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.
GPU+Carryover	174.6	1124.3	102.4	794.8	235.9	1497.5	129.1	953.6
EDIVA GPU	105.9	484.1	88.8	354.9	109.0	495.0	102.7	369.9
EDIVA GPU+A.+I.	65.6	378.6	54.5	253.2	67.3	374.4	57.4	255.2

Table 11: Impact of Direct Carryover Check. Units of latency (Lat.) and energy (Eng.) are millisecond and millijoule. EDIVA GPU+A.+I. denotes EDIVA GPU+ACC+In-ISP.

Configuration	ESAM		PSP		SF		HR	
	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.	Lat.	Eng.
EDIVA GPU+ACC w/o Comp.	97.9	495.3	77.7	328.5	99.2	501.6	89.9	361.2
EDIVA GPU+ACC	85.1	409.3	68.2	280.8	86.0	411.9	77.5	294.9

Table 12: Impact of RLE compression.

6.7 Impact of Direct Carryover Check

To isolate the benefit of Direct Carryover check introduced in Section 3.6, we introduce **GPU+Carryover** baseline, which implements only the Direct Carryover check on the GPU without the EDIVA accelerator. This baseline separates the contribution of simple reuse from the rest of EDIVA.

Using the same settings as Section 6.1, we evaluate GPU + Carryover, EDIVA GPU, and EDIVA GPU+ACC+In-ISP (EDIVA GPU+A.+I.) with ESAM, PSPNet, SegFormer, and HRNet. As shown in the Table 11, EDIVA GPU reduces latency by $1.56\times$ and energy by $2.54\times$ over GPU+Carryover, showing that reprojection recovers many frames that carryover-only reuse would re-segment. EDIVA GPU+ACC+In-ISP further improves this to $2.58\times$ latency and $3.46\times$ energy reduction, confirming that EDIVA’s gains primarily come from reprojection and dedicated hardware rather than simple reuse.

6.8 Impact of RLE Mask Compression

RLE Compression introduced in Section 4.2.1 is particularly effective for masks because they are binary and spatially contiguous compared to RGB images. We cache a 4×4 downsampled version of the RGB image for depth estimation (Section 3.2) to reduce storage overhead. To quantify the benefit of RLE mask compression, we add a baseline, **EDIVA GPU+ACC w/o Compression**, which is identical to EDIVA GPU+ACC but stores cached masks uncompressed. We evaluate both configurations on the ADT dataset using ESAM, PSPNet, SegFormer, and HRNet. As shown in Table 12, removing compression increases per-frame latency by $1.15\times$ and energy by $1.21\times$ on average. With RLE compression, the $66\times$ reduction in mask footprint allows the full working set of cache entries to remain resident in the 2 MB SRAM, converting off-chip accesses into on-chip hits. This shows that compression is not merely a storage optimization, but is necessary to keep the reprojection pipeline free from DRAM bottlenecks.

6.9 User Study

To evaluate the combined impact of segmentation quality and latency, we conducted a user study on a Meta Quest Pro comparing

EDIVA and the FV baseline using HRNet. We selected four fifteen-second clips from the AEA dataset that reflect typical AR scenarios. For each clip, all frames were segmented offline by both methods, producing two mask sets with distinct accuracy characteristics consistent with our algorithmic evaluation. During the study, each clip was replayed on the Quest Pro while the eye tracker recorded participants' gaze. For each frame, the instance mask at the gaze location was rendered as an overlay. To emulate end-to-end latency, we apply method-specific delays based on our measurements (51 ms for EDIVA and 160 ms for FV): the mask shown at time t corresponded to the gaze position at $t - \Delta$. As a result, the two conditions differ in both mask accuracy and the delay between gaze and the displayed mask.

Sixteen participants completed the study in a seated configuration with hand controllers. We employ a two-interval forced-choice (2IFC) design [133]. In each trial, a single AEA clip was shown twice, once with our method and once with the baseline. On each trial, the order of which method corresponded to A or B was randomized. After viewing both, they select the condition that provided the better overall experience, considering both mask quality and segmentation delay. Each participant completed 32 trials (4 clips $\times$ 2 orders $\times$ 4 repeats). Across all 512 trials, our method was chosen 486 times ($94.9\% \pm 5.1\%$), as summarized in Figure 11. A one-sided binomial test against random choice ($p = 0.5$) shows that this preference is highly significant ($p \ll 10^{-10}$), indicating a strong user preference for our method over the baseline. Overall, results show a much better user experience of EDIVA.

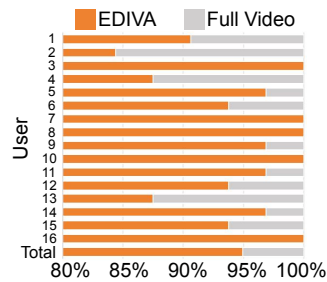


Figure 11: Aggregate user choice results.

7 Discussion

7.1 Importance of Accelerating Segmentation

Segmentation is a fundamental building block in AR systems, supporting downstream tasks such as object recognition, optical character recognition (OCR), scene understanding, and multimodal reasoning. To provide a visually seamless user experience, prior studies suggest that the end-to-end latency should remain within 50–70 ms [6]. However, Table 1 shows that segmentation alone already exceeds this budget, with latency ranging from 252 ms to 5,462 ms.

