TokenSense: Tokenized Sensing for Efficient Visual Processing

Yiwen Liang, Weidong Cao
Department of ECE, The George Washington University, Washington, D.C., USA
{yiwen.liang, weidong.cao}@gwu.edu

Abstract

Intelligent vision processing powers various AI applications, yet remains constrained by the physical separation and structural decoupling between front-end sensing and back-end neural processing. In this pipeline paradigm, traditional vision sensors indiscriminately acquire and transmit massive amounts of raw visual data, creating severe bottlenecks in acquisition, communication, and computation along the processing pipeline. This work introduces tokenized sensing (TokenSense), a co-optimization framework that seamlessly bridges visual sensing and neural processing for end-to-end efficiency in continuous vision. By enabling the sensor itself as an active and intelligent filter, our framework intelligently identifies and captures only task-critical tokens at its absolute origin, fundamentally alleviating the data movement burden across the entire pipeline. Architecturally, the module is implemented in the physical domain on standard stacked image sensors, reusing existing hardware and adding a lightweight processing layer with minimal area cost. Evaluated across benchmarks, tokenized sensing reduces sensor data by 50% while preserving task accuracy, yielding a 1.9× saving in sensing energy and a 1.8× end-to-end speedup. In future work, we plan to refine and expand the framework to achieve higher end-to-end efficiency.

1 Introduction

Intelligent vision processing is the cornerstone of many AI applications, ranging from augmented/virtual reality (AR/VR) and autonomous driving to healthcare and embodied intelligence [5, 11]. Yet, current vision systems remain inherently inefficient and slow. At the heart of this challenge is the rigid, passive integration of front-end sensing and back-end processing. Today's sensors act as unintelligent data generators, indiscriminately acquiring and transmitting massive volumes of raw data. This creates severe data acquisition, movement, and computation bottlenecks, as AI processors must consume unfiltered data for specific tasks, bearing the full computational burden of a pipeline that is both power-hungry and slow [2, 13]

Most current efforts focus on improving the back end of the visual processing pipeline through model optimization or specialized hardware accelerators [6, 9], while the front end that creates the vast frames consumed by AI models remains largely overlooked. Recently, in-sensor processing has emerged to tackle the front-end data deluge by integrating early layers of AI models into sensors, thereby only transmitting compact features instead of raw pixels [4, 21]. Yet, such methods are often ad-hoc and unable to guarantee efficiency: in many cases, intermediate feature maps of AI models are larger than the raw images, negating any bandwidth savings [17, 24]. As such, the transformative potential of optimizing the vision sensor itself as a **first-class computing element**, to reduce the burden on front-end acquisition and back-end processing, and ultimately to amplify overall system efficiency and performance, remains largely untapped. *How can we jointly design visual sensing and AI processing to achieve holistic optimization and potentially enable generic acceleration of continuous vision applications?*

This work introduces tokenized sensing (TokenSense), an algorithm hardware co-design framework that transforms the image sensor from a passive data generator into an active, intelligent filter. Recognizing that not all pixels contribute equally to downstream tasks and that large portions of a scene remain unchanged across frames, TokenSense suppresses redundant visual tokens at the point of capture and transmits only task-relevant information, enabling more efficient end-to-end vision processing. Experiments show that our framework reduces 50% of sensor data while preserving the accuracy, yielding a 1.9× in sensor energy saving, and 1.8× end-to-end speedup, which directly benefits from sensing. Future work will address residual redundancy among the retained tokens by compacting similar tokens prior to transmission, enabling a more efficient representation and further reducing sensing bandwidth and back-end computational overhead.

2 Background

Fig. 1 illustrates today's intelligent vision processing pipeline, where the front-end image sensor continuously captures high-fidelity frames and streams massive raw pixel data to the back-end host processor for further processing by vision models [23]. Vision models typically process individual frames, while video workloads process sequences of frames within each clip. The system's power and performance are therefore primarily determined by two key components: the image sensor and AI processing.

