BlissCam: Boosting Eye Tracking Efficiency with Learned In-Sensor Sparse Sampling

Yu Feng*♣
Shanghai Jiao Tong University
University of Rochester
y-feng@sjtu.edu.cn

Tianrui Ma*
Washington University in St. Louis
tianrui.ma@wustl.edu

Yuhao Zhu#
University of Rochester
yzhu@rochester.edu

Xuan Zhang#
Northeastern University
xuan.zhang@northeastern.edu

***Abstract*—Eye tracking is becoming an increasingly important task domain in emerging computing platforms such as Augmented/Virtual Reality (AR/VR). Today's eye tracking system suffers from long end-to-end tracking latency and can easily eat up half of the power budget of a mobile VR device. Most existing optimization efforts exclusively focus on the computation pipeline by optimizing the algorithm and/or designing dedicated accelerators while largely ignoring the front-end of any eye tracking pipeline: the image sensor. This paper makes a case for co-designing the imaging system with the computing system.**

In particular, we propose the notion of "in-sensor sparse sampling", whereby the pixels are drastically downsampled (by 20×) within the sensor. Such in-sensor sampling enhances the overall tracking efficiency by significantly reducing 1) the power consumption of the sensor readout chain and sensor-host communication interfaces, two major power contributors, and 2) the work done on the host, which receives and operates on far fewer pixels. With careful reuse of existing pixel circuitry, our proposed BlissCam requires little hardware augmentation to support the in-sensor operations. Our synthesis results show up to 8.2 × energy reduction and 1.4 × latency reduction over existing eye tracking pipelines.

***Index Terms*—In-Sensor Computing; Eye Tracking; Sparse Sensing; AR/VR**

I. Introduction

Eye tracking provides a fundamental utility in many fields, ranging from medical studies [26], [28] and human-machine interaction [32], [69], [89], [91], [116] to augmented/virtual reality (AR/VR) and spatial computing [41], [63], [68], [100], [103], [124], [134]. Accurately and efficiently tracking eye gazes play an important role in understanding human cognition [123], enabling gaze-based human-machine interactions [113], and improving communication and computation efficiency of AR/VR systems [31], [44], [51], [61], [101].

Despite its essentiality, eye tracking is known to be slow and power-hungry [92]. In our measurement of commercial eye trackers (e.g., HTC Vive Pro Eyes and Tobii), its latency is usually in excess of 15 ms, enough to introduce visual artifacts (e.g., in gaze-contingent rendering), and its always-on status constantly consumes over 2 W [18], eating up half of the power budget of a typical VR system [2], [4], [9].

Most of today's efforts in optimizing eye tracking focus on the algorithm pipeline, either by optimizing the tracking algorithms [21], [34], [49], [71], [75] or by designing dedicated hardware accelerators [131], [135], while largely ignoring the indispensable front-end of any eye tracking pipeline: the image sensor, which generates near-eye images for the tracking algorithms to consume.

Rationale. This paper makes a case for jointly designing the imaging system and the tracking algorithm to significantly reduce energy consumption while satisfying the stringent tracking latency requirement. In particular, we propose the notion of "in-sensor sparse sampling", whereby the pixels are drastically down-sampled (retaining only about 5% of the pixels) *within the sensor*. The downstream eye tracking algorithm is carefully co-designed to be robust and take advantage of the sparse inputs.

Such sensor/algorithm co-design offers two unique opportunities. First, we can optimize a previously untapped system component with significant power and latency implications, namely the image sensor. Modern image sensors, along with their communication interfaces, are power hungry; they consume power upwards of a few Watts, making up half of the eye tracking power. The sensor power is dominated by the analog readout chain and the sensor-host data transfer, both of which are decreased with in-sensor data reduction. Second, with sparsely sampled sensor data, the host eye tracking algorithm receives and, thus, operates on far few pixels, further reducing the tracking latency and energy consumption.

Contributions. We make both algorithmic and architectural contributions. Algorithm-wise, we show how to design the image sampling algorithm to reduce tracking latency and energy without hurting the accuracy. We observe that the background in near-eye images is stationary and the only moving pixels in an image contribute to the gaze result. This observation leads to a two-stage sampling algorithm, which first detects the moving parts of an image as the Region-of-

♣Work done while at University of Rochester
*Equal contribution
#Corresponding authors

Interest (ROI) followed by random sampling within the ROI.

We propose an eye tracking algorithm to take advantage of the pixel reduction. The algorithm is based on Vision Transformers whose accuracy is robust against pixel sparsity and whose cost of computation naturally reduces as the pixel volume reduces. Critically, both the data sampling algorithm and the eye tracking algorithm are (approximately) differentiable, which allows us to jointly train the in-sensor and off-sensor operations to maximize end-to-end tracking accuracy.

The architectural contribution of the paper is to minimally augment the sensor architecture to support various in-sensor operations. We base our design on the increasingly popular die-stacked Digital Pixel Sensor (DPS) architecture, where the top layer is the usual pixel array and the bottom layer integrates per-pixel ADC/SRAM and a DNN accelerator [23], [47], [62], [111]. We show how to map the in-sensor sampling algorithm to the bottom layer by time-multiplexing existing circuit components between the sparse sampling mode and the usual imaging mode.

Results. According to results obtained through digital logic synthesis and analog circuit-level simulation, we show that our eye tracking system reduces pixel volume by about 95%, leading to an 8.2× energy reduction and a 1.4× tracking latency reduction compared to existing eye tracking systems, all with little degradation on the tracking accuracy. Our end-to-end trained in-sensor sampling strategy and eye tracking algorithm consistently outperform baselines and other variants in accuracy across a range of sampling rates, showing the benefits of joint design of in-sensor and off-sensor algorithms.

In summary, we present a new form of algorithm-hardware co-design, where the hardware spans both the conventional accelerators and, critically, the image sensor. We expand the research scope from optimizing only for (DNN) accelerators to the end-to-end eye tracking pipeline, which necessarily includes the image sensor. The key in our work is to optimize the sensor architecture *jointly* with the off-sensor computation — through sparse in-sensor sampling. We hope that the paper can inspire follow-up work on joint sensing-computing system optimization. Our specific contributions are:

- We analyze the technology trend and pinpoint the system-level bottleneck of today's eye tracking pipeline.
- We propose an in-sensor sparse sampling algorithm jointly designed with an off-sensor eye tracking algorithm. They collectively reduce the pixel volume while preserving high gaze prediction accuracy.
- We propose hardware augmentations, both analog (Fig. 10) and digital (Fig. 11), to support in-sensor sparse sampling for the first time. The hardware extensions are *intentionally* kept minimum, enabled by intelligently reusing existing hardware components.
- We demonstrate a systematic integration of eventification, ROI prediction, sampling, and readout, which serves as a reference design for future stacked image sensors that are increasingly integrating computation capabilities.
- We propose a new timing design that schedules the hardware components, within and off the sensor, to ensure that the FPS is unaffected by the addition of new computations and hardware components (Fig. 8).
- Together, we achieve an 8.2× energy saving and 1.4× latency reduction compared to existing eye tracking systems with negligible accuracy compromise.

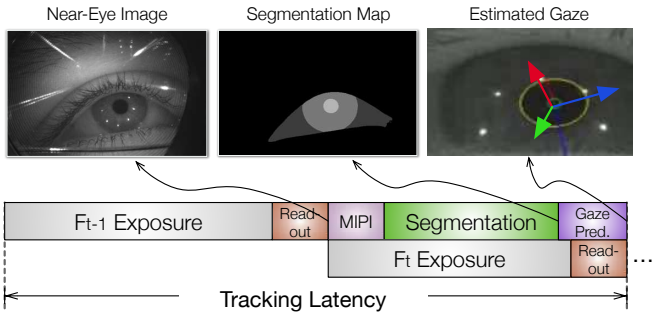


Fig. 1: A typical eye tracking pipeline, which starts from image sensing (exposure and readout) to obtain an near-eye image, which is transferred to the host processor through the MIPI CSI-2 interface. The host processor first segments important eye parts, from which the gaze is estimated. Different frames are overlapped to improve tracking frequency. Figure not drawn to scale; readout delay is usually three orders of magnitude shorter than the exposure time.

II. Background and Motivation

We first review today's mainstream eye-tracking pipeline (Sec. II-A) and the basics of image sensors (Sec. II-B). We then discuss the scaling trends of eye tracking technologies in AR/VR, motivating the paper (Sec. II-C).

A. Eye Tracking Basics

The goal of eye tracking is to estimate the user's real-time gaze—a 3D vector indicating where the eye is looking. It provides a core utility for a variety of human-machine interfaces. In particular, eye tracking is essential to next-generation AR/VR systems, where the rendering is contingent on gaze information [72], [105] and the user interface (UI) is controlled by gaze [32], [69], [89], [91], [116]. Apart from AR/VR, eye tracking is also widely used in vision science [26], [67], [107], cognitive study [104], [110], and education [70].

Eye Tracking Pipeline. A typical eye tracking pipeline is illustrated in Fig. 1. An image captured by a near-eye camera goes through two stages: eye segmentation, which dissects the foreground eye parts (e.g., pupil, iris, cornea), and gaze prediction, which predicts the gaze from the segmentation map [35], [57], [79], [129], [133], [136]. In today's state-of-the-art eye tracking system, the eye segmentation stage is usually performed through Deep Neural Networks (DNNs), whereas the gaze prediction stage employs regression models based on the geometric model of human eyes, making eye segmentation considerably more time-consuming than gaze estimation.

System Specifications. It is shown that the tracking frequency needs to be around 120 Hz with a tracking latency of sub-10 ms and an accuracy of 0.5-1.0° [5], [6], [7], [12].

120 Hz is necessary because humans frequently make rapid eye movements (i.e., saccades) whose speed can be up to 700°/s [22], the tracking rate must be high to track such rapid movements. Meanwhile, we must work with a tight power envelope available on mobile AR/VR devices (around 3-6 W [2], [4], [9]) to avoid user discomfort induced by the thermal effect. The image sensor, host processor, and sensor-host communication all contribute to the power consumption of eye tracking, which we will describe next.

B. Image Sensor Basics

When exposed to light, an image sensor transforms optical signals in the scene to analog signals (using the photoelectric effect [46]). The analog signals are converted to digital pixel values through the readout chain (including the ADCs). Through the Mobile Industry Processor Interface Camera Serial Interface 2 (MIPI CSI-2) [55], the pixel values are then transferred to a host processor to undergo further algorithmic processing (e.g., eye tracking). These operations are necessarily serialized within a frame but can be overlapped across frames. Fig. 1 illustrates a typical overlapping between frames, where the next frame can start its exposure while the previous frame's pixels are being transferred out.

Frame Rate. A key performance metric of image sensors is the frame rate, quantified by Frames Per Second (FPS). Ideally, as is the case in Fig. 1, the MIPI transfer and host processing delay (of the current frame) is completely hidden by the exposure and the readout delay (of the previous frame). In this case, the frame rate is limited only by the sum of exposure time and readout delay. Note that the readout delay (tens of µs) is usually 3-4 orders of magnitude shorter than the exposure time (tens of ms).

Stacked Image Sensors. Image sensors today are increasingly integrating advanced *computation* capabilities through 3D die stacking technologies, presenting opportunities for architectural exploration. Nowadays almost all mobile image sensors are stacked [109]: the pixel array layer that converts photons to analog signals and the processing layer which contains the readout circuitry and the preprocessing image signal processor (ISP) are located on two separate dies.

With the additional stacking dimension, computational image sensors now routinely integrate digital logic (e.g., DNN accelerators [23], [24], [47]) and memories (e.g. DRAMs [119], and SRAMs [62], [111]). Moreover, stacking allows for heterogeneous integration, where the pixel array layer and the processing layer can each use their respective process node. A recent survey [40] shows that it is common for the processing layer to adopt a process node (e.g., 22 nm) that is several generations ahead of the one used by the pixel array layer (e.g., 65 nm) in order to accommodate high-density energy-efficient digital processing.