To further demonstrate its impact, we profile representative OCR pipelines on Jetson Orin NX. In the OCR pipeline, ESAM-S [127] first segments the text region from a 640×640 image, which is then cropped and passed to TrOCR-base [58] for recognition. Segmentation takes 788 ms and 5028 mJ, while TrOCR-base takes only 81 ms and 615 mJ, making segmentation the primary latency and energy bottleneck. We also evaluate a foundation model pipeline (SmolVLM-256M [70]) on a video question-answering task with a 640×640 image and 156 prompt tokens, which takes 2620 ms and 23318 mJ. Although the foundation model inference has a higher

per-query cost, segmentation still contributes a substantial portion of the computation. More importantly, segmentation runs on every frame, whereas the foundation model is triggered only intermittently, making segmentation the dominant continuous workload for real-time AR.

EDIVA is presented in the context of segmentation. However, EDIVA's key idea can also extend to other pixel-level AR tasks, such as object detection. For object detection, reprojection can carry forward bounding boxes or keypoints across frames, which are cheaper to reproject than dense masks, and the detector only needs to run again when the reprojection-validity check fails.

7.2 Contribution Beyond Prior Work

Temporal correlation has been exploited in prior video perception systems such as Deep Feature Flow [144] and NetWarp [39], while ReFrame [64] leverages inter-frame similarity to reduce DNN processing overhead. In contrast, EDIVA is not simply an AR-specific application of feature or mask warping. It uses gaze, pose, depth, and quality checks to identify, reproject, and validate cached segmentation masks. This enables instance-level mask reuse across non-adjacent frames when the user returns to a previously viewed region, while skipping the entire segmentation network. Unlike Deep Feature Flow and NetWarp, EDIVA is plug-and-play and does not require retraining a dedicated propagation network. ReFrame, in contrast, reuses intermediate activations to skip layers within the network, whereas EDIVA reuses final segmentation masks and can therefore skip the entire segmentation network. FovealSeg [135] also explores segmentation mask reuse; however, EDIVA differs from FovealSeg in several key aspects. EDIVA uses pose-based filtering for Direct Carryover and jointly models eye and head motion for saccade detection, reducing unnecessary segmentation triggered by head motion. Beyond temporal reuse, EDIVA exploits multimodal signals available on AR devices to enable geometry-based mask reprojection, refinement, and verification across viewpoints.

Prior works [20, 34, 36, 102, 134, 142, 143] exploit temporal redundancy for hardware acceleration. Beyond its AR-specific contributions, EDIVA differs by maintaining multiple historical segmentation results and selecting the cached result that best matches the current viewpoint through geometric reprojection and quality checks, rather than relying on an adjacent frame or fixed key frame. This allows reuse across arbitrary temporal intervals when the user returns to a previously viewed region, reducing the number of full segmentations while maintaining accuracy.

8 Conclusion

Image segmentation is critical for AR but running it every frame is costly. EDIVA reduces this overhead by exploiting user behavior patterns to avoid unnecessary segmentation, improving hardware efficiency. Its core idea can also extend to other pixel-level AR tasks such as object detection or optical flow, suggesting a broader design approach that aligns AI processing with human attention.

Acknowledgement

We acknowledge Meta Reality Labs for their valuable technical discussions and support of this work.