Conventional image sensors often operate as passive data acquisition devices. After exposure, the pixel array converts incident photons into analog signals, which are digitized by readout circuitry and ADCs before transmission via high-speed interfaces (e.g., MIPI CSI-2). However, this sensing

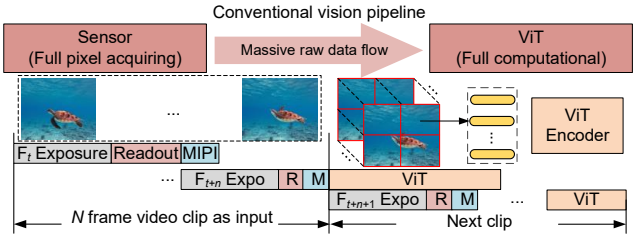

Figure 1. Conventional intelligent vision pipeline.

pipeline is energy-intensive: power-hungry ADCs and output buffers account for up to 69% of total sensor power [20], while data transmission incurs additional overhead approaching 100 pJ/Byte [19]. To reduce sensing energy, recent works embed early network layers of a backbone model directly into the image sensor to extract task-relevant features, thereby reducing data movement [18]. However, such in-sensor processing is often tightly coupled to specific backbone networks, sacrificing flexibility, and often fails to guarantee bandwidth efficiency: in many cases, the intermediate feature maps are larger than the raw images. On the backend, specialized hardware accelerators are used to speed up vision models (mainly Vision Transformers) [7]. To improve efficiency, extensive efforts have focused on AI-side optimizations such as token pruning, merging, and model quantization [3, 12]. However, these models still require full-resolution pixel input to determine which information can be discarded. While effective at reducing computation within the accelerator, these approaches only act after sensing, leaving the front-end data-acquisition cost unchanged.

In practice, visual data contain substantial redundancy, and not all information contributes equally to downstream tasks. Exploiting these redundancies opens new opportunities for system-level innovation, where the sensor itself can actively act as an intelligent filter, transmitting only the features most relevant to the downstream task. We argue that this pruning of vision data at the origin of the visual pipeline, without integrating early layers of backbone models, provides the most effective and generic leverage for end-to-end optimization. Motivated by this insight, we propose tokenized sensing, which achieves this goal through co-designing the algorithm and sensor architecture.

3 Tokenized Sensing (TokenSense) Framework

Tokenized sensing (TokenSense) is an algorithm–hardware co-design framework to enable end-to-end efficiency and high performance for vision processing. At the algorithm level, TokenSense identifies and suppresses redundant visual information directly at the sensor front-end, reducing unnecessary data acquisition, movement, and downstream computation. At the hardware level, the proposed operations are mapped onto existing image sensor circuitry with

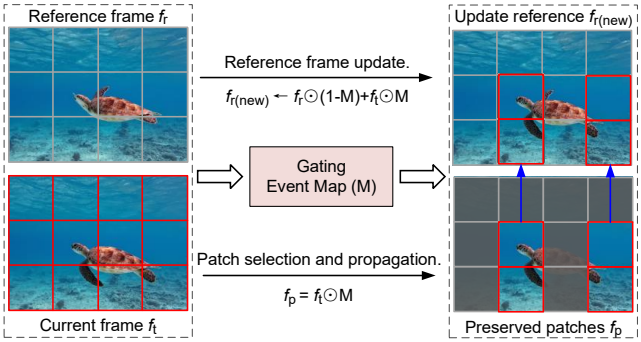

Figure 2. TokenSense algorithm design.

minimal modification, enabling efficient reuse of standard readout circuits in the physical domain.

3.1 Algorithm Design

The proposed TokenSense algorithm draws inspiration from the human vision system that allocates perceptual resources efficiently by focusing on the regions containing motion or fine details, while providing only coarse updates elsewhere [27]. By emulating this selective attention mechanism, for each frame, TokenSense *identifies regions (patches) with significant variation and selectively preserves these patches, allowing the backend model to operate only on these informative regions.* To achieve this, we design a variation-aware gating mechanism that dynamically measures differences between consecutive frames and suppresses static or redundant areas. Formally, at each time step t, TokenSense compares the current frame f_t, partitioned into N non-overlapping patches $p_{i(i=1,2,\ldots,N)}$, against a stored reference frame, f_r, from the previous update. It then updates its internal state and determines the output according to the following procedure (Fig. 2).