Digital Pixel Sensor. Our paper adopts the stacked Digital Pixel Sensor (DPS), a particular form of image sensor architecture that is gaining popularity [126]. In DPS, the processing layer has a per-pixel ADC, which inherently supports a global shutter (critical for high-speed capturing) and a per-pixel SRAM to store the digitized pixel value and naturally act as the input buffer to a digital accelerator. Examples of DPS include the imager prototypes by Meta [65], Samsung [111], OmniVision [15], and Sony [17].

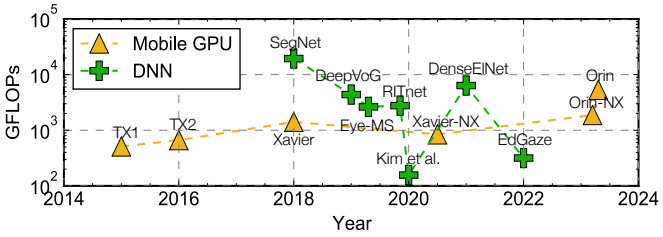


Fig. 2: The computational capabilities, quantified in GFLOPS, of today's mobile GPUs (using Nvidia Jetson series as examples) vs. the computational demands of state-of-the-art eye tracking algorithms (assuming a tracking rate of 120 FPS).

The recent trend in computational image sensors suggests that it is possible to integrate domain-specific accelerators into the processing layer of a stacked DPS sensor with minimal area overhead. For instance, under a 22 nm process node, our experiment shows that integrating a DNN processor merely introduces 5.8% of area overhead (Sec. VI-D). Many such designs have been proposed by previous works using Arithmetic Logic Units (ALUs) under a group of pixels [93] and CNN processor under the entire pixel array [47], [112].

C. Scaling Trends of Eye Tracking Technologies

Tracking Frequency. Eye tracking performance is mainly quantified by the tracking frequency: how many gaze estimations can be made in a second. This paper assumes a 120 Hz tracking rate, which is shown to be sufficient for many eye tracking use cases (e.g., AR/VR) [33], [60] and is on par with commodity eye trackers [5], [7], [12].

Meeting the tracking rate does not pose any issue as technologies scale. This is because the speed of the eye tracking algorithm on recent mobile processors with embedded accelerators (e.g., GPUs), is already higher than that of the image sensor's capturing rate. Thus, the algorithm delay can be hidden by the exposure time (Fig. 1).

To quantify this argument, Fig. 2 compares the Giga Floating Point Operations per second (GFLOPs) of the mobile GPUs on Nvidia Jetson series [8] with the GFLOPs requirement of a set of state-of-the-art eye tracking algorithms operating at 120 Hz [34], [49], [75], [129]. The GPUs and algorithms are placed on the x-axis based on their release dates. As mobile computing capabilities (unsurprisingly) increase over the years, eye tracking algorithms also become more efficient. Hence, the tracking rate requirement can be adequately met by today's mobile GPUs.

Tracking Latency. While tracking rate is unlikely an issue, tracking latency is. Tracking latency is the delay between the start of a frame exposure and when the eye tracking algorithm finishes on that frame (Fig. 1). Our measurements of commercial eye trackers (e.g., HTC Vive Pro Eyes [6]) show that the tracking latency is usually in excess of 15 ms, enough

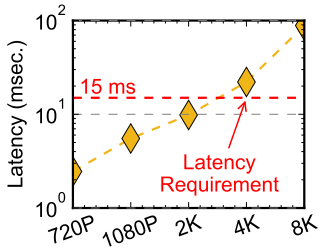


Fig. 3: MIPI communicating latency under different image resolutions. The red line shows the eye tracking latency requirement (15 ms).

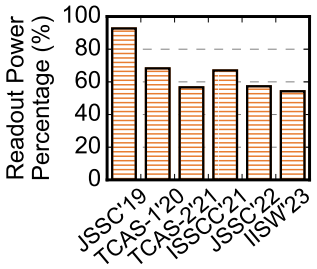


Fig. 4: Percentage of image sensor power attributed to by the readout circuitry; data from six recent sensors [64], [97], [98], [99], [111], [114].

for many gaze-contingent AR/VR systems to report tracking delay as a main cause of user-observable artifacts [20], [66]. 15 ms is roughly the end-to-end latency under a 120 Hz tracking rate when capturing and processing are fully overlapped (as shown in Fig. 1): 15 ms $\approx \frac{1s}{120} \times 2$.

Exposure time accounts for a large proportion of the tracking latency. Simply reducing the exposure time, however, has noise implications, because the Signal to Noise Ratio (SNR) of image sensing drops *quadratically* with exposure time [106]. Therefore, the downstream operations must be robust against exposure time changes. Among other components, MIPI CSI-2 transfer is poised to become a latency bottleneck as image resolution increases in future. Fig. 3 shows the MIPI latency under various image resolutions [16], [19]. As the image resolution increases to 4K, the transmission latency (22 ms) alone already surpasses the end-to-end latency requirement, which means the MIPI latency can not be hidden by the processing of the next frame.

This paper will demonstrate techniques that reduce the MIPI transfer latency and, consequently, the subsequent eye tracking algorithm latency while being robust against (even drastic) changes to the exposure time.

Power Consumption. Eye tracking power consumption is known to be high. Two recent eye tracking algorithms, RITnet and EdGaze, consume 2.3 W and 1.9 W, respectively, on a mobile Volta GPU [14]. Apart from the computation power, the power consumption of image sensors has been steadily increasing. Recent high-speed (120 FPS) image sensors routinely consume hundreds of milliwatts [3], [77] or even a few Ws [1], [13], taking 10-60% of the total power budget of a typical standalone VR device (around 3-6W [2], [4], [9], [78]).

The image sensor's power is dominated by two components: the sensor-host data transfer and the readout circuitry. Measurements show that transmitting one byte from the image sensor via MIPI CSI-2 interface consumes about 100 pJ energy [83] which translates to 300 mW when transmitting 4K images at 120 FPS. The readout peripheral circuit that converts the analog pixel value to locally stored digital bits is another dominating component in the image sensor [38]. While power-efficient ADC design is an active area of research, a survey on the recent image sensors from the past decade [85] shows that the readout circuitry still consumes 66% of the sensor's power on average, as shown in Fig. 4.

Summary. Our goal is to significantly reduce the power consumption and latency of eye tracking without hurting the (already sufficient) tracking frequency. We focus on co-designing the computational image sensor front-end with the eye tracking algorithm, while leaving further optimization of the host SoC hardware to future work. Optimizing the image sensor not only directly reduces the sensor readout and sensor-host communication power, but also indirectly reduces the amount of work off-sensor algorithms perform.

Hardware acceleration of the off-sensor operations (e.g. segmentation) is orthogonal and complementary to our front-end solution and, thus, out of scope. In this paper, we assume a standard systolic array architecture to execute any DNNs and claim no novelty for the neural processing unit (NPU) design.

III. Sparse Sampling-Based Eye Tracking

We first describe the sparse sampling algorithm (Sec. III-A), followed by an eye segmentation algorithm that is robust against sparse inputs (Sec. III-B). We then discuss how two algorithms are jointly trained (Sec. III-C). This section focuses on the algorithm and leave hardware design to the next section.

A. Sparse Sampling Algorithm

Intuition and Overview. To reduce latency and power consumption, our idea is to perform sparse sampling *inside the sensor*. In-sensor sampling has two advantages. First, it reduces the amount of pixels that have to be read out and transferred to the host, two of the main contributors to sensor power consumption. Second, by reducing the data volume, we also reduce the cost of the downstream eye tracking algorithm.

The overall algorithm pipeline is shown in Fig. 5. Each frame first gets sparsely sampled by inside the sensor; the sampled pixels are then transmitted to the host, which executes the eye segmentation and gaze prediction to produce the ultimate gaze information.

Conventional image sampling aims to maximize image reconstruction quality for human vision [45], [76]. Instead, our sampling strategy leverages the unique characteristics of, and is thus tailored to, the eye tracking task. In particular, in eye tracking only the fore-ground eye parts (e.g., pupil, iris, cornea) contribute to the final gaze information. Naturally, we can approach sampling in two stages: first localizing the fore-ground parts of the eye as the region-of-interest (ROI), followed by sampling within the ROI. The ROI prediction DNN is jointly learned with subsequent sampling and downstream eye segmentation to minimize end-to-end loss.

Our two-stage sampling algorithm consists of three serialized stages: eventification, ROI prediction, and sampling. Let us describe the three stages in detail.

Eventification. While one could apply generic, heavy-duty object detection DNNs to detect ROIs (e.g., Mask R-CNN [56]), the cost of executing such networks would be prohibitively high for in-sensor computing.

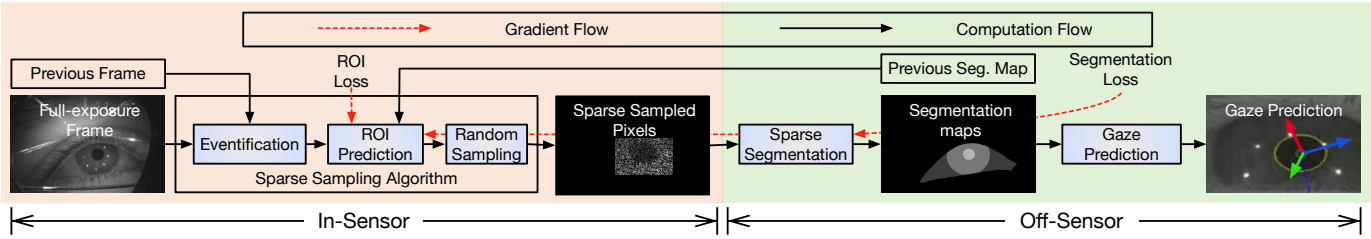


Fig. 5: Our sparse sampling-based eye tracking pipeline. Each frame first gets sampled by our sparse sampling algorithm inside the sensor to dramatically reduce the sensor-host data volume (Sec. III-A); the sampled pixels then go through a sparse eye segmentation algorithm on the host, which is designed to be robust against sparse inputs (Sec. III-B). The ROI prediction algorithm and the sparse segmentation algorithm are jointly trained to maximize end-to-end tracking accuracy (Sec. III-C).

To design a lightweight ROI detection algorithm, the key observation we leverage [49] is that in virtually any eye tracking scenario (e.g., AR/VR), the near-eye camera is tightly mounted on the headset, which is in turn tightly mounted on the head. This means the background in eye images is stationary: there is no relative motion between the camera and the eye background. Therefore, any pixel intensity *changes* between consecutive frames inherently indicate the foreground, moving eye parts. The inter-frame pixel differences, thus, provide a natural guide to ROI prediction.

Therefore, the first step in our sampling algorithm is to obtain inter-frame pixel difference, which is expressed as:

$$E_{t+1}(x,y) = \Phi(|F_{t+1}(x,y) - F_t(x,y)|, \sigma) \tag{1}$$

where $F_t(x,y)$ and $F_{t+1}(x,y)$ are the pixel values at the (x,y) coordinates at time T and $(T+1)$, respectively; Φ is an activation function which outputs 1 if the difference is greater than the threshold σ (and 0 otherwise). The threshold is a parameter that can be tuned for a specific application or scenario. We empirically find that $\sigma = 15$ yields good results.

The resulting E is essentially a binary event map, where each pixel value indicates whether the corresponding pixel has changed significantly across frames (i.e., an event has occurred) and, thus, belongs to the foreground eye parts.

ROI Prediction. With the guidance of the event map, we design a lightweight ROI prediction network. Our ROI prediction network is intentionally small; it contains three convolution (`Conv`) layers followed by two full-connected (`FC`) layers, amounting to only 2.1×10^7 MAC operations. The event map is used as the input to `Conv` layers.