References

- [1] [n. d.]. Nangate freepdk45 open cell library. <https://silvaco.com/services/library-design/>
- [2] [n. d.]. Vive Pro 2. <https://www.vive.com/us/product/vive-pro2/specs/>.
- [3] [n. d.]. What is Mobile Industry Processor Interface (MIPI) Protocol? <https://www.synopsys.com/blogs/chip-design/what-is-mobile-industry-processor-interface-protocol.html>
- [4] 2023. NVIDIA Jetson Orin NX 8 GB Specs. TechPowerUp GPU Database. <https://www.techpowerup.com/gpu-specs/jetson-orin-nx-8-gb.c4081>
- [5] Abdullah M Al-Ansi, Mohammed Jaboo, Askar Garad, and Ahmed Al-Ansi. 2023. Analyzing augmented reality (AR) and virtual reality (VR) recent development in education. *Social Sciences & Humanities Open* 8, 1 (2023), 100532.
- [6] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. 2017. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)* 14, 4 (2017), 1–13.
- [7] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. 2017. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *British machine vision conference*, Vol. 1.
- [8] John Y Aloimonos. 1990. Perspective approximations. *Image and Vision Computing* 8, 3 (1990), 179–192.
- [9] Steven C Althoen and Renate McLaughlin. 1987. Gauss-Jordan reduction: A brief history. *The American mathematical monthly* 94, 2 (1987), 130–142.
- [10] Hyochan An, Sam Schiferl, Siddharth Venkatesan, Tim Wesley, Qirui Zhang, Jingcheng Wang, Kyojin D Choo, Shiyu Liu, Bowen Liu, Ziyun Li, et al. 2020. An ultra-low-power image signal processor for hierarchical image recognition with deep neural networks. *IEEE Journal of Solid-State Circuits* 56, 4 (2020), 1071–1081.
- [11] Dora E Angelaki and Kathleen E Cullen. 2008. Vestibular system: the many facets of a multimodal sense. *Annu. Rev. Neurosci.* 31, 1 (2008), 125–150.
- [12] Michael Antonov. 2015. Asynchronous TimeWarp Examined. Meta Developer Blog. <https://developers.meta.com/horizon/blog/asynchronous-timewarp-examined/>
- [13] Apple Inc. 2024. Apple Vision Pro. <https://www.apple.com/apple-vision-pro/>
- [14] Simon Baker, Daniel Scharstein, James P Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. 2011. A database and evaluation methodology for optical flow. *International journal of computer vision* 92, 1 (2011), 1–31.
- [15] Jesús Bermejo-Berros and Miguel Angel Gil Martinez. 2021. The relationships between the exploration of virtual space, its presence and entertainment in virtual reality, 360° and 2D. *Virtual Reality* 25, 4 (2021), 1043–1059.
- [16] Jing Bi, Nguyen Manh Nguyen, Ali Vosoughi, and Chenliang Xu. 2023. Misar: A multimodal instructional system with augmented reality. *arXiv preprint arXiv:2310.11699* (2023).
- [17] Birtukan Birawo and Pawel Kasprowski. 2022. Review and evaluation of eye movement event detection algorithms. *Sensors* 22, 22 (2022), 8810.
- [18] Bert De Brabandere, Davy Neven, and Luc Van Gool. 2017. Semantic Instance Segmentation with a Discriminative Loss Function. *ArXiv abs/1708.02551* (2017). <https://api.semanticscholar.org/CorpusID:2328623>
- [19] Adolfo M. Bronstein, M. Patel, and Q. Arshad. 2015. A brief review of the clinical anatomy of the vestibular-ocular connections—how much do we know? *Eye* 29 (2015), 163–170. doi:10.1038/eye.2014.262
- [20] Mark Buckler, Philip Bedoukian, Suren Jayasuriya, and Adrian Sampson. 2018. EVA²: Exploiting temporal redundancy in live computer vision. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 533–546.
- [21] David C. Burr, Maria Concetta Morrone, and John Ross. 1994. Selective suppression of the magnocellular visual pathway during saccadic eye movements. *Nature* 371, 6497 (1994), 511–513. doi:10.1038/371511a0
- [22] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587 [cs.CV]* <https://arxiv.org/abs/1706.05587>
- [23] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. 2019. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 12472–12482. <https://api.semanticscholar.org/CorpusID:208248153>
- [24] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. 2021. Per-Pixel Classification is Not All You Need for Semantic Segmentation. In *Neural Information Processing Systems*. <https://api.semanticscholar.org/CorpusID:235829267>
- [25] Andrea Chirico, Fabio Lucidi, Michele De Laurentiis, Carla Milanese, Alessandro Napoli, and Antonio Giordano. 2016. Virtual reality in health system: beyond entertainment. A mini-review on the efficacy of VR during cancer treatment. *Journal of cellular physiology* 231, 2 (2016), 275–287.
- [26] Microsoft Corporation. 2019. Microsoft HoloLens 2 Hardware and Inside-Out Tracking System. <https://learn.microsoft.com/en-us/hololens/hololens2-hardware>. Optical inside-out head tracking with 4 visible-light cameras + IMU for real-time 6DoF pose; accessed 2025-11-15.