Gating. For each patch, the gating module computes the pixel-wise difference between the current and reference frames and aggregates it into a patch-level statistic:

$$E_m(p_i) = \Phi\Big(\sum_{(x,y)\in p_i} \mathbb{I}\big(|F_t(x,y) - F_r(x,y)| > \sigma\big)\Big), \quad (1)$$

where $F_t(x,y)$ and $F_r(x,y)$ denote the pixel intensities at coordinate (x,y) within the current and reference patches, respectively. $\mathbb{I}(\cdot)$ is an indicator function that outputs 1 if the pixel-wise difference exceeds the threshold σ, and 0 otherwise. The activation function $\Phi(\cdot)$ then maps the difference in pixels in patch p_i to a binary decision map $E_m(p_i) \in \{0,1\}$, indicating whether there exhibits significant changes. The threshold σ controls the sensitivity of the gate and can be either manually tuned during inference or learned jointly during training. In a learned configuration, we introduce a soft constraint to regulate the expected drop ratio: $\mathcal{L}_{\text{gate}} = \alpha(G(E)-\mu)^2$, where $G(E)$ denotes the percentage of dropped patches within an input frame, μ is the target drop ratio, and α balances regularization with downstream task loss. This auxiliary objective allows the gating to adaptively trade off

between accuracy and preserving. Finally, the binary outputs for all patches are combined into a global event map $\mathbf{M} \in \{0,1\}^N$, representing the spatial activation pattern for a frame that determines which regions are retained.

Patch selection and propagation. Given the event map, the gate extracts the preserved patches by applying the binary mask to the current frame, $f_p = f_t \odot \mathbf{M}$, where f_p denotes the preserved regions. Conceptually, this operation corresponds to a gather along the patch dimension, transmitting only the selected tokens downstream and discarding non-event patches. The proportion of active patches naturally varies over time, enabling adaptive sensing that scales with scene dynamics.

Reference frame update. To preserve the consistency and completeness, the reference frame, f_r, is updated after each gating step as: $f_r \leftarrow f_r \odot (1-\mathbf{M}) + f_t \odot \mathbf{M}$. This operation functions as a scatter-update on the buffered tensor without reloading the entire frame. At the first time step, all patches are preserved to initialize the reference state.

Training procedure. We insert the TokenSense before a pre-trained model and fine-tune the system to learn the optimal gating threshold. During training, the backbone is operated directly on the preserved tokens subset. We employ standard knowledge distillation [14] to maintain accuracy. The gating loss coefficient α is selected to balance the target drop ratio and task accuracy.

3.2 Hardware Support

To support TokenSense in hardware, we reconfigured the analog readout circuit, thereby extending the conventional Digital Pixel Sensor (DPS) architecture to support both traditional imaging and the proposed TokenSense operation. We augment the SS ADC with several switch transistors and a compact logic unit (highlighted blue in Fig. 3). The time-multiplexing reuse allows the same readout circuit to alternate between normal quantization and TokenSense operations. The associated behaviors are discussed below.

Gating. During gating, a reference frame f_r is maintained to compute the absolute difference with the incoming frame f_t, which is then compared against bipolar thresholds $\pm\sigma$ to determine activation. To achieve this, during the exposure of f_t, the comparator operates as an analog buffer by closing the HOLD switch to establish a negative feedback path (Fig. 3 ①), and f_r is stored on the capacitor C_{az1} for subsequent subtraction. When the HOLD switch opens, the comparator transitions into differencing mode. The charge associated with f_t is redistributed onto C_{az1}, producing a differential signal proportional to $(f_t - f_r)$, as depicted in Fig. 3 ②. The comparator then performs thresholding by connecting its second input capacitor C_{az2} to the threshold voltage σ. To evaluate both positive and negative changes, two comparisons are performed sequentially using $V_{\text{th1}} = +\sigma$ and $V_{\text{th2}} = -\sigma$. The comparator output thus directly provides the pixel-level binary indicator of whether the corresponding