While event maps are generally effective, there are corner cases where events are not indicative of foreground parts (e.g., blinks, saccades). To improve the robustness of our ROI prediction, we feed back the segmentation map from the previous frame as a corrective cue similar to prior work [49].

Random Sampling. Given an ROI, we randomly sample the pixels inside the ROI. We find that random sampling is effective even at high sampling rates (Sec. VI-F). There are many other sampling alternatives we consider; none works as well. We will quantitatively compare across different sampling strategies in Sec. VI-E; here we provide an intuitive account. For instance, one can sample the entire image rather than just the ROI, but the non-ROI regions of an eye image make no contribution to eye tracking result, wasting precious sampling budget. Alternatively, one can uniformly, rather than randomly, sample the ROI, which would simplify the hardware. Our results, consistent with prior findings in compressed sensing [29], [30], show that uniform sampling significantly reduces the eye segmentation accuracy, suggesting the difficulty of reconstructing the eye parts from uniformly sparse samples. Finally, one can consider using an additional network to predict which pixels to sample. The computation cost of doing so is prohibitively high with little accuracy benefit.

B. Robust Eye Segmentation From Sparse Inputs

Unlike existing eye segmentation algorithms that operate on full eye images [34], [73], [75], [129], our algorithm operates on sparse images (about 5% of the pixels as Sec. VI-A shows). Using sparse images means the inputs are susceptible to noise, hurting the accuracy of the algorithm [42].

Specifically, our experiment (Sec. VI-A) shows that existing Convolutional Neural Network (CNN)-based algorithms often struggle to retain high accuracy at low sampling rates. CNN-based algorithm's accuracy drops rapidly once the sampling rate (the percentage of pixels that are retained) is below 50%. This is because CNNs inherently rely on local information rather than global information to make predictions. Consequently, as the sampling rate increases (i.e., fewer pixels), less local information is retained, which leads to accuracy drops.

Instead, we propose a Vision Transformer (ViT)-based segmentation algorithm. Unlike CNN-based algorithms, ViT leverages an attention-based mechanism that takes into account information from all valid pixels within the input image [121]. Even when the sampling rate is low, ViT still can extract the relationship among pixels that are far away from each other.

Our ViT algorithm consists of an encoder and a decoder (Fig. 6). The ViT encoder uses 12 multi-head attention (MHA) modules, similar to the network architecture in Strudel et al. [117]. Each MHA layer has three heads with a channel size of 192. To correlate global information, the MLP operations in MHA module compute across all image tokens. Similarly, ViT decoder uses two MHA layers with the same setting and ends with an argmax layer for the final segmentation prediction.

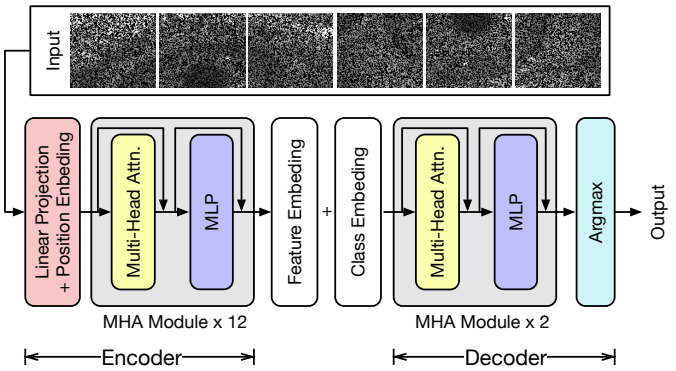


Fig. 6: Overall architecture of our ViT segmentation, consisting of an encoder and a decoder. The encoder is composed of a linear projection and 12 MHA modules. The decoder comprises two MHA modules and an argmax layer.

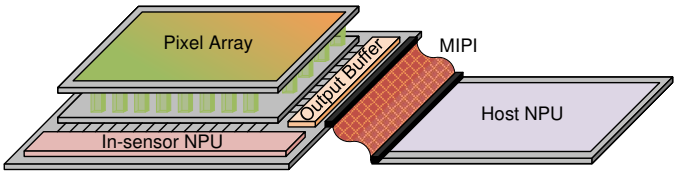


Fig. 7: High-level architecture of our eye tracking system. The image sensor is connected to the host NPU through the MIPI interface. The sensor has a 2-layer DPS pixel array (as with many recent image sensors [109]), an in-sensor NPU, and an output buffer. Our architectural augmentations lie in augmenting the bottom layer of the pixel array.

Note that a ViT network can be readily executed on a typical DNN accelerator (e.g., a systolic array) [36], [128]. Optimizing the accelerator architecture for ViT networks is an active area of reasearch [82], [122]; we leave it to future work to co-design the accelerator with our ViT network.

C. Training Procedure

The end-to-end tracking algorithm contains two DNNs, one for ROI prediction and the other for ViT segmentation. We propose a joint training procedure to improve the overall accuracy. Two loss terms guide our training: a segmentation loss and an ROI loss. The segmentation loss is a cross-entropy loss that governs the accuracy of eye segmentation [34], while the ROI loss uses the conventional mean-square-error loss that governs the ROI prediction accuracy. During training, the segmentation loss is back-propagated to both the ROI prediction and the sparse segmentation DNN. We explicitly mask the gradients belonging to the pixels that are not selected by the random sampling. That is, only the unmasked gradients are used to update the ROI prediction DNN.

IV. Architectural Support

We first introduce the overall BlissCam system operation (Sec. IV-A) and present an overview of the sensor architecture (Sec. IV-B), followed by the detailed description of the sensor hardware designs (Sec. IV-C).

A. BlissCam System Overview

We co-design BlissCam to support the learned sparse sampling algorithm. The system consists of a computational image sensor and a host NPU connected by the MIPI CSI interface. The system organization is illustrated in Fig. 7. Our main contribution is to architecturally augment the image sensor to support in-sensor sparse sampling (Sec. III) with minimum hardware overhead while leaving the host NPU as is to perform the eye segmentation and gaze estimation tasks.

Fig. 8 depicts the system timing diagram of BlissCam, which has two main differences when compared with the original eye tracking pipeline in Fig. 1. First, each frame now goes through three additional in-sensor processing stages: eventification, ROI prediction, and in-ROI sampling. Second, there is a new constraint when pipelining across frames: Frame_t's ROI prediction must wait for the segmentation map of Frame_{t-1} to be sent back from the host via the MIPI CSI interface. This dependency is purely algorithmic: the previous frame's segmentation map is used as an input to the ROI prediction of the current frame (Sec. III-A). The two dependencies are denoted by the arrows in Fig. 8.

Observing Fig. 8, it would seem that additional in-sensor operations would increase the eye tracking energy and latency. As we will describe in the rest of this section, however, the hardware design is such that the additional in-sensor operations introduce negligible latency and energy overhead (2-3 orders of magnitude lower) compared to that of a baseline frame; meanwhile, in-sensor sampling significantly reduces the data volume involved in readout, MIPI CSI transfer as well as the off-sensor segmentation work which now operates on far fewer pixels. Since both the readout and the MIPI transfer contribute the major sensor energy, and the MIPI transfer contributes the major sensor latency, the sensor's overall energy and latency are notably reduced.

B. Sensor Architecture

The main design consideration for the BlissCam sensor chip is to support the various in-sensor operations in addition to the normal imaging mode. Although prior works have introduced designs that could meet one or two of the required operations (see summary in Sec. VII), none supports the full gamut of the required in-sensor operations. Instead of employing dedicated hardware for each function which would incur intolerable area costs at the pixel level, our design principle, is to maximally reuse existing hardware across multiple operations while introducing only minimal augmentation.

As shown in Fig. 7, the proposed BlissCam sensor consists of a pixel array, an in-sensor NPU, and an output buffer. Our hardware augmentation is limited to the pixel array while adopting the standard design strategies for the in-sensor NPU (i.e. systolic array) and the output buffer (i.e. parallel-in-serial-out shift register). For each frame, the pixel array captures an array of pixels and generates a binary event map (i.e., eventification). The event map is transferred to the in-sensor NPU, where the ROI prediction DNN resides. The ROI bounding box is then fedback to the pixel array, which

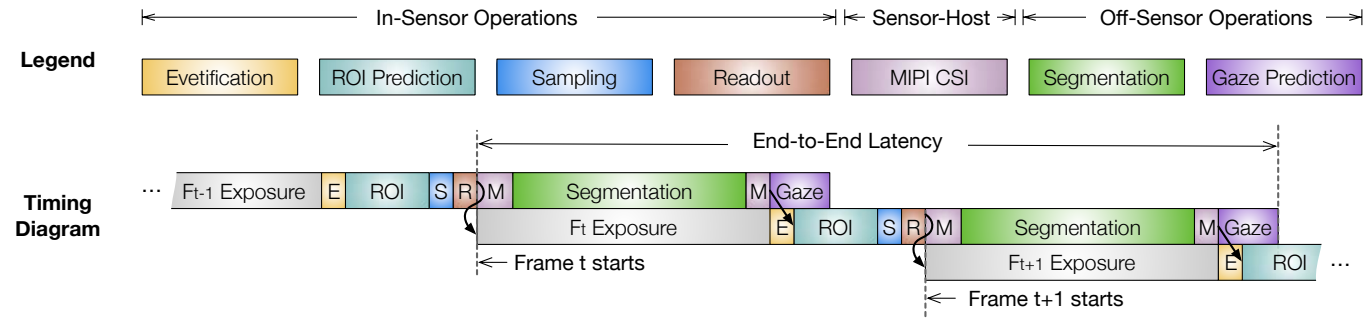


Fig. 8: Timing diagram of our eye tracking system, which includes operations both in sensor (Exposure, Eventification, ROI Prediction, Sampling, Readout) and off sensor (Eye Segmentation and Gaze Prediction). We overlap processing of different frames to ensure high frame rate while respecting data dependencies (indicated by filled arrows). The figure is not drawn to scale; the additional latency introduced by the new in-sensor operations is much smaller compared to the exposure time, so the overall tracking rate is minimally impacted.

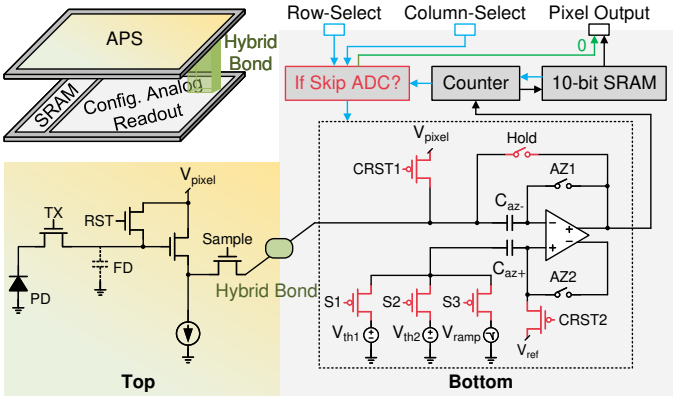


Fig. 9: The circuit diagram of the proposed DPS; the red components are new hardware added to a conventional DPS. The top layer is a standard pixel design (4T APS) that converts photons to charges, and the bottom layer performs eventification, analog memory, ADC, and sparse readout by reusing the same circuitry. Blue arrows: signals used to determine if the pixel performs ADC. Green arrows: signals that output 0 if the pixel skips ADC. Black arrow: output analog pixel value if the pixel performs ADC (sampled).

randomly samples pixels in the ROI and reads out only those sampled pixels to the output buffer. The output buffer connects to the MIPI interface, which transfers the pixels to the host.

The pixel array is implemented following a typical two-layer DPS architecture where the top layer is the array of pixel cells (e.g. a 4-transistor Active Pixel Sensor [95]), and the bottom layer consists of an array of per-pixel ADC and SRAM. The two layers are connected by per-pixel hybrid bonds. In Sec. IV-C, we will dive into our novel augmentation to each DPS pixel at the bottom logic layer to support additional operations (i.e., eventification and sampling).