- [27] Kathleen E Cullen, Marko Huterer, Danielle A Braidwood, and Pierre A Sylvestre. 2004. Time course of vestibuloocular reflex suppression during gaze shifts. *Journal of neurophysiology* 92, 6 (2004), 3408–3422.
- [28] Pierre M Daye, Dale C Roberts, David S Zee, and Lance M Optican. 2015. Vestibulo-ocular reflex suppression during head-fixed saccades reveals gaze feedback control. *Journal of Neuroscience* 35, 3 (2015), 1192–1198.
- [29] Matthew R. Diamond, John Ross, and M. Concetta Morrone. 2000. Extraretinal control of saccadic suppression. *Journal of Neuroscience* 20, 9 (2000), 3449–3455. doi:10.1523/JNEUROSCI.20-09-03449.2000
- [30] Mingyu Ding, Zhe Wang, Bolei Zhou, Jianping Shi, Zhiwu Lu, and Ping Luo. 2020. Every frame counts: Joint learning of video segmentation and optical flow. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 10713–10720.
- [31] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Douragarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, et al. 2020. DepthLab: Real-time 3D interaction with depth maps for mobile augmented reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 829–843.
- [32] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brigid Meredith, et al. 2023. Project aria: A new tool for egocentric multi-modal ai research. *arXiv preprint arXiv:2308.13561* (2023).
- [33] Gunnar Farnéback. 2003. Two-Frame Motion Estimation Based on Polynomial Expansion. In *Scandinavian Conference on Image Analysis*. <https://api.semanticscholar.org/CorpusID:15601477>
- [34] Yu Feng, Zihan Liu, Jingwen Leng, Minyi Guo, and Yuhao Zhu. 2024. Cicero: Addressing algorithmic and architectural bottlenecks in neural rendering by radiance warping and memory optimizations. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 1293–1308.
- [35] Yu Feng, Tianrui Ma, Yuhao Zhu, and Xuan Zhang. 2024. BlissCam: Boosting Eye Tracking Efficiency with Learned In-Sensor Sparse Sampling. *arXiv preprint arXiv:2404.15733* (2024).
- [36] Yu Feng, Paul Whatmough, and Yuhao Zhu. 2019. Asv: Accelerated stereo vision system. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 643–656.
- [37] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. 2015. FlowNet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852* (2015).
- [38] Edward G Freedman and David L Sparks. 1997. Eye-head coordination during head-unrestrained gaze shifts in rhesus monkeys. *Journal of neurophysiology* 77, 5 (1997), 2328–2348.
- [39] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. 2017. Semantic video cnns through representation warping. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 4463–4472.
- [40] Ashkan Ganj, Yiqin Zhao, Hang Su, and Tian Guo. 2024. Mobile ar depth estimation: Challenges & prospects. In *Proceedings of the 25th International Workshop on Mobile Computing Systems and Applications*. 21–26.
- [41] Antonin Gilles, Pierre Le Gargasson, Grégory Hocquet, and Patrick Gioia. 2023. Holographic near-eye display with real-time embedded rendering. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.
- [42] Jorge Gomez, Saavan Patel, Syed Shakib Sarwar, Ziyun Li, Raffaele Capoccia, Zhao Wang, Reid Pinkham, Andrew Berkovich, Tsung-Hsun Tsai, Barbara De Salvo, et al. 2022. Distributed on-sensor compute system for ar/vr devices: A semi-analytical simulation framework for power estimation. *arXiv preprint arXiv:2203.07474* (2022).
- [43] Santiago González Izard, Juan A Juanes Méndez, Pablo Ruisoto Palomera, and Francisco J García-Peñalvo. 2019. Applications of virtual and augmented reality in biomedical imaging. *Journal of medical systems* 43, 4 (2019), 102.
- [44] Santiago González Izard, Ramiro Sánchez Torres, Oscar Alonso Plaza, Juan Antonio Juanes Méndez, and Francisco José García-Peñalvo. 2020. Nextmed: automatic imaging segmentation, 3D reconstruction, and 3D model visualization platform using augmented and virtual reality. *Sensors* 20, 10 (2020), 2962.
- [45] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. 2024. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19383–19400.
- [46] Daniel Guitton. 1992. Control of eye-head coordination during orienting gaze shifts. *Trends in neurosciences* 15, 5 (1992), 174–179.
- [47] Daniel Guitton and Michel Volle. 1987. Gaze control in humans: eye-head coordination during orienting movements to targets within and beyond the oculomotor range. *Journal of neurophysiology* 58, 3 (1987), 427–459.
- [48] Mingcong Han, Hanze Zhang, Rong Chen, and Haibo Chen. 2022. Microsecond-scale Preemption for Concurrent GPU-accelerated DNN Inferences. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 539–558. <https://www.usenix.org/conference/osdi22/presentation/han>