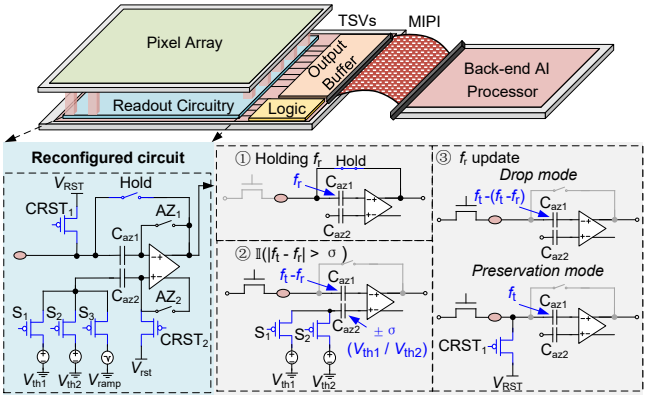

Figure 3. TokenSense hardware design.

patch should be retained. These operations are simultaneously applied to all pixels, given the global shutter exposure of the DPS architecture.

Patch selection and propagation. Following the gating, a small augmented logic reads the generated binary events and aggregates them into patch-level maps to determine whether each patch should be preserved or discarded based on a binary flag, "Drop" (1 for drop; otherwise, preserve). This flag also controls ADCs to enable selective quantization. For each preserved patch, the logic also produces a four-element tuple, (x_1, x_2, y_1, y_2), which specifies the xy-coordinates of two opposing corners of the patch. Only pixels in preserved patches are quantized and stored in the output buffer for further processing.

Reference frame update. The reference update operates in two modes (Fig. 3 ③). In the *Drop mode*, C_{az1} is recharged with f_t, which effectively performs an inverse operation that restores the capacitor voltage to $f_t - (f_t - f_r) = f_r$, thereby retaining the previous reference. In the *Preservation mode*, C_{az1} is first reset by V_{RST} and then charged with f_t, updating the buffer with the new reference value.

4 Experiments and Results

4.1 Experimental Setup and Baselines

The hardware system (Fig. 3) comprises a customized image sensor and an off-chip conventional accelerator (GPU). The sensor adopts a stacked architecture. Consistent with current industry practices [15], the top pixel layer uses a standard 65 nm CMOS process, while the bottom analog/logic layer is implemented in a 22 nm process. The top-layer adopts a standard 4-T pixel design, and sensor energy and latency are estimated using circuit-level analytical models from prior work covering readout, ADC, and MIPI transfer [22]. The bottom-layer digital logic for TokenSense computing is implemented in RTL and synthesized using a Synopsys/Cadence EDA flow. We compare the in-sensor efficiency of our design against a conventional CMOS image sensor (CNV-CIS). Backend (ViT) performance is measured on a GPU, and inference efficiency is compared against a baseline that processes full-resolution pixel inputs.

We assess our algorithm on three standard video benchmarks: Kinetics-400 [16], Something-Something V2 (SSv2) [10], and UCF-101 [26]. We adopt a ViT-based architecture, a widely used backbone across prior video understanding works [1]. The input video clip contains 16 frames, each center-cropped to a resolution of 224×224 and partitioned into non-overlapping 16×16 patches following the default format used by ViT. The performance is reported using the clip Top-1 accuracy. To quantitatively assess the gating policy, we compare our strategy against three baselines. Uniform dropping: Drops patches at fixed spatial locations across all frames, maintaining a uniform spatial sampling pattern. Random dropping: Randomly selects and discards patches within each frame, resulting in stochastic sparsity. Top-r dropping: Computes the L_2 norm between each patch and its corresponding preserved reference, preserving the top-r patches with the largest L_2 norm per frame [8].