C. Design

The detailed design of the pixel circuits, which are based on the standard DPS design, is illustrated in Fig. 9. For *each* pixel, a standard 4T Active Pixel Sensor (APS) circuit (responsible for converting photons to charges based on the photoelectric effect [46]) resides in the top layer; the bottom layer contains a 10-bit SRAM and a configurable analog readout circuit.

In a standard DPS, the pixel readout is performed by an Single-Slope ADC (SS ADC) for pixel quantization. The SS ADC operates as follows: a comparator receives the analog pixel value and a monotonically decreasing ramp signal (V_{ramp}) at its two input Auto-Zero (AZ) capacitors (C_{az+} and C_{az-}), respectively; the comparator's output will not toggle until the ramp signal crosses the analog pixel value; a counter counts the number of cycles it takes for the toggle of the comparator's output, and the counted cycles is the quantized pixel value.

Conventional SS ADC is a fixed-function unit that performs just the quantization. To perform the in-sensor operations required by our learned sampling algorithm, BLISSCAM augments the SS ADC with a few extra switch transistors and a simple logic unit (highlighted in red in Fig. 9) while reusing many existing ADC components—the AZ capacitors/switches, the comparator, and the counter. In a sense, we time-multiplex the same analog readout circuit between different operations, e.g., analog buffering, eventification, and normal quantization. Fig. 10 shows different configurations of the readout circuit.

We also minimally augment the circuitry to support sparse readout, where only sampled pixels within an ROI go through the ADC and MIPI interface. This is achieved by the "If Skip ADC" logic in Fig. 9, which is conditioned upon the row-select and column-select signals. Finally, we reuse the per-pixel SRAM for storing the eventification result and for in-ROI random sampling. We now discuss the circuit-level behaviors.

Eventification. Eventification generates a binary map as the input to the ROI prediction DNN. According to Eqn. 1, eventification requires the retention of the previous frame Frame_{t-1}, the subtraction between the current frame Frame_t and the previous frame Frame_{t-1}, and comparing the frame difference with predefined bipolar thresholds $\pm\sigma$.

To hold Frame_{t-1} during the exposure of Frame_t, we

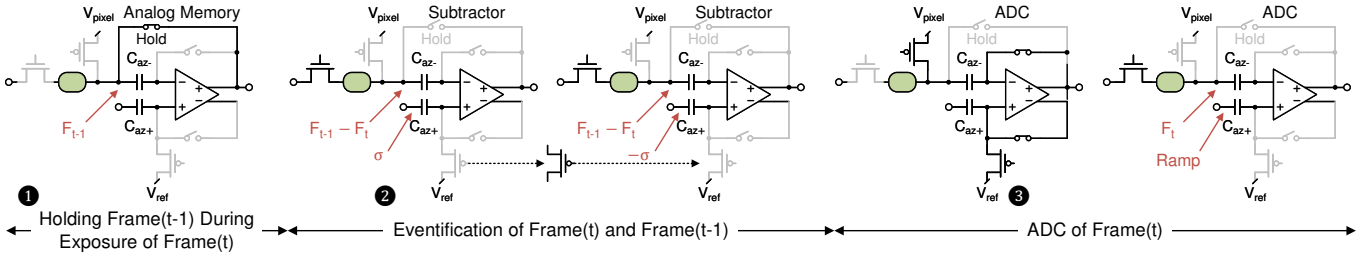


Fig. 10: Different configurations of the pixel's analog readout circuit. The quantization is only performed for sampled pixels.

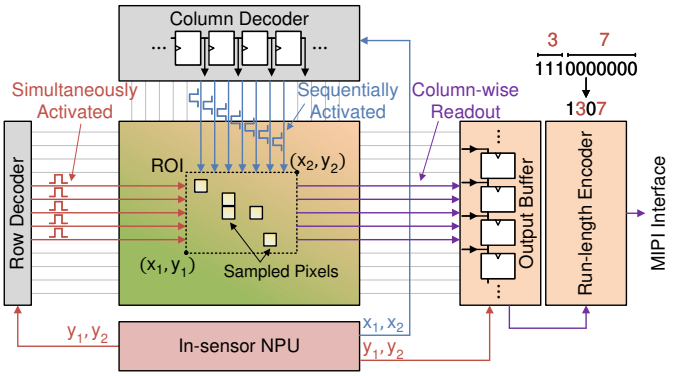


Fig. 11: The pixel array architecture that allows sparse readout. The in-sensor NPU generates the coordinates the two ROI corners (x_1, y_1) and (x_2, y_2). The coordinates are used to driven the row/column decoders to select the ROI. All pixels inside the ROI are read to the output buffer (in a column-wise manner), but only the sampled pixels are quantized while the unsampled pixels output 0s. The output buffer transmits the bits to the MIPI interface through a run-length encoder.

configure the comparator as an analog buffer by closing the $Hold$ switch to form a negative feedback loop, as shown in Fig. 10 ❶. Frame_{t-1} is held on C_{az-}, one of the two input AZ capacitors of the analog readout circuit.

The subtraction is done through a switched-capacitor configuration of the comparator circuit. As shown in Fig. 10 ❷, with $Hold$ open to disconnect the negative feedback loop, the charges of the new frame Frame_t are thus transferred onto C_{az-} and the result is naturally the frame difference $(\text{Frame}_{t-1} - \text{Frame}_t)$.

To compare against a threshold, we utilize the comparison function that an ADC intrinsically performs by simply connecting the other input AZ capacitor, C_{az+}, to the thresholding value σ. The comparison result is naturally the output of the analog readout circuit, as shown in Fig. 10 ❷. Note, however, that the mathematical formulation (Eqn. 1) requires comparing against the absolute value of the threshold, so we apply $\pm\sigma$ sequentially through V_{th1} and V_{th2} (Fig. 9).

ROI Prediction. Following the eventification step, the output of the frame difference compared with the thresholds forms the binary event map and is stored using the per-pixel SRAM. The SRAM is read by the in-sensor NPU to execute the ROI prediction DNN. Our design uses a systolic array-like accelerator, and we claim no novelty here (see Sec. V).

The output of the ROI prediction is four numbers (x_1, x_2, y_1, y_2), representing the xy-coordinates of the two opposing corners of the ROI box. Fig. 11 shows how these values are used to drive the ROI selection. The two row coordinates (y_1, y_2) are sent to the row decoder and the output buffer, and the two column coordinates (x_1, x_2) are sent to the column decoder. The row decoder activates all the rows between y_1 and y_2 simultaneously, whereas the column decoder activates all the columns between x_1 and x_2 *sequentially* because the read-out to the output buffer is necessarily column-by-column. Note that not all the pixels in the ROI will be read out; only those that are sampled do, the mechanism of which will be discussed next.

Random Sampling. A random bit is generated locally at every pixel to determine whether the pixel will be quantized and read-out. To avoid additional in-pixel circuitry, we utilize the metastability [118] of the inherent 10-bit SRAM for random bit generation. The randomness comes from the meta-stability of a typical 6-transistor (6T) SRAM cell when the SRAM is powered-up. The meta-stable state will randomly latch to a 1 or a 0 due to random noise when the cell is powered-up [120]. The randomness is not spatially correlated due to the differential signaling of the cross-coupled pair.

Although using SRAM for random bit generation requires intermittent SRAM power-up/down, it does not affect the system timing nor the SRAM's memory function (i.e. storing the result of eventification and ADC). This is because the SRAM is intrinsically power-gated during its inactive periods in the normal functional pipeline: the SRAM is powered-down after the event map is used by the ROI prediction DNN, and will then be powered up to store the quantized pixels. We leverage the SRAM's intrinsic duty cycle to generate the random bits for every frame.

To control the sample rate, during a one-time offline calibration all the SRAM cells are powered up and down multiple cycles to profile the distribution of the sum of the 10 power-up bits in each pixel. From the profiling result, we build a look-up table that translates a sampling rate to a threshold θ. The θ is in 4-bit, thus the table has only 2^4=16 entries to cover the sum which ranges from 0 to 10. In our simulation, we use the statistics from measurements in prior work [58], [125].

During run time, at each power-up event the counter of each pixel sums the 10 power-up bits by counting the number of 1s

in that pixel. This (4-bit) number is compared with θ in the "If Skip ADC" logic in Fig. 9. Only when the sum surpasses θ will the pixel be actively sampled. Summing the power-up bits of all the 10 bits in a pixel mitigates the non-uniform randomness across SRAM cells due to the process variation [58].

Sparse Readout. At this stage, the sampled pixels must be read out. Fig. 11 shows that all the pixels within the ROI are transferred to the output buffer in a column-wise manner, where the column select signals are sequentially activated (as is in the baseline DPS). However, only when a pixel is sampled will it be quantized by the ADC, which is controlled by the "If Skip ADC" logic in Fig. 9. If the pixel is sampled, its comparator is configured to the normal SS ADC (Fig. 10 ❸). If a pixel is not selected, the "If Skip ADC" logic connects a constant 0 to the pixel's output port.

The output buffer thus contains both the sampled pixels and the un-selected ones within the ROI. Since only approximately 20% of the pixels within the ROI are sampled, we use the run-length encoder [54] to compress the data. For example, a sequence of 1110000000 is compressed to 1307 where 3 and 7 denote the number of 1s and 0s in the sequence, respectively. A corresponding run length decoder is implemented in the host NPU to decompress the ROI images before being processed by the eye segmentation algorithm.

V. Experimental Setup

Hardware Configurations. The overall hardware system consists of a custom designed image sensor and a conventional DNN accelerator (NPU); we claim no contribution in the latter. Without losing generality, we assume a systolic array-like NPU, which consist of a 32×32 MAC array operating at 1 GHz. This NPU is responsible for computations outside the sensor. The NPU's global buffer is sized at 2 MB and is banked at a 128 KB granularity. We also assume a systolic array-style NPU sitting at the bottom layer of the image sensor. The NPU is consists of an 8×8 MAC array clocked at 0.5 GHz with a 512 KB SRAM, which is sized to hold the input and intermediate feature maps needed for ROI prediction.

Experimental Methodology. All the digital logic is implemented in RTL. We synthesize, place, and route the design using an EDA flow consisting of Synopsys and Cadence tools. The SRAMs are compiled by an ARM memory compiler. Power is simulated using Synopsys PrimeTimePX, with fully annotated switching activity. The pixel design on the sensor top layer follows that in Seo et al. [111]. The analog circuit on the bottom layer is implemented in standard CMOS 65 nm technology and simulated using Cadence Virtuoso.

Following the typical technology nodes in today's image sensors, we assume that the top layer of the image sensor uses a standard CMOS 65 nm process node, the bottom analog and logic layer uses a 22 nm process node, and the off-sensor NPU uses a 7 nm process node. We use the synthesis results from a TSMC 16 nm FinFET library and scale the results to other process nodes using the DeepScaleTool [108], [115], which models the classic CMOS scaling by "*fitting published data by a leading commercial fabrication company for silicon fabrication technology generations from 130* nm *to 7* nm."

We model noises in the image sensor, following classic analytical models of various noise sources [27], [48], [95], [106]. Specifically, the analog readout circuits (on the bottom layer of the sensor) are carefully designed such that its read noise does not introduce functional errors to the binary eventification and ADC quantization. We model the photon shot noise using the classic method (drawing from a Poisson distribution) [95] and considered it during training and evaluation.

The DRAM parameters are modeled after Micron 16 Gb LPDDR3-1600 (4 channels) as detailed in its datasheet [10]. The calculation of DRAM energy is based on Micron's System Power Calculators [11] using the memory traffic, including kernels and activations of segmentation ViT. We use the energy per byte over the MIPI CSI interface from Liu et al. [83].

Algorithm Baselines. To evaluate the accuracy of our ViT-based eye segmentation algorithm (specifically designed to leverage the sparse eye image input; Sec. III-B), we compare against two state-of-the-art eye segmentation algorithms, both operate on dense eye images: RITNET [34], which uses an encoder-decoder architecture, and EDGAZE [49], which uses depthwise separable convolution network.