- [49] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. <https://api.semanticscholar.org/CorpusID:54465873>
- [50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [51] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. 2024. OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning. *arXiv preprint arXiv:2406.08858* (2024).
- [52] Meta Platform Inc. 2022. Meta Quest Pro. <https://www.meta.com/quest/quest-pro/>.
- [53] Wei Jin, Guanghui He, Weifeng He, and Zhigang Mao. 2017. A 12-bit 4928×3264 pixel CMOS image signal processor for digital still cameras. *Integration* 59 (2017), 206–217.
- [54] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4015–4026.
- [55] George A. Koulouris, Kaan Akşit, Michael Stengel, Rafal K. Mantiuk, Katerina Mania, and Christian Richardt. 2019. Near-Eye Display and Tracking Technologies for Virtual and Augmented Reality. *Computer Graphics Forum* 38, 2 (2019), 493–519. doi:10.1111/cgf.13654
- [56] Yuna Kwak, Eric Penner, Xuan Wang, Mohammad R Saeedpour-Parizi, Olivier Mercier, Xiuyun Wu, Scott Murdison, and Phillip Guan. 2024. Saccade-contingent rendering. In *ACM SIGGRAPH 2024 Conference Papers*. 1–9.
- [57] Joseph J LaViola Jr. 2000. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin* 32, 1 (2000), 47–56.
- [58] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 13094–13102.
- [59] Xiangtai Li, Henghui Ding, Wenwei Zhang, Haobo Yuan, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. 2023. Transformer-Based Visual Segmentation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46 (2023), 10138–10163. <https://api.semanticscholar.org/CorpusID:258212528>
- [60] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. 2020. Fully Convolutional Networks for Panoptic Segmentation. 2021 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), 214–223. <https://api.semanticscholar.org/CorpusID:227239035>
- [61] Hao Lin, Darragh Mullins, Dara Molloy, Enda Ward, Fiachra Collins, Patrick Denry, Martin Glavin, Brian Deegan, and Edward Jones. 2024. Optimizing Camera Exposure Time for Automotive Applications. *Sensors* 24, 16 (2024), 5135.
- [62] Huaijia Lin, Ruizheng Wu, Shu Liu, Jiangbo Lu, and Jiaya Jia. 2021. Video instance segmentation with a propose-reduce paradigm. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1739–1748.
- [63] Chiao Liu, Song Chen, Tsung-Hsun Tsai, Barbara De Salvo, and Jorge Gomez. 2022. Augmented reality-the next frontier of image sensors and compute systems. In 2022 *IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 426–428.
- [64] Lufei Liu and Tor M. Aamodt. 2025. ReFrame: Layer Caching for Accelerated Inference in Real-Time Rendering. In *Proceedings of the 42nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR, 38049–38065. <https://proceedings.mlr.press/v267/liu25a.html>
- [65] Xinglong Luo, Kunming Luo, Ao Luo, Zhengning Wang, Ping Tan, and Shuaicheng Liu. 2023. Learning optical flow from event camera with rendered dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9847–9857.
- [66] Zhaoyang Lv, Nicholas Charron, Pierre Moulon, Alexander Gamino, Cheng Peng, Chris Sweeney, Edward Miller, Huixuan Tang, Jeff Meissner, Jing Dong, Kiran Somasundaram, Luis Pesqueira, Mark Schwesinger, Omkar Parkhi, Qiao Gu, Renzo De Nardi, Shangyi Cheng, Steve Saarinen, Vijay Baiyya, Yuyang Zou, Richard Newcombe, Jakob Julian Engel, Xiqing Pan, and Carl Ren. 2024. Aria Everyday Activities Dataset. *arXiv:2402.13349* [cs.CV] <https://arxiv.org/abs/2402.13349>
- [67] Tianrui Ma, Yu Feng, Xuan Zhang, and Yuhao Zhu. 2023. Camj: Enabling system-level energy modeling and architectural exploration for in-sensor visual computing. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–14.
- [68] Magic Leap. 2024. Magic Leap 2. <https://www.magicleap.com/magic-leap-2/>
- [69] Malith Maheepala, Matthew A Joordens, and Abbas Z Kouzani. 2020. Low power processors and image sensors for vision-based IoT devices: A review. *IEEE Sensors Journal* 21, 2 (2020), 1172–1186.
- [70] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lohkov, Nouamane Tazi, et al. 2025. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299* (2025).
- [71] Mitsuhiro Mase, Shoji Kawahito, Masaaki Sasaki, Yasuo Wakamori, and Masanori Furuta. 2005. A wide dynamic range CMOS image sensor with multiple exposure-time signal outputs and 12-bit column-parallel cyclic A/D converters. *IEEE journal of solid-state circuits* 40, 12 (2005), 2787–2795.
- [72] Ethel Martin. 1974. Saccadic suppression: a review and an analysis. *Psychological bulletin* 81, 12 (1974), 899.
- [73] Shervin Minaee, Yuri Boykov, Fatih Murat Porikli, Antonio J. Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. 2020. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2020), 3523–3542. <https://api.semanticscholar.org/CorpusID:210702798>
- [74] Andrew YC Nee, Soh Khim Ong, George Chrysosolouris, and Dimitris Mourtzis. 2012. Augmented reality applications in design and manufacturing. *CIRP annals* 61, 2 (2012), 657–679.
- [75] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. 2019. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. 2019 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 8829–8837. <https://api.semanticscholar.org/CorpusID:195658142>
- [76] NVIDIA Corporation. 2024. *CUDA C++ Programming Guide*. <https://docs.nvidia.com/cuda/cuda-c-programming-guide/>
- [77] NVIDIA Corporation. 2025. *Tegra System Profiler*. NVIDIA GameWorks Documentation. Available documentation: 3.7 / 3.6 / 3.1 / 2.8 / 2.7 / 2.5 User Guides. Retrieved 2025-08-09 from https://docs.nvidia.com/gameworks/content/developertools/mobile/tegra_system_profiler/tegraprofiler_main.htm
- [78] Marcus Nyström and Kenneth Holmqvist. 2010. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior research methods* 42, 1 (2010), 188–204.
- [79] EAB Over, ITC Hooge, BNS Vlaskamp, and CJ17617434 Erkelens. 2007. Coarse-to-fine eye movement strategy in visual search. *Vision Research* 47, 17 (2007), 2272–2280.
- [80] Junrui Pan and Timothy G Rogers. 2024. CRISP: Concurrent Rendering and Compute Simulation Platform for GPUs. In 2024 *IEEE International Symposium on Workload Characterization (ISWC)*. IEEE, 1–14.
- [81] Xiqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. 2023. Aria Digital Twin: A New Benchmark Dataset for Egocentric 3D Machine Perception. *arXiv:2306.06362* [cs.CV] <https://arxiv.org/abs/2306.06362>
- [82] Tianfan Peng, Tianhua Xia, Jiajun Qin, and Sai Qian Zhang. 2025. HAAN: A Holistic Approach for Accelerating Normalization Operations in Large Language Models. In 2025 *Design, Automation & Test in Europe Conference (DATE)*. IEEE, 1–7.
- [83] Toby Perrett, Ahmad Darkhalil, Saptarshi Sinha, Omar Emara, Sam Pollard, Kranti Kumar Parida, Kaiting Liu, Prajwal Gatti, Siddhant Bansal, Kevin Flanagan, Jacob Chalk, Zhifan Zhu, Rhodri Guerrier, Fahd Abdelazim, Bin Zhu, Davide Moltisanti, Michael Wray, Hazel Doughty, and Dima Damen. 2025. HD-EPIC: A Highly-Detailed Egocentric Video Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 23901–23913.
- [84] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. 2017. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675* (2017).
- [85] Matt Poremba, Sparsh Mittal, Dong Li, Jeffrey S Vetter, and Yuan Xie. 2015. Destiny: A tool for modeling emerging 3d nvm and edram caches. In 2015 *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1543–1546.
- [86] Jiajun Qin, Tianhua Xia, Cheng Tan, Jeff Zhang, and Sai Qian Zhang. 2025. PICACHU: Plug-In CGRA Handling Upcoming Nonlinear Operations in LLMs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 845–861.
- [87] Qualcomm Technologies, Inc. 2024. Qualcomm AI Hub. <https://app.aihub.qualcomm.com/docs/index.html>. Accessed: 2025-04-19.
- [88] Qualcomm Technologies, Inc. 2024. Snapdragon XR2 Gen 2 Platform. <https://www.qualcomm.com/products/mobile/snapdragon/xr-vr-ar/snapdragon-xr2-gen-2-platform>. Accessed: 2025-04-19.
- [89] Arnab Raha, Souvik Kundu, Sharath Nittur Sridhar, Shamik Kundu, Soumendu Kumar Ghosh, Alessandro Palla, Arghadip Das, Darren Crews, and Deepak A Mathaikutty. 2025. LLM-NPU: Towards Efficient Foundation Model Inference on Low-Power Neural Processing Units. In 2025 *IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 1–8.
- [90] Frano Rajčić, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. 2023. Segment anything meets point tracking. *arXiv preprint arXiv:2307.01197* (2023).
- [91] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