4.2 System Evaluation Results

Accuracy. Fig. 4 compares the accuracy degradation of different dropping policies under patch drop ratios (PDR) from 10% to 80% under UCF-101. Our policy consistently outperforms all baselines, showing a stronger ability to preserve task-relevant information while eliminating redundancy. Among the baselines, Top-r dropping performs reasonably well at moderate PDR, but its per-patch L_2 norm computation is hardware-unfriendly, especially for analog domains. In contrast, our gating strategy is lightweight, and continues to excel Top-r at high PDR by learning a global threshold that selectively retains only the most informative patches. We identify an effective trade-off point at around a 50% PDR (i.e., "sweet spot"), where redundant data is substantially reduced with minimal loss of accuracy. Knowledge distillation plays a crucial role in stabilizing performance; removing it results in an additional ~3% accuracy degradation. Based on these results, we further sweep the PDR around 50% on the larger and more complex K-400 and SSv2 benchmarks (the subset of Fig. 4). The results confirm that a 50% PDR consistently yields minimal accuracy loss (within 4%) even under diverse and challenging video scenarios.

Sensor Efficiency. *Energy:* Comparing to the CNV-CIS which consumes 218.2 pJ/pixel, under the "sweet spot", TokenSense achieves a 1.9× energy reduction at the sensor level. Although exposure energy remains unchanged, the dominant ADC and MIPI transmission, historically the most power-hungry components in image sensors, is sharply reduced, while the in-sensor computation overhead remains lightweight. A higher patch drop rate (PDR) can further improve energy savings. *Latency:* We further evaluate sensing latency under a 120 FPS requirement and compare it against a CNV-CIS. For transmission, we assume a 4-lane MIPI CSI-2 interface operating at 1.2 Gbps per lane, consistent with Sony's IMX500 specifications [25]. In modern image sensors,

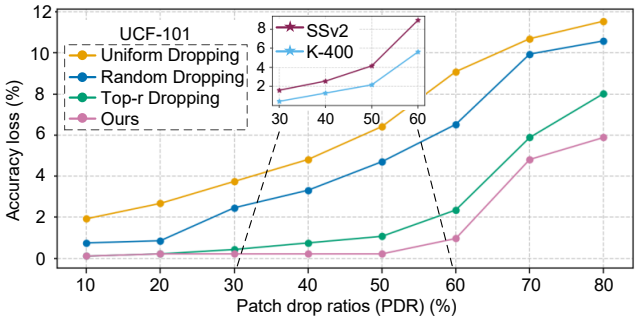

Figure 4. TokenSense algorithm evaluation.

total per-frame latency is dominated by exposure time (8.3 ms/frame), whereas analog readout and digital transmission are orders of magnitude faster. TokenSense introduces only approximately 5 μs per frame due to comparator settling during absolute-difference thresholding. Combined with reduced transmission volume, TokenSense achieves up to a 1.2× single-frame latency improvement. *Area:* The additional logic required for TokenSense is minimal, only seven extra switching transistors plus a small amount of control logic. In total, each bottom-tier readout includes 3 233-fF auto-zero capacitors, one comparator, 13 switches, 10 6T-SRAM cells, and a compact digital block (a 2-bit comparator and 21 logic gates), all implemented in 22-nm technology, resulting in 25 μm^2 bottom-tier readout cell per pixel.

Backend and End-to-End Performance. To quantify the impact of token reduction on back-end efficiency, we evaluate memory footprint and inference speed. Our method demonstrates clear efficiency gains: at a 50% PDR, memory usage reduces to 0.65× and inference time to 0.52× of the baseline. In the pipeline situation, MIPI transmission latency is hidden within the exposure interval; thus, TokenSense introduce 5 μs per frame of additional latency. However, the back-end inference latency reduction at 50% PDR, ~140 ms per clip, far exceeds this sensing overhead. As a result, joint front-end sparsification and backend acceleration enable a 1.8× end-to-end speedup without accuracy degradation.

Residual redundancy may still exist among the retained tokens. Our future work will focus on compressing redundant tokens or merging similar tokens prior to transmission to enable a more efficient representation and further reduce bandwidth and backend computation.

5 Conclusion

This work introduces tokenized sensing, a co-design framework that seamlessly bridges image sensing and neural processing for efficient end-to-end continuous vision. Tokenized sensing leverages both algorithm and hardware innovations to reduce the data acquisition, movement, and computing across the pipeline and support flexible deployment across diverse vision platforms. Future work will further extend this framework to achieve even greater efficiency across the vision pipeline.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6836–6846.