We follow the same training procedure in prior work [49], [75] and use OpenEDS [53], a widely-used eye tracking dataset. We train the eye segmentation algorithm using a batch size of 4 with 250 epochs. We train the ROI prediction network for 100 epochs with a batch size of 8.

System Variants. To tease apart the contribution of different components in our system and to support ablation studies, we compare against the following variants:

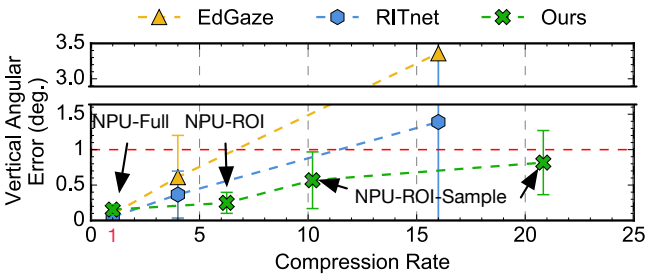
- NPU-FULL: represents a conventional eye tracking system: a non-computational image sensor with a host NPU. The sensor transmits the full-size eye images to the host NPU, which executes the eye segmentation algorithm.
- NPU-ROI: this variant has the same hardware configuration as NPU-FULL, except the host NPU executes the ROI prediction DNN to extract the ROI, on which the subsequent eye segmentation algorithm operates.
- S+NPU: same as our proposed design except it executes sparse sampling in the digital domain inside the sensor.

VI. Evaluation

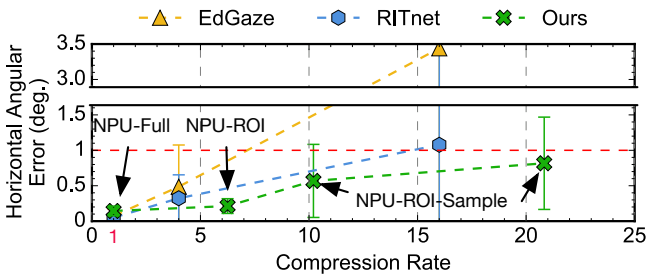
This section starts by demonstrating that BLISSCAM achieves adequate accuracy against baselines even when significantly reducing the pixels (Sec. VI-A). Following this, we demonstrate that our sensor design reduces the overall energy consumption (Sec. VI-B) and tracking latency (Sec. VI-C). We show that our hardware augmentation introduces little area overhead (Sec. VI-D) and that our sampling strategy out-performs alternatives (Sec. VI-E). Finally, we conduct a sensitivity study to understand how BLISSCAM's performance and energy savings vary under diverse settings (Sec. VI-F).

A. Accuracy vs. Compression Rate

Our eye tracking algorithm achieves higher accuracy compared to existing eye tracking algorithms across a range of



(a) Vertical angular error.



(b) Horizontal angular error.

Fig. 12: End-to-end gaze prediction vs. compression rate (uncompressed size over compressed size; 1 for full frame). The error bars denote one standard deviation. We annotate different variants of our method for ablation studies. **NPU-Full** opeates on full eye images. **NPU-ROI** applies operates on ROI images. **NPU-ROI-Sample** is our full-fledged pipeline.

compression rates (uncompressed size over compressed size). Fig. 12 presents the accuracy-vs-compression-rate comparisons on both vertical angular error (Fig. 12a) and horizontal angular error (Fig. 12b). The input images to the two baseline algorithms are downsampled by different amounts to achieve different compression rates.

Across all compression ratios, our algorithm consistently maintains the gaze estimation accuracy within the acceptable error threshold (1°) in both directions [5], [6], [7], [12]. Specifically, we achieve a $20.6\times$ data reduction with 0.8° vertical angular error and 0.7° horizontal angular error. Unless otherwise noted, this is the compress rate we will use in the rest of the evaluation. Our algorithm also consistently outperforms existing algorithms across all compression rates with much higher robustness. The robustness can be seen by comparing the standard deviation of our method with that of the two baselines: our method has a much smaller accuracy variation, showing a stronger ability to tolerate temporal drifts. While not shown in the figure, our algorithm is also more computationally efficient compared to RITNET and EDGAZE. For instance, compared to RITNET, we reduce the MAC operation counts by a factor of 4.

B. Energy Reduction

BLISSCAM also significantly reduces the eye tracking energy consumption. Fig. 13 compares BLISSCAM with three variants disucssed as the end of Sec. V using a stacked bar plot, dissecting the contributions of different hardware components.

Compared to NPU-FULL, BLISSCAM achieves a $4.0\times$ energy reduction. The reduction comes from three sources: the analog readout energy, MIPI data transfer energy, and the off-sensor work (e.g., eye segmentation NPU and accessing the on-chip buffer). The latter is especially significantly, contributing to 60.1% energy of NPU-FULL. By predicting ROIs and operating only on sampled pixels, BLISSCAM reduces the off-sensor work significantly.

While performing ROI prediction reduces overall energy, where ROI prediction is executed also matters. NPU-ROI executes ROI prediction on the host SoC, taking advantage of the more advanced process nodes, thus reducing the energy spent on executing ROI prediction. In contrast, S+NPU executes ROI prediction inside the sensor, which has the advantage of reducing readout and MIPI energy but increases the ROI prediction and buffer energy, because the process node of the sensor uses an older process node. As a result, S+NPU actually increases the energy by $1.1\times$ over NPU-ROI, mainly due to the high leakage power of the in-sensor frame buffer. The leakage power of the frame buffer can not be eliminated by power gating because the frame buffer must continuously retain the previous frames for eventification.

BLISSCAM combines the benefits of both S+NPU and NPU-ROI by storing the previous frames in analog memory and executes eventification in the analog domain. That way, BLISSCAM reduces both the in-sensor frame buffer energy and the MIPI data transfer. As a result, BLISSCAM achieves $1.7\times$ and $1.6\times$ energy reduction compared to S+NPU and NPU-ROI, respectively.

Overhead. The results above show that the overhead introduced by BLISSCAM is clearly out-weighted by its benefits. In particular, there are two main energy overhead: the additional transfer of the previous frame's segmentation map from the host SoC (in assisting ROI prediction; see Fig. 8) and the RLE (see Fig. 11). The two sources account for only 0.6% and 0.04% of the overall energy, respectively.

C. Tracking Latency and Frequency

The energy saving of BLISSCAM comes with little impact on the overall tracking frequency while significantly reducing the tracking latency. To ensure a fair comparison, we choose the same process node combination across all variants, and the clock rates of sensors and host SoCs are set to 0.5 GHz and 1 GHz, respectively, across all variants.

Fig. 14 compares the end-to-end tracking latency under the 120 FPS requirement. BLISSCAM reduces the tracking latency by $1.4\times$ over NPU-FULL, primarily because the segmentation DNN latency is accelerated by $7.7\times$, since it operates only on a small amount of pixels (10.8%). The average execution time of eye segmentation is 0.87 ms with a standard deviation of 0.48 ms. The latency varies across frames, because different frames

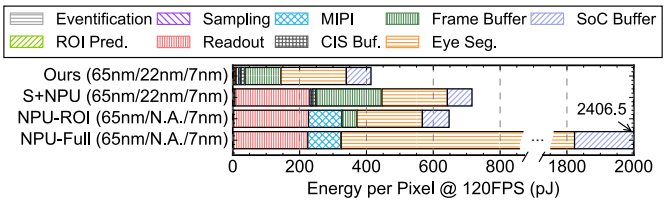


Fig. 13: Comparison of energy savings across different sensor-SoC designs at 120 FPS. Numbers inside each parenthesis represent the process node of sensor analog layer, sensor digital logic layer and host SoC, respectively.

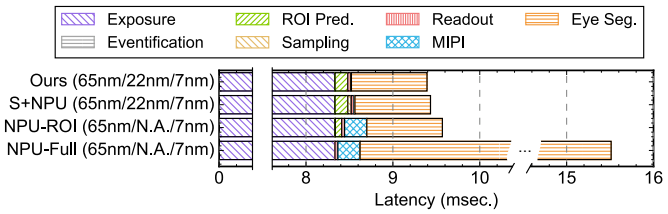


Fig. 14: End-to-end latency comparison across different sensor-SoC designs at 120 FPS. Our sensor design does not affect the sensor exposure and achieves similar latency as S+NPU and NPU-ROI. Numbers inside each parenthesis represent the process node of sensor top layer, sensor bottom logic layer and the host SoC, respectively.

have different ROI sizes: the average ROI size is 34257.8 pixels with a standard deviation of 18803.6.

Even with additional work in the analog domain, our latency is similar to that of S+NPU and NPU-ROI. This is because the latency in all three schemes is by far dominated by the exposure time, which is held constant. The additional computations introduce a latency overhead that is orders of magnitude shorter than the exposure time. For instance, compared to a 8.3 ms exposure time, eventification and ROI prediction introduce an overhead of 5 µs and 150 µs, respectively.

Because the in-sensor analog operations are much faster than the exposure time, BLISSCAM also has little effect on the exposure time (see Fig. 8). Overall, BLISSCAM reduces the overall exposure time by only 1.8%, which has a minimal impact on the overall eye tracking accuracy (as results in Fig. 12 factor in this exposure time change).

D. Area Estimation

While the power and the timing of the proposed DPS are directly obtained from circuit simulations, the area has to be estimated: DPS consists mostly of analog circuitry whose area is sensitive to manual layout and, thus, is not directly available from synthesis. Our pixel area estimation is based on previous DPS designs that have similar complexity.

Specifically, compared to a typical DPS with the ADC function only, our hardware augmentation to support functional multiplexing takes up only 7 extra switching transistors and simple digital logic (red components in Fig. 9), whose area is estimated to be comparable to 12 single-bit SRAM cells. The bottom layer of our design has 2 capacitors (233 fF each),

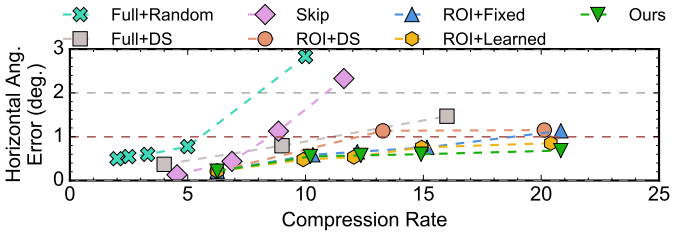


Fig. 15: Comparison between our sparse sampling and other sampling alternatives. Our method can retain acceptable accuracy even at high compression rates.

one comparator, 13 switching transistors, 10 6T SRAM cells, and trivial digital logic (a 4-bit digital comparator, 21 gates) in 22 nm technology.

Comparably, a stacked DPS by Meta [65] has 2 capacitors, one comparator, 28 switching/logic transistors, and 10 6T SRAM cells on its bottom layer, achieving 4.6 µm pixel size in 65 nm node. Another stacked DPS by Samsung [111] has one comparator, one positive-feedback amplifier, and 22 6T SRAM cells on its bottom layer, achieving 4.95 µm pixel size in 28 nm node. Thus, we choose a pixel pitch of 5 µm × 5 µm.

With this pixel size, the pixel array (640×400), the in-sensor NPU, and the output buffer (including run-length encoder) attribute to 6.4 mm^2, 0.4 mm^2, and 0.1 mm^2, respectively. The run-length decoder area on the host is also negligible; it is estimated to be less than 0.1% of the host area.