- [92] Yeongil Ryu and Eun-Seok Ryu. 2021. Overview of motion-to-photon latency reduction for mitigating VR sickness. *KSII Transactions on Internet and Information Systems (TIIS)* 15, 7 (2021), 2531–2546.
- [93] Mohammadreza Saed, Yuan Hsi Chou, Lufei Liu, Tyler Nowicki, and Tor M. Aamodt. 2022. Vulkan-Sim: A GPU Architecture Simulator for Ray Tracing. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 263–281. doi:10.1109/MICRO56248.2022.00027
- [94] Chandan K Sahu, Crystal Young, and Rahul Rai. 2021. Artificial intelligence (AI) in augmented reality (AR)-assisted manufacturing applications: a review. *International journal of production research* 59, 16 (2021), 4903–4959.
- [95] Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications (ETRA)*. ACM, 71–78. doi:10.1145/355017.355028
- [96] Satyabrata Sarangi and Bevan Baas. 2021. DeepScaleTool: A tool for the accurate estimation of technology scaling in the deep-submicron era. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
- [97] ON Semiconductor. [n.d.]. AR1335 – 1/3.2-Inch 13 MP CMOS Image Sensor. <https://www.onsemi.com/products/sensors/image-sensors/ar1335>.
- [98] ON Semiconductor. 2025. AP0101CS High-Dynamic Range (HDR) Image Signal Processor (ISP). <http://www.onsemi.com/pub/Collateral/AP0101CS-D.PDF>.
- [99] ON Semiconductor. 2025. MT9P031 – 1/4-Inch 2.0-Mega-Pixel CMOS Image Sensor. <https://www.onsemi.com/products/sensors/image-sensors/mt9p031>. Accessed: 2025-11-14.
- [100] Min-Woong Seo, Myunglae Chu, Hyun-Yong Jung, Suksan Kim, Jiyou Song, Daehye Bae, Sanggwon Lee, Junan Lee, Sung-Yong Kim, Jongyeon Lee, et al. 2022. 2.45 e-RMS low-random-noise, 598.5 mW low-power, and 1.2 kfps high-speed 2-Mp global shutter CMOS image sensor with pixel-level ADC and memory. *IEEE Journal of Solid-State Circuits* 57, 4 (2022), 1125–1137.
- [101] Haikuo Shao and Zhongfeng Wang. 2025. ASTRA: Reconfigurable Training Architecture Design for Nonlinear Softmax and Activation Functions in Transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2025).
- [102] Zhuoran Song, Feiyang Wu, Xueyuan Liu, Jing Ke, Naifeng Jing, and Xiaoyao Liang. 2020. VR-DANN: Real-time video recognition via decoder-assisted neural network acceleration. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 698–710.
- [103] Qi Sun, Anjul Patney, Li-Yi Wei, Omer Shapira, Jingwan Lu, Paul Asente, Suwen Zhu, Morgan McGuire, David Luebke, and Arie Kaufman. 2018. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.
- [104] Xiaoyu Sun, Xiaochen Peng, Sai Zhang, Jorge Gomez, Win-San Khwa, Syed Sarwar, Ziyun Li, Weidong Cao, Zhao Wang, Chiao Liu, et al. 2024. Estimating Power, Performance, and Area for On-Sensor Deployment of AR/VR Workloads Using an Analytical Framework. *ACM Transactions on Design Automation of Electronic Systems* (2024).
- [105] Santosh Kumar Swain and Debasmita Dubey. 2023. Vestibulo-ocular Reflex – A Narrative Review. *Matrix Science Medica* 7, 4 (2023), 90–94. doi:10.4103/MTSM.MTSM_24_22
- [106] Synopsys, Inc. 2026. *Design Compiler: Concurrent Timing, Area, Power, and Test Optimization*. <https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler.html>
- [107] Tomu Tahara, Takashi Seno, Gaku Narita, and Tomoya Ishikawa. 2020. Retargetable AR: Context-aware augmented reality in indoor scenes based on 3D scene graphs. In *2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, 249–255.
- [108] Tianxiang Tan and Guohong Cao. 2024. Thermal-aware scheduling for deep learning on mobile devices with NPU. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 10706–10719.
- [109] Leonardo Tanzi, Pietro Piazzolla, Francesco Porpiglia, and Enrico Vezzetti. 2021. Real-time deep learning semantic segmentation during intra-operative surgery for 3D augmented reality assistance. *International Journal of Computer Assisted Radiology and Surgery* 16, 9 (2021), 1435–1445.
- [110] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2020. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*. <https://api.semanticscholar.org/CorpusID:229363322>
- [111] Tsung-Hsun Tsai, Kwang-Han Chang, Andrew Berkovich, Raffaele Capocchia, Song Chen, Zhao Wang, Chiao Liu, Yi-Hsuan Lin, ShengYeh Lai, Hao-Ming Hsu, et al. 2025. A 400×400 3.24 μm 117dB-Dynamic-Range 3-Layer Stacked Digital Pixel Sensor. *ITE Technical Report; ITE Tech. Rep.* 49, 13 (2025), 53–57.
- [112] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J Black. 2016. Video segmentation via object flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3899–3908.
- [113] Julien Valentin, Adarsh Kowdle, Jonathan T Barron, Neal Wadhwa, Max Dzitsiuk, Michael Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, et al. 2018. Depth from motion for smartphone AR. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 1–19.
- [114] Haiyu Wang, Wenxuan Liu, Kenneth Chen, Qi Sun, and Sai Qian Zhang. 2025. Process Only Where You Look: Hardware and Algorithm Co-optimization for Efficient Gaze-Tracked Foveated Rendering in Virtual Reality. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture*. 344–358.
- [115] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Loddon Yuille, and Liang-Chieh Chen. 2020. Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation. In *European Conference on Computer Vision*. <https://api.semanticscholar.org/CorpusID:212737096>
- [116] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. 2020. Deep High-Resolution Representation Learning for Visual Recognition. arXiv:1908.07919 [cs.CV] <https://arxiv.org/abs/1908.07919>
- [117] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. 2021. End-to-end video instance segmentation with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8741–8750.
- [118] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13 (2004), 600–612. <https://api.semanticscholar.org/CorpusID:207761262>
- [119] Thomas Westin, José Neves, Peter Mozelius, Carla Sousa, and Lara Mantovan. 2022. Inclusive AR-games for education of deaf children: Challenges and opportunities. In *European Conference on Games Based Learning*, Vol. 16. 597–604.
- [120] Diana Wolk, Fan Ma, Towaki Kravitz, Amir Yang, Stefano Soatto, and Stephan Mandt. 2019. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*. 6101–6108.
- [121] Junfeng Wu, Yi Jiang, Song Bai, Wenqing Zhang, and Xiang Bai. 2022. Seqformer: Sequential transformer for video instance segmentation. In *European Conference on Computer Vision*. Springer, 553–569.
- [122] Jonas Wulff and Michael J Black. 2015. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 120–130.
- [123] Tianhua Xia, Haiyu Wang, and Sai Qian Zhang. 2026. Segment Only Where You Look: Leveraging Human Gaze Behavior for Efficient Computer Vision Applications in Augmented Reality. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 1693–1710.
- [124] Tianhua Xia and Sai Qian Zhang. 2024. Hyft: A Reconfigurable Softmax Accelerator with Hybrid Numeric Format for both Training and Inference. In *Proceedings of the 29th ACM/IEEE International Symposium on Low Power Electronics and Design (Newport Beach, CA, USA) (ISLPED '24)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/3665314.3670816
- [125] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems* 34 (2021), 12077–12090.
- [126] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. 2019. UPSNet: A Unified Panoptic Segmentation Network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 8810–8818. <https://api.semanticscholar.org/CorpusID:58004609>
- [127] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. 2024. EfficientSAM: Leveraged masked image pretraining for efficient segment anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16111–16121.
- [128] Bin Yan, Yi Jiang, Jiannan Wu, Dong Wang, Ping Luo, Zehuan Yuan, and Huchuan Lu. 2023. Universal instance perception as object discovery and retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15325–15336.
- [129] Linjie Yang, Yuchen Fan, and Ning Xu. 2019. Video instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5188–5197.
- [130] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth anything v2. *Advances in Neural Information Processing Systems* 37 (2024), 21875–21911.
- [131] Lita Yang, Changjung Kao, Sriseshan Srikanth, H Ekin Sumbul, Tony F Wu, Huichu Liu, and Edith Beigne. 2024. Characterization and Design of 3D-Stacked Memory for Image Signal Processing on AR/VR Devices. In *Proceedings of the International Symposium on Memory Systems*. 38–44.
- [132] Linwei Yang, Fan Wang, Jian Huang, Hao Li, Zhen Xu, and Yanwei Fu. 2023. RT-MonoDepth: Real-Time Memory Efficient Monocular Depth Estimation on Embedded Systems. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv:2308.10569.
- [133] Yaffa Yeshurun, Marisa Carrasco, and Laurence T Maloney. 2008. Bias and sensitivity in two-interval forced choice procedures: Tests of the difference model. *Vision research* 48, 17 (2008), 1837–1851.