[2] Yongmin Baek, Byungjoon Bae, Hyojin Shin, Charana Sonnadara, Haein Cho, Ching-Yi Lin, Yujia Mu, Cong Shen, Sahil Shah, Gunuk Wang, et al. 2025. Edge intelligence through in-sensor and near-sensor computing for the artificial intelligence of things. *npj Unconventional Computing* 2, 1 (2025), 25.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461* (2022).

[4] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)*. IEEE, 253–257.

[5] Anis Davoudi, Kumar Rohit Malhotra, Benjamin Shickel, Scott Siegel, Seth Williams, Matthew Ruppert, Emel Bihorac, Tezcan Ozrazgat-Baslanti, Patrick J Tighe, Azra Bihorac, et al. 2019. Intelligent ICU for autonomous patient monitoring using pervasive sensing and deep learning. *Scientific reports* 9, 1 (2019), 8020.

[6] Pudi Dhilleswararao, Srinivas Boppu, M Sabarimalai Manikandan, and Linga Reddy Cenkeramaddi. 2022. Efficient hardware architectures for accelerating deep neural networks: Survey. *IEEE access* 10 (2022), 131788–131828.

[7] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[8] Matthew Dutson, Yin Li, and Mohit Gupta. 2023. Eventful transformers: Leveraging temporal redundancy in vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 16911–16923.

[9] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. 2019. Computer vision algorithms and hardware implementations: A survey. *Integration* 69 (2019), 309–320.

[10] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*. 5842–5850.

[11] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of field robotics* 37, 3 (2020), 362–386.

[12] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[13] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 2016. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 123–136.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[15] Rimon Ikeno, Kazuya Mori, Masayuki Uno, Ken Miyauchi, Toshiyuki Isozaki, Isao Takayanagi, Junichi Nakamura, Shou-Gwo Wuu, Lyle Bainbridge, Andrew Berkovich, et al. 2022. A 4.6-μm, 127-dB dynamic range, ultra-low power stacked digital pixel sensor with overlapped triple quantization. *IEEE Transactions on Electron Devices* 69, 6 (2022), 2943–2950.

[16] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

[17] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. 2020. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review* 53, 8 (2020), 5455–5516.

[18] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. 2016. Redeye: analog convnet image sensor architecture for continuous mobile vision. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 255–266.

[19] Chiao Liu, Song Chen, Tsung-Hsun Tsai, Barbara De Salvo, and Jorge Gomez. 2022. Augmented reality-the next frontier of image sensors and compute systems. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 426–428.

[20] Tianrui Ma, Adith Jagadish Boloor, Xiangxing Yang, Weidong Cao, Patrick Williams, Nan Sun, Ayan Chakrabarti, and Xuan Zhang. 2023. Leca: In-sensor learned compressive acquisition for efficient machine vision on the edge. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 1–14.

[21] Tianrui Ma, Weidong Cao, Fei Qiao, Ayan Chakrabarti, and Xuan Zhang. 2022. 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*. 1–6.

[22] Tianrui Ma, Zhe Gao, Zhe Chen, Ramakrishna Kakarala, Charles Shan, Weidong Cao, and Xuan Zhang. 2025. Systematic Methodology of Modeling and Design Space Exploration for CMOS Image Sensors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[23] John Scott-Thomas. 2023. Trends and developments in state-of-the-art cmos image sensors. In *2023 International Image Sensor Workshop*.

[24] Zhuang Shao, Xiaoliang Chen, Li Du, Lei Chen, Yuan Du, Wei Zhuang, Huadong Wei, Chenjia Xie, and Zhongfeng Wang. 2021. Memory-efficient CNN accelerator based on interlayer feature map compression. *IEEE Transactions on Circuits and Systems I: Regular Papers* 69, 2 (2021), 668–681.

[25] Sony. [n. d.]. Sony IMX500. https://developer.sony.com/imx500.

[26] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[27] Colin Ware. 2010. *Visual thinking for design*. Elsevier.