E. Comparison with Sampling Alternatives

We show that the in-ROI pseudo-random sampling strategy outperforms other sampling alternatives:

- FULL+RANDOM: a method that uniformly at random samples the full-size frame without ROI prediction
- FULL+DS: a method that uniformly downsamples the full-size frame without ROI prediction
- SKIP: a method that detects the event density within each frame to determine whether to skip subsequent eye segmentations and reuse previous segmentation results [49]
- ROI+DS: a method that uniformly downsamples within the predicted ROI
- ROI+FIXED: a method that uses dataset statistics to overfit a fixed ROI sampling mask offline
- ROI+LEARNED: a method that uses an additional ViT network to learn the pixel sampling within the ROI

Fig. 15 compares the horizontal angular error under different compression rates. BLISSCAM consistently outperforms all other methods across all compression rates. The highest accuracy gains are achieved against the full-frame methods, showing the benefits of ROI prediction. At a 21× compress rate, only ours and ROI+LEARNED can achieve an accuracy below the tolerable threshold of 1°. ROI+LEARNED, however, requires an additional in-sensor DNN to predict the sampling pattern, introducing intolerable pixel-wise overhead.

F. Sensitivity Study

Frame Rate. Fig. 16 shows how the end-to-end horizontal gaze accuracy (left y-axis) and energy saving over NPU-FULL

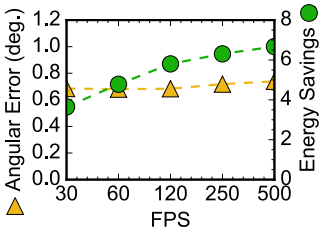


Fig. 16: Sensitivity of end-to-end gaze error and energy saving over NPU-FULL with respect to frame rate.

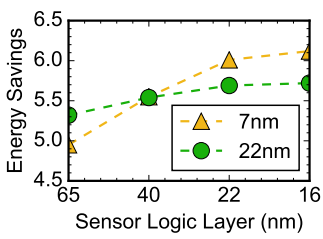


Fig. 17: Energy saving over NPU-FULL with respect to logic layer's process node under two SoC process nodes.

TABLE I: Sensitivity of gaze error, standard deviation, and energy savings (over NPU-ROI) to ROI reuse window.

Reuse Window	1	4	16
Vertical Angular Error (std.)	0.25 (0.15)	0.49 (0.30)	0.75 (0.69)
Energy Savings	0%	0.023%	0.029%

(right y-axis) change with the prescribed frame rate. The overall gaze error slightly increases by 0.03° as the frame rate increases from 30 FPS to 500 FPS. The primary contributor to the accuracy drop is that a higher frame rate reduces exposure time, which leads to a lower SNR (primarily driven by the photon shot noise [106]). Nevertheless, BLISSCAM retains tolerable gaze accuracy (1°) even at 500 FPS. As the frame rate increases from 30 FPS to 500 FPS, the energy saving over NPU-FULL increases from 3.6× to 6.7×. The energy reduction is because a higher frame rate means shorter exposure time, which reduces the retention time of the analog frame buffer and reduces the leakage energy.

Process Node. Throughout the sensitivity study we keep the process nodes for both the top layer and the bottom analog circuits fixed; and we synthesize the logic layer and off-sensor NPU with a TSMC 16 nm FinFET library, and use DeepScaleTool [108], [115] to scale them to different nodes. We note that mixing technology nodes *is* the norm in image sensors, as discussed in Sec. II-B.

Fig. 17 shows the sensitivity of the energy saving (over NPU-FULL) as the sensor and SoC process nodes change. We study two SoC process nodes (the two curves), 7 nm and 22 nm, and sweep (on the x-axis) the process node of the sensor's logic layer from 16 nm to 65 nm.

The overall energy saving is more sensitive to the processor node of the logic layer when the SoC uses a 7 nm node over that of a 22 nm node. This reason is simple: when the SoC uses a 22 nm node, the off-sensor work tends to dominates the total energy, leaving less room for optimization.

ROI Reuse. Instead of predicting an ROI for each frame, one can also reuse a previously ROI. We implement a ROI-reuse version of NPU-ROI, where a previously predicted ROI is reused over a number of subsequent frames quantified by the *reuse window*. Tbl. I shows how the gaze error and energy savings of the ROI-reused version of NPU-ROI over itself without ROI reuse change with the reuse window.

Overall, reusing previous ROIs leads to a significant accuracy drop with negligible energy savings. For instance, when we reuse an ROI for the next consecutive 16 frames, there is only a 0.03% energy saving but an 0.75° error increase. Worse, the standard deviation of the angular error increases with the reuse window, showing a decrease in robustness. The reason the energy gain is small from ROI reuse is because the prediction network's energy consumption is insignificant (1.04% of the total in-sensor energy).

VII. Related Work

Computational Image Sensors. Image sensors are increasingly integrating computation capabilities. The computation is conventionally done in the digital domain, such as Sony IMX 500 CIS [47], which integrates DNN accelerator inside the sensor. Recent proposals move computation into the analog domain to, e.g., computes logarithmic gradients [132], extract HOG features [86], eventification [50], [80], random sampling [96], ROI-based readout [37], and even analog DNNs [39], [127]. BLISSCAM performs eventification, random sampling, and ROI-based readout inside the sensor with little area overhead by reusing existing in-sensor hardware.

RedEye [81] is proposed as an analog ConvNet image sensor architecture. BLISSCAM exhibits three major differences. First, most importantly, the two works differ in high-level system design strategies: RedEye splits pre-trained DNNs to execute early DNN layers inside the sensor. It does not consider co-design or joint training of the algorithm with the hardware and is highly constrained in the type and size of the DNN layers it can accommodate. In contrast, BLISSCAM co-designs the in-sensor operations (sparse sampling) with the off-sensor downstream DNN, and therefore works more flexibly with diverse downstream vision algorithms and network architectures. Second, they have different hardware implementations: RedEye assumes a conventional analog image sensor and implements the DNN layers in the analog domain whereas BLISSCAM adopts a stacked DPS, where the DNN executes in the digital layer, naturally mitigating the noise issue in analog processing. Finally, BLISSCAM also proposes in-sensor sparse sampling and readout, which are unconcerned with in RedEye.

BLISSCAM not only augments the sensor hardware but jointly designs the in-sensor work (sampling) with off-sensor algorithm (segmentation) to ensure high task accuracy. Prior work explored such co-designs. LeCA [85] jointly trains an in-sensor encoder with downstream tasks; Bong et al. construct image sensor-processor systems for eye tracking [25] and face recognition [24], respectively; their eye tracking system only achieves 30 FPS and consumes 4.3 nJ per pixel, which is more than 10 × higher than that of BLISSCAM.

Random Sampling Image Sensors. The concept of random sampling in image sensors has been explored in compressive sensing and HDR imaging applications. In compressive sensing, random numbers are spatially assigned to the pixel array. However, the random number generator is either off-chip in the optical domain [43] or on-chip but resides beside the pixel array for row-wise processing [96]

or requires complicated in-pixel logic [88]. Thus, none meets the requirement for compact DPS in BLISSCAM. In HDR imaging, random numbers are temporally assigned to each pixel's sub-exposure slots. However, the random numbers are generated with coarse granularity (in pixel tiles) [130] or complicated in-pixel logic [90]. More crucially, such coded exposure scheme destroys the original pixel value, making it unsuitable for our design.

In contrast, BLISSCAM exploits existent in-pixel SRAMs with simple logic to realize fine-granular random sampling with a compact design, and buffer the necessary pixel values for eventification. The sampling method implemented in our sensor is customized to eye tracking, but prior work [84] has shown that other computer vision tasks such as classification can also benefit from sparse sampling. While our sampling network will have to adapt to different tasks, the actual hardware support for random sampling (Fig. 11) readily applies.

Eye Tracking Acceleration. Researchers explored dedicated accelerators to accelerate eye tracking. Bong et al. [25] and Hong et al. [59] design accelerators for in-sensor gaze estimation. They target non-DNN algorithms with much inferior accuracy as compared to the state of art DNN-based algorithms. I-flatcam [135] and EyeCoD [131] design accelerators for an eye tracking algorithm targeting lensless cameras while leaving the sensor front-end un-optimized. BLISSCAM shows that reducing sensor readout and sensor-host communication leads to significant overall energy reduction.

Prior studies have explored lightweight eye segmentation algorithms [49], [87], [131], [135]. For instance, EdGaze [49] predicts the ROI of an image before segmentation. Previous work has also explored ROI-based machine vision systems [50], [74], [87]. Built on top of the ROI prediction idea, we show that pseudo-random sampling within the ROI can further reduce energy consumption with minimal hardware support. We also co-design a ViT-based segmentation DNN to be robust against sparse inputs, as the accuracy of previous eye segmentation algorithms tends to drop under sparse inputs.

Event Cameras. Readers familiar with event cameras [52] might recognize that our eventification algorithm (Eqn. 1) is an emulation of an event camera. Indeed, our idea is inspired by event cameras — with one crucial difference: classic event cameras normalize pixel difference with respect to the previous value. We simplify the design to remove the normalization operation, which only complicates the sensor hardware design without providing noticeable accuracy benefits for eye tracking as we empirically find.

While there are generic object/ROI detection algorithms in event cameras [94], [102], our lightweight ROI detection can be seen as a specialized algorithm tailored for eye tracking, based on the observation that background pixel values in eye tracking do not change much over frames, so a very lightweight frame differencing would reveal ROI.

VIII. CONCLUSION

Sparse in-sensor sampling is critical to reducing the energy consumption and end-to-end latency of eye tracking, a crucial component in emerging domains such as AR/VR. Sampling within an image sensor dramatically reduces the amount of data that has to go through the energy-intensive image readout chain and sensor-host communication interface, two major power contributors to image sensors. The host, as a result, also operates on far fewer pixels, further reducing the computation cost. To support in-sensor sparse sampling with little hardware augmentation, BLISSCAM reuses the existing pixel-level analog readout circuitry to support eventification, random sampling, and sparse readout; and BLISSCAM uses a small in-sensor NPU to support ROI prediction. BLISSCAM reduces pixel volume by about 95% and thus achieves an $8.2\times$ energy reduction and a $1.4\times$ tracking latency reduction with little tracking accuracy degradation, compared to existing eye tracking systems.

IX. ACKNOWLEDGEMENT

We thank anonymous reviewers from ISCA 2024 for their valuable comments. The work is partially supported by NSF Awards #2328856, #2416375, #1942900.

REFERENCES

[1] "1/1.395" Color CMOS 200 Megapixel (16384x12288) Image Sensor with PureCel®Plus-S Technology," https://www.ovt.com/products/ovb0a/.
[2] "Battery capacity, battery life and charging time of the oculus quest 2." [Online]. Available: https://www.sir-apfelot.de/en/battery-capacity-battery-life-and-charging-time-of-the-oculus-quest-2-38530/
[3] "Canon High Speed Global Shutter," https://asia.canon/en/campaign/cmos-image-sensors/high-speed-global-shutter.
[4] "Does vr use a lot of energy?" [Online]. Available: https://blog.constellation.com/2022/02/22/vr-power-consumption/
[5] "Eyelink 1000 plus." [Online]. Available: https://www.sr-research.com/eyelink-1000-plus/
[6] "HTC VIVE PRO EYE: Next-generation virtual reality," https://www.tobii.com/products/integration/xr-headsets/device-integrations/htc-vive-pro-eye.
[7] "Introducing smart eye pro 10: Streamlined set up with enhanced user-friendly experience." [Online]. Available: https://smarteye.se/news/introducing-smart-eye-pro-10-streamlined-set-up-with-enhanced-user-friendly-experience/
[8] "Jetson modules." [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules
[9] "Meta quest 3 release date, price, and specs." [Online]. Available: https://www.pcgamesn.com/meta/quest-3-release-date-price-specs
[10] "Micron 178-Ball, Single-Channel Mobile LPDDR3 SDRAM Features." [Online]. Available: https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr3/178b_8-16gb_2c0f_mobile_lpddr3.pdf
[11] "Micron System Power Calculators." [Online]. Available: https://www.micron.com/support/tools-and-utilities/power-calc
[12] "Most advanced eye tracking system — tobii pro spectrum." [Online]. Available: https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-spectrum
[13] "MST Family of Products," https://www.baesystems.com/en/product/mst-sensors.
[14] "Nvidia reveals xavier soc details." [Online]. Available: https://www.forbes.com/sites/moorinsights/2018/08/24/nvidia-reveals-xavier-soc-details/amp/
[15] "OMNIVISION Launches World's First Image Sensor for Automotive Viewing Cameras with 140 dB HDR and Top LED Flicker Mitigation Performance," https://www.ovt.com/press-releases/omnivision-launches-worlds-first-image-sensor-for-automotive-viewing-cameras-with-140-db-hdr-and-top-led-flicker-mitigation-performance/.
[16] "Real Examples of MIPI Interfaces in AR/VR, Mobile, and Other Image Applications." [Online]. Available: https://2384176.fs1.hubspotusercontent-na1.net/hubfs/2384176/DevCon-2022/2022-MIPI-DevCon-Real-Examples-of-MIPI-IF-in-ARVR-Mobile-.pdf

[17] "Sony to Release World's First Intelligent Vision Sensors with AI Processing Functionality," https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/.