- [134] Ziyu Ying, Shulin Zhao, Haibo Zhang, Cyan Subhra Mishra, Sandeepa Bhuyan, Mahmut T Kandemir, Anand Sivasubramaniam, and Chita R Das. 2022. Exploiting frame similarity for efficient inference on edge devices. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1073–1084.
- [135] Hongyi Zeng, Wenxuan Liu, Tianhua Xia, Jinhui Chen, Ziyun Li, and Sai Qian Zhang. 2025. Foveated Instance Segmentation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 24496–24505.
- [136] Baoheng Zhang, Yizhao Gao, Jingyuan Li, and Hayden Kwok-Hay So. 2024. Co-designing a Sub-millisecond Latency Event-based Eye Tracking System with Submanifold Sparse CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5771–5779.
- [137] Milin Zhang and Amine Bermak. 2010. CMOS Image Sensor with On-Chip Image Compression: A Review and Performance Analysis. *J. Sensors* 2010 (2010), 920693:1–920693:17. <https://api.semanticscholar.org/CorpusID:1124694>
- [138] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.
- [139] Zhiyong Zhang, Huaizu Jiang, and Hanumant Singh. 2024. NeufLOW: Real-time, high-accuracy optical flow estimation on robots using edge devices. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 5048–5055.
- [140] Ziliang Zhang, Zexin Li, Hyoseung Kim, and Cong Liu. 2024. BOXR: Body and head motion Optimization framework for eXtended Reality. *arXiv preprint arXiv:2410.13084* (2024).
- [141] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2881–2890.
- [142] Shulin Zhao, Haibo Zhang, Sandeepa Bhuyan, Cyan Subhra Mishra, Ziyu Ying, Mahmut T Kandemir, Anand Sivasubramaniam, and Chita R Das. 2020. Déjà view: Spatio-temporal compute reuse for ‘energy-efficient 360 vr video streaming. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 241–253.
- [143] Shulin Zhao, Haibo Zhang, Cyan Subhra Mishra, Sandeepa Bhuyan, Ziyu Ying, Mahmut Taylan Kandemir, Anand Sivasubramaniam, and Chita Das. 2021. Holoar: On-the-fly optimization of 3d holographic processing for augmented reality. In *MICRO-54: 54th annual IEEE/ACM international symposium on microarchitecture*. 494–506.
- [144] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Deep feature flow for video recognition. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 4141–4150.
- [145] Yucheng Zhu, Xionghuo Min, Dandan Zhu, Guangtao Zhai, Xiaokang Yang, Wenjun Zhang, Ke Gu, and Jiantao Zhou. 2023. Toward visual behavior and attention understanding for augmented 360 degree videos. *ACM Transactions on Multimedia Computing, Communications and Applications* 19, 2s (2023), 1–24.
- [146] Yuhao Zhu, Anand Samajdar, Matthew Mattina, and Paul Whatmough. 2018. Euphrates: Algorithm-soc co-design for low-power mobile continuous vision. *arXiv preprint arXiv:1803.11232* (2018).

A Artifact Appendix

A.1 Abstract

The artifact packages the EDIVA algorithm of Section 3: the Depth Estimation Module (Section 3.2), the geometry-based Mask Reprojection (Section 3.1), and the Direct Carryover check and Reprojection check that decide among them (Section 3.6). It ships with EfficientSAM-Small (ESAM-S) [127] as the segmentation model and one Aria Digital Twin (ADT) [81] sequence carrying the gaze, pose, and RGB inputs the algorithm consumes. Running it reproduces the algorithmic results in Table 2: EDIVA serves most frames from the EDIVA cache instead of invoking the segmentation model, at negligible IoU loss with respect to the Full Video (FV) baseline. We give the check-list, then the installation, the experiment workflow, and the expected results.

A.2 Artifact check-list (meta-information)

- **Algorithm:** EDIVA algorithm (Figure 6 (c)).
- **Model:** ESAM-S [127] as the segmentation model, plus the Depth Estimation and Mask Refinement Modules; checkpoints are downloaded by the setup script.
- **Data set:** One ADT sequence, downloaded by the setup script.
- **Hardware:** Any CUDA GPU.
- **Metrics:** IoU, GFLOPs per frame.
- **How much disk space required (approximately)?:** 1 GB.
- **How much time is needed to prepare workflow (approximately)?:** 30 minutes.
- **How much time is needed to complete experiments (approximately)?:** 5 minutes.
- **Publicly available?:** Yes.
- **Code licenses (if publicly available)?:** The source code is released under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.
- **Data licenses (if publicly available)?:** The artifact uses datasets from the sources listed in the paper. The licenses and usage terms of the original datasets apply.
- **Archived (provide DOI)?:** The artifact is archived on Zenodo and is available at <https://doi.org/10.5281/zenodo.23225199>

A.3 Description

A.3.1 How to access. Code is available at <https://doi.org/10.5281/zenodo.23225199>. The sequence slice and the checkpoints can be fetched by `download_assets.py`.

A.3.2 Hardware dependencies. Any machine with a CUDA GPU.

A.3.3 Software dependencies. Python 3.12 or newer with a CUDA build of PyTorch and a small set of standard scientific packages, installed from the provided requirements file.

A.3.4 Data sets. A contiguous 150-frame window of an ADT sequence, providing per frame the RGB image, the gaze location g_t , the ground-truth instance masks, and the device pose U_t , together with the camera calibration. Poses and calibration ship as small files, so the multi-gigabyte raw ADT recording is not required.

A.4 Installation

- Clone the repository.
- Create an environment, install PyTorch matching the local CUDA runtime, then
`pip install -r requirements.txt`.
- Run
`python download_assets.py` to fetch the sequence slice and the checkpoints.

A.5 Suggested experiment workflow

`bash run_repro.sh` runs the whole flow in three steps. First, the FV baseline of Section 5, which executes the segmentation model on every frame. Second, EDIVA, which follows Figure 6 (c): it attempts the Direct Carryover check, then the Reprojection check, and invokes the segmentation model only when both fail. Third, `verify.py`, which compares the two passes against recorded reference values and prints a pass/fail table. `python measure_flops.py` then reports the per-frame compute figures.

A.6 Evaluation and expected results

EDIVA preserves the segmentation quality of per-frame execution while invoking the segmentation model on a fraction of frames. On the shipped window, FV reaches an IoU of 0.461 and EDIVA reaches 0.457 while serving 62% of frames from direct carryover or reprojection, which lowers the per-frame cost from 208.2 to 79.1 GFLOPs. Absolute values differ from Table 2, which averages over ten test sequences per dataset, but the trend is the one reported there.

Inference runs without sampling, so these values are deterministic: they reproduce exactly on any GPU, and `verify.py` asserts them to 10^{-4} . The console output additionally reports the measured latency of each decision path on the host machine.