[18] "Specifications for Eye Tracker 5." [Online]. Available: https://help.tobii.com/hc/en-us/articles/360012483818-Specifications-for-Eye-Tracker-5

[19] "The Impact of Higher Data Rate Requirements on MIPI CSI and MIPI DSI Designs." [Online]. Available: https://www.rambus.com/wp-content/uploads/2020/09/MIPI_DEVCON_Presentation_1_01.pdf

[20] I. B. Adhanom, P. MacNeilage, and E. Folmer, "Eye tracking in virtual reality: A broad review of applications and challenges," *Virtual Reality*, pp. 1–25, 2023.

[21] A. N. Angelopoulos, J. N. Martel, A. P. Kohli, J. Conradt, and G. Wetzstein, "Event based, near eye gaze tracking beyond 10,000 hz," *arXiv preprint arXiv:2004.03577*, 2020.

[22] M. D. Binder, N. Hirokawa, and U. Windhorst, Eds., *Saccade, Saccadic Eye Movement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 3557–3557.

[23] K. Bong, S. Choi, C. Kim, D. Han, and H.-J. Yoo, "A low-power convolutional neural network face recognition processor and a cis integrated with always-on face detector," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 1, pp. 115–123, 2017.

[24] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "14.6 a 0.62 mw ultra-low-power convolutional-neural-network face-recognition processor and a cis integrated with always-on haar-like face detector," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 248–249.

[25] K. Bong, I. Hong, G. Kim, and H.-J. Yoo, "A 0.5 error 10 mw cmos image sensor-based gaze estimation processor," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1032–1040, 2016.

[26] M. Borys and M. Plechawska-Wójcik, "Eye-tracking metrics in perception and visual attention research," *EJMT*, vol. 3, pp. 11–23, 2017.

[27] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 036–11 045.

[28] T. T. Brunyé, T. Drew, D. L. Weaver, and J. G. Elmore, "A review of eye tracking for understanding and improving diagnostic interpretation," *Cognitive research: principles and implications*, vol. 4, no. 1, pp. 1–16, 2019.

[29] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[30] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[31] P. Chakravarthula, Z. Zhang, O. Tursun, P. Didyk, Q. Sun, and H. Fuchs, "Gaze-contingent retinal speckle suppression for perceptually-matched foveated holographic displays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 11, p. 4194–4203, 2021.

[32] S. Chandra, G. Sharma, S. Malhotra, D. Jha, and A. P. Mittal, "Eye tracking based human computer interaction: Applications and their uses," in *2015 International Conference on Man and Machine Interfacing (MAMI)*. IEEE, 2015, pp. 1–5.

[33] Y. Chang, C. He, Y. Zhao, T. Lu, and N. Gu, "A high-frame-rate eye-tracking framework for mobile devices," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 1445–1449.

[34] A. K. Chaudhary, R. Kothari, M. Acharya, S. Dangi, N. Nair, R. Bailey, C. Kanan, G. Diaz, and J. B. Pelz, "Ritnet: Real-time semantic segmentation of the eye for gaze tracking," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3698–3702.

[35] J. Chen and Q. Ji, "3d gaze estimation with a single camera without ir illumination," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[36] Y. Chen, T. Li, X. Chen, Z. Cai, and T. Su, "High-frequency systolic array-based transformer accelerator on field programmable gate arrays," *Electronics*, vol. 12, no. 4, p. 822, 2023.

[37] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon, "A spatial-temporal multiresolution cmos image sensor with adaptive frame rates for tracking the moving objects in region-of-interest and suppressing motion blur," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2978–2989, 2007.

[38] J. Choi, S. Park, J. Cho, and E. Yoon, "An energy/illumination-adaptive cmos image sensor with reconfigurable modes of operations," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 6, pp. 1438–1450, 2015.

[39] J. Choi, S. Lee, Y. Son, and S. Y. Kim, "Design of an always-on image sensor using an analog lightweight convolutional neural network," *Sensors*, vol. 20, no. 11, p. 3101, 2020.

[40] J. Chossat, "Image sensors in 3d stacking technology: Retrospective and perspectives from a digital architect's point of view," in *2023 International Image Sensor Workshop*, 2023.

[41] V. Clay, P. König, and S. Koenig, "Eye tracking in virtual reality," *Journal of Eye Movement Research*, vol. 12, no. 1, 2019.

[42] S. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *2017 26th international conference on computer communication and networks (ICCCN)*. IEEE, 2017, pp. 1–7.

[43] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[44] B. Duinkharjav, K. Chen, A. Tyagi, J. He, Y. Zhu, and Q. Sun, "Color-perception-guided display power reduction for virtual reality," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–16, 2022.

[45] K. Egiazarian, A. Foi, and V. Katkovnik, "Compressed sensing image reconstruction via recursive spatially adaptive filtering," in *2007 IEEE International Conference on Image Processing*, vol. 1. IEEE, 2007, pp. I–549.

[46] A. Einstein, "Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen," *Annalen der physik*, vol. 4, 1905.

[47] R. Eki, S. Yamada, H. Ozawa, H. Kai, K. Okuike, H. Gowtham, H. Nakanishi, E. Almog, Y. Livne, G. Yuval *et al.*, "9.6 a 1/2.3 inch 12.3 mpixel with on-chip 4.97 tops/w cnn processor back-illuminated stacked cmos image sensor," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 154–156.

[48] J. E. Farrell, P. B. Catrysse, and B. A. Wandell, "Digital camera simulation," *Applied optics*, vol. 51, no. 4, pp. A80–A90, 2012.

[49] Y. Feng, N. Goulding-Hotta, A. Khan, H. Reyserhove, and Y. Zhu, "Real-time gaze tracking with event-driven eye segmentation," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022, pp. 399–408.

[50] Y. Feng, T. Ma, A. Boloor, Y. Zhu, and X. Zhang, "Learned in-sensor visual computing: From compression to eventification," in *International Conference on Computer-Aided Design*, 2023.

[51] Y. Feng, P. Whatmough, and Y. Zhu, "Asv: Accelerated stereo vision system," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 643–656.

[52] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[53] S. J. Garbin, Y. Shen, I. Schuetz, R. Cavin, G. Hughes, and S. S. Talathi, "Openeds: Open eye dataset," *arXiv preprint arXiv:1905.03702*, 2019.

[54] S. Golomb, "Run-length encodings (corresp.)," *IEEE transactions on information theory*, vol. 12, no. 3, pp. 399–401, 1966.

[55] C. W. Group, "Mipi white paper: An introductory guide to mipi automotive serdes solutions (mass)," 2021. [Online]. Available: https://www.mipi.org/introductory-guide-to-mass

[56] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[57] C. Hennessey, B. Noureddin, and P. Lawrence, "A single camera eye-gaze tracking system with free head motion," in *Proceedings of the 2006 symposium on Eye tracking research & applications*, 2006, pp. 87–94.

[58] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up sram state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2008.

[59] I. Hong, K. Bong, D. Shin, S. Park, K. J. Lee, Y. Kim, and H.-J. Yoo, "A 2.71 nj/pixel gaze-activated object recognition system for low-power mobile smart glasses," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 45–55, 2015.

[60] I. T. Hooge, D. C. Niehorster, R. S. Hessels, J. S. Benjamins, and M. Nyström, "How robust are wearable eye trackers to slow and fast head and body movements?" *Behavior Research Methods*, pp. 1–15, 2022.

[61] X. Hou, J. Liu, X. Tang, C. Li, J. Chen, L. Liang, K.-T. Cheng, and M. Guo, "Architecting efficient multi-modal aiot systems," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–13.

[62] T.-H. Hsu, G.-C. Chen, Y.-R. Chen, R.-S. Liu, C.-C. Lo, K.-T. Tang, M.-F. Chang, and C.-C. Hsieh, "A 0.8 v intelligent vision sensor with tiny convolutional neural network and programmable weights using mixed-mode processing-in-sensor technique for image classification," *IEEE Journal of Solid-State Circuits*, 2023.

[63] Z. Hu, "Gaze analysis and prediction in virtual reality," in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 2020, pp. 543–544.

[64] R. Ikeno, K. Mori, M. Uno, K. Miyauchi, T. Isozaki, H. Abe, M. Nagamatsu, I. Takayanagi, J. Nakamura, S.-G. Wuu *et al.*, "Evolution of a 4.6 μm, 512$\times$ 512, ultra-low power stacked digital pixel sensor for performance and power efficiency improvement," in *Proceedings of the 2023 International Image Sensor Workshop, Scotland, UK*, 2023, pp. 21–25.

[65] R. Ikeno, K. Mori, M. Uno, K. Miyauchi, T. Isozaki, I. Takayanagi, J. Nakamura, S.-G. Wuu, L. Bainbridge, A. Berkovich *et al.*, "A 4.6-μm, 127-db dynamic range, ultra-low power stacked digital pixel sensor with overlapped triple quantization," *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2943–2950, 2022.

[66] Y. Imaoka, A. Flury, and E. D. De Bruin, "Assessing saccadic eye movements with head-mounted display virtual reality technology," *Frontiers in Psychiatry*, vol. 11, p. 572938, 2020.

[67] J. Intoy and M. Rucci, "Finely tuned eye movements enhance visual acuity," *Nature communications*, vol. 11, no. 1, p. 795, 2020.

[68] Y. Itoh and G. Klinker, "Interaction-free calibration for optical see-through head-mounted displays based on 3d eye localization," in *2014 IEEE symposium on 3d user interfaces (3dui)*. IEEE, 2014, pp. 75–82.

[69] R. J. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," in *The mind's eye*. Elsevier, 2003, pp. 573–605.

[70] H. Jarodzka, I. Skuballa, and H. Gruber, "Eye-tracking in educational practice: Investigating visual perception underlying teaching and learning in the classroom," *Educational Psychology Review*, vol. 33, no. 1, pp. 1–10, 2021.

[71] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*. Springer, 2016, pp. 349–364.

[72] H. Kim, J. Yang, J. Lee, S. Yoon, Y. Kim, M. Choi, J. Yang, E.-S. Ryu, and W. Park, "Eye tracking-based 360 vr foveated/tiled video rendering," in *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE Computer Society, 2018, pp. 1–1.

[73] S.-H. Kim, G.-S. Lee, H.-J. Yang *et al.*, "Eye semantic segmentation with a lightweight model," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3694–3697.

[74] V. Kodukula, A. Shearer, V. Nguyen, S. Lingutla, Y. Liu, and R. LiKamWa, "Rhythmic pixel regions: multi-resolution visual sensing system towards high-precision visual computing at low power," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 573–586.

[75] R. S. Kothari, A. K. Chaudhary, R. J. Bailey, J. B. Pelz, and G. J. Diaz, "Ellseg: An ellipse segmentation framework for robust gaze tracking," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2757–2767, 2021.

[76] V. Kravets and A. Stern, "Progressive compressive sensing of large images with multiscale deep learning reconstruction," *Scientific reports*, vol. 12, no. 1, p. 7228, 2022.

[77] M. Kwon, S. Lim, H. Lee, I.-S. Ha, M.-Y. Kim, I.-J. Seo, S. Lee, Y. Choi, K. Kim, H. Lee *et al.*, "A low-power 65/14nm stacked cmos image sensor," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–4.

[78] Y. Leng, C.-C. Chen, Q. Sun, J. Huang, and Y. Zhu, "Energy-efficient video processing for virtual reality," in *Proceedings of the 46th International Symposium on Computer Architecture*, 2019, pp. 91–103.

[79] B. Li, H. Fu, D. Wen, and W. Lo, "Etracker: A mobile gaze-tracking system with near-eye display based on a combined gaze-tracking algorithm," *Sensors*, vol. 18, no. 5, p. 1626, 2018.

[80] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$128 120db 15μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[81] R. LiKamWa, Y. Hou, J. Gao, M. Polansky, and L. Zhong, "Redeye: analog convnet image sensor architecture for continuous mobile vision," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 255–266, 2016.

[82] Z. Lit, M. Sun, A. Lu, H. Ma, G. Yuan, Y. Xie, H. Tang, Y. Li, M. Leeser, Z. Wang *et al.*, "Auto-vit-acc: An fpga-aware automatic acceleration framework for vision transformer with mixed-scheme quantization," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2022, pp. 109–116.

[83] C. Liu, S. Chen, T.-H. Tsai, B. De Salvo, and J. Gomez, "Augmented reality-the next frontier of image sensors and compute systems," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 426–428.

[84] Y. Liu, C. Matsoukas, F. Strand, H. Azizpour, and K. Smith, "Patchdropout: Economizing vision transformers using patch dropout," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3953–3962.

[85] T. Ma, A. J. Boloor, X. Yang, W. Cao, P. Williams, N. Sun, A. Chakrabarti, and X. Zhang, "Leca: In-sensor learned compressive acquisition for efficient machine vision on the edge," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–14.

[86] T. Ma, W. Cao, F. Qiao, A. Chakrabarti, and X. Zhang, "Hogeye: neural approximation of hog feature extraction in rram-based 3d-stacked image sensors," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2022, pp. 1–6.

[87] T. Ma, Y. Feng, X. Zhang, and Y. Zhu, "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*, 2023, pp. 1–14.

[88] V. Majidzadeh, L. Jacques, A. Schmid, P. Vandergheynst, and Y. Leblebici, "A (256$\times$ 256) pixel 76.7 mw cmos imager/compressor based on real-time in-pixel compressive sensing," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2010, pp. 2956–2959.

[89] K. Mania, A. McNamara, and A. Polychronakis, "Gaze-aware displays and interaction," in *ACM SIGGRAPH 2021 Courses*, 2021, pp. 1–67.

[90] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, "Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 7, pp. 1642–1653, 2020.

[91] P. Mathur, T. Mittal, and D. Manocha, "Dynamic graph modeling of simultaneous eeg and eye-tracking data for reading task identification," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 1250–1254.

[92] A. Mayberry, P. Hu, B. Marlin, C. Salthouse, and D. Ganesan, "ishadow: design of a wearable, real-time mobile gaze tracker," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 82–94.

[93] L. Millet, S. Chevobbe, C. Andriamisaina, L. Benaissa, E. Deschaseaux, E. Beigne, K. B. Chehida, M. Lepecq, M. Darouich, F. Guellec *et al.*, "A 5500-frames/s 85-gops/w 3-d stacked bsi vision chip based on parallel in-focal-plane acquisition and processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1096–1105, 2019.

[94] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[95] J. Ohta, *Smart CMOS image sensors and applications*. CRC press, 2020.

[96] Y. Oike and A. El Gamal, "Cmos image sensor with per-column $\sigma\delta$ adc and programmable compressed sensing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 318–328, 2012.

[97] I. Park, W. Jo, C. Park, B. Park, J. Cheon, and Y. Chae, "A 640$\times$640 fully dynamic cmos image sensor for always-on operation," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 898–907, 2019.

[98] K. Park, S. Yeom, and S. Y. Kim, "Ultra-low power cmos image sensor with two-step logical shift algorithm-based correlated double sampling scheme," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3718–3727, 2020.

[99] W. Park, C. Piao, H. Lee, and J. Choi, "Cmos image sensor with two-step single-slope adcs and a detachable super capacitive dac," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 849–853, 2021.

[100] A. Patney, J. Kim, M. Salvi, A. Kaplanyan, C. Wyman, N. Benty, A. Lefohn, and D. Luebke, "Perceptually-based foveated virtual reality," in *ACM SIGGRAPH 2016 Emerging Technologies*, 2016, pp. 1–2.

[101] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn, "Towards foveated rendering for gaze-tracked virtual reality," *ACM Trans. Graph.*, vol. 35, no. 6, Nov. 2016. [Online]. Available: https://doi.org/10.1145/2980179.2980246

[102] E. Perot, P. De Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 639–16 652, 2020.

[103] A. Plopski, J. Orlosky, Y. Itoh, C. Nitschke, K. Kiyokawa, and G. Klinker, "Automated spatial calibration of hmd systems with unconstrained eye-cameras," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2016, pp. 94–99.

[104] R.-M. Rahal and S. Fiedler, "Understanding cognitive and affective mechanisms in social psychology through eye-tracking," *Journal of Experimental Social Psychology*, vol. 85, p. 103842, 2019.

[105] T. Roth, M. Weier, A. Hinkenjann, Y. Li, and P. Slusallek, "A quality-centered analysis of eye tracking data in foveated rendering," *Journal of eye movement research*, vol. 10, no. 5, 2017.

[106] A. Rowlands, *Physics of digital photography*. IOP Publishing, 2017.

[107] M. Rucci, R. Iovin, M. Poletti, and F. Santini, "Miniature eye movements enhance fine spatial detail," *Nature*, vol. 447, no. 7146, pp. 852–855, 2007.

[108] S. Sarangi and B. Baas, "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, 2021, pp. 1–5.

[109] J. Scott-Thomas, "Trends and developments in state-of-the-art cmos image sensors," in *2023 International Image Sensor Workshop*, 2023.

[110] K. Semmelmann and S. Weigelt, "Online webcam-based eye tracking in cognitive science: A first look," *Behavior Research Methods*, vol. 50, pp. 451–465, 2018.

[111] M.-W. Seo, M. Chu, H.-Y. Jung, S. Kim, J. Song, D. Bae, S. Lee, J. Lee, S.-Y. Kim, J. Lee *et al.*, "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*, vol. 57, no. 4, pp. 1125–1137, 2022.

[112] J. Sharda, W. Li, Q. Wu, S. Chang, and S. Yu, "Temporal frame filtering for autonomous driving using 3d-stacked global shutter cis with iwo buffer memory and near-pixel compute," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[113] L. Sidenmark, M. Parent, C.-H. Wu, J. Chan, M. Glueck, D. Wigdor, T. Grossman, and M. Giordano, "Weighted pointer: Error-aware gaze-based interaction through fallback modalities," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 11, pp. 3585–3595, 2022.

[114] R. Singh, S. Bailey, P. Chang, A. Olyaei, M. Hekmat, and R. Winoto, "34.2 a 21pj/frame/pixel imager and 34pj/frame/pixel image processor for a low-vision augmented-reality smart contact lens," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 482–484.

[115] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, 2017.

[116] T. Strandvall, "Eye tracking in human-computer interaction and usability research," in *IFIP Conference on Human-Computer Interaction*. Springer, 2009, pp. 936–937.

[117] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[118] Y. Su, J. Holleman, and B. Otis, "A 1.6 pj/bit 96% stable chip-id generating circuit using process variations," in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. IEEE, 2007, pp. 406–611.

[119] H. Tsugawa, H. Takahashi, R. Nakamura, T. Umebayashi, T. Ogita, H. Okano, K. Iwase, H. Kawashima, T. Yamasaki, D. Yoneyama *et al.*, "Pixel/dram/logic 3-layer stacked cmos image sensor technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 3–2.

[120] V. Van der Leest, E. Van der Sluis, G.-J. Schrijen, P. Tuyls, and H. Handschuh, "Efficient implementation of true random number generator based on sram pufs," in *Cryptography and Security: From Theory to Applications: Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*. Springer, 2012, pp. 300–318.

[121] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[122] T. Wang, L. Gong, C. Wang, Y. Yang, Y. Gao, X. Zhou, and H. Chen, "Via: A novel vision-transformer accelerator based on fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4088–4099, 2022.

[123] Y. Wang, S. Lu, and D. Harter, "Multi-sensor eye-tracking systems and tools for capturing student attention and understanding engagement in learning: A review," *IEEE Sensors Journal*, vol. 21, no. 20, pp. 22 402–22 413, 2021.

[124] E. Whitmire, L. Trutoiu, R. Cavin, D. Perek, B. Scally, J. Phillips, and S. Patel, "Eyecontact: scleral coil eye tracking for virtual reality," in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, 2016, pp. 184–191.

[125] M. Wieckowski, D. Sylvester, D. Blaauw, V. Chandra, S. Idgunji, C. Pietrzyk, and R. Aitken, "A black box method for stability analysis of arbitrary sram cell structures," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 2010, pp. 795–800.

[126] S.-G. Wuu, H.-L. Chen, H.-C. Chien, P. Enquist, R. M. Guidash, and J. McCarten, "A review of 3-dimensional wafer level stacked backside illuminated cmos image sensor process technologies," *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2766–2778, 2022.

[127] H. Xu, N. Lin, L. Luo, Q. Wei, R. Wang, C. Zhuo, X. Yin, F. Qiao, and H. Yang, "Senputing: An ultra-low-power always-on vision perception chip featuring the deep fusion of sensing and computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 1, pp. 232–243, 2021.

[128] W. Ye, X. Zhou, J. T. Zhou, C. Chen, and K. Li, "Accelerating attention mechanism on fpgas based on efficient reconfigurable systolic array," *ACM Transactions on Embedded Computing Systems (TECS)*, 2022.

[129] Y.-H. Yiu, M. Aboulatta, T. Raiser, L. Ophey, V. L. Flanagin, P. Zu Eulenburg, and S.-A. Ahmadi, "Deepvog: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning," *Journal of neuroscience methods*, vol. 324, p. 108307, 2019.

[130] M. Yoshida, T. Sonoda, H. Nagahara, K. Endo, Y. Sugiyama, and R.-i. Taniguchi, "High-speed imaging using cmos image sensor with quasi pixel-wise exposure," *IEEE Transactions on Computational Imaging*, vol. 6, pp. 463–476, 2019.

[131] H. You, C. Wan, Y. Zhao, Z. Yu, Y. Fu, J. Yuan, S. Wu, S. Zhang, Y. Zhang, C. Li *et al.*, "Eyecod: eye tracking system acceleration via flatcam-based algorithm & accelerator co-design," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 610–622.

[132] C. Young, A. Omid-Zohoor, P. Lajevardi, and B. Murmann, "A data-compressive 1.5/2.75-bit log-gradient qvga image sensor with multi-scale readout for always-on object detection," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 2932–2946, 2019.

[133] X. Zhang, Y. Sugano, and A. Bulling, "Evaluation of appearance-based methods and implications for gaze-based applications," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.

[134] X. Zhang, X. Liu, S.-M. Yuan, and S.-F. Lin, "Eye tracking based control system for natural human-computer interaction," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[135] Y. Zhao, Z. Li, Y. Fu, Y. Zhang, C. Li, C. Wan, H. You, S. Wu, X. Ouyang, V. Boominathan *et al.*, "I-flatcam: A 253 fps, 91.49 μj/frame ultra-compact intelligent lensless camera for real-time and efficient eye tracking in vr/ar," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2022, pp. 108–109.

[136] Z. Zhu and Q. Ji, "Eye gaze tracking under natural head movements,"

in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 918–923.