INVITED PAPER

Software-Defined Imaging: A Survey

By Suren Jayasuriya, *Member IEEE*, Odrika Iqbal, Venkatesh Kodukula, Victor Torres, Robert LiKamWa, and Andreas Spanias, *Fellow IEEE*

ABSTRACT | Huge advancements have been made over the years in terms of modern image-sensing hardware and visual computing algorithms (e.g., computer vision, image processing, and computational photography). However, to this day, there still exists a current gap between the hardware and software design in an imaging system, which silos one research domain from another. Bridging this gap is the key to unlocking new visual computing capabilities for end applications in commercial photography, industrial inspection, and robotics. In this survey, we explore existing works in the literature that can be leveraged to replace conventional hardware components in an imaging system with software for enhanced reconfigurability. As a result, the user can program the image sensor in a way best suited to the end application. We refer to this as software-defined imaging (SDI), where image sensor behavior can be altered by the system software depending on the user's needs. The scope of our survey covers imaging systems for single-image capture, multi-image, and burst photography, as well as video. We review works related to the sensor primitives, image signal processor (ISP) pipeline, computer architecture, and operating system elements of the SDI stack. Finally, we outline the infrastructure and resources for SDI systems, and we also discuss possible future research directions for the field.

KEYWORDS | Codesign; computational photography; computer vision; programmability; software-defined imaging (SDI).

I. INTRODUCTION

Image sensing has become ubiquitous in modern society, ranging from industrial uses in the workplace all the way to personal entertainment through the sharing of photos and videos via social media. Driven by the development of CMOS image sensors in the 1990s and 2000s, image sensing has become cheap, affordable, and, when integrated with smartphones, extremely portable and easy to use. Indeed, billions of photos are taken and uploaded on the internet each day.

In accordance with the development of image sensor technology, the rise of image processing, computer vision, and computational photography has been similarly meteoric. Advances in algorithms, including state-of-the-art machine learning using deep neural networks, have enabled high-fidelity visual computing. However, while both image-sensing hardware and the software that implements and realizes the algorithms have grown exponentially, there is still relatively little codesign between the two domains. This is due to a number of factors: not only technical challenges of interfacing analog sensing components with digital computation but also social and cultural challenges of different communities of researchers and industries communicating with one another across the stack.

For example, the traditional image sensor faces major hurdles to allow for flexibility and reconfiguration in its operating modes. A recent study showed that changing image sensor resolution costs hundreds of milliseconds in latency [1], and this was mostly due to software operating system (OS) bottlenecks. Most image sensors have been slow to expose knobs to the software developer, such as resolution, exposure, and quantization; in contrast, most

Manuscript received 29 September 2021; revised 7 February 2023; accepted 4 April 2023. This work was supported in part by Qualcomm, in part by NSF under Grant CCF-1909663, and in part by the Sensor Signal and Information Processing (SenSIP) Center, Arizona State University. *(Corresponding author: Odrika Iqbal.)*

Suren Jayasuriya and **Robert LiKamWa** are with the School of Arts, Media and Engineering and the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA.

Odrika Iqbal, **Venkatesh Kodukula**, **Victor Torres**, and **Andreas Spanias** are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA (e-mail: oiqbal@asu.edu).

Digital Object Identifier 10.1109/JPROC.2023.3266736

imaging and vision algorithms do not exploit these sensing mechanisms efficiently and adaptively at runtime. This leads to lost opportunities for improved reconfigurability of imaging systems in practice.

To address these issues, an interdisciplinary community of researchers and practitioners has embraced **software-defined imaging** (SDI) to improve the technology of visual computing systems. A software-defined image sensor offers several dimensions of sensing configurability along with system support and programming abstractions to support application-specific needs. Hardware parameters, such as exposure, resolution, and frame rate, are programmable in image sensors, and software-defined image sensors exploit this programmability alongside software algorithms to optimize certain imaging task metrics, such as energy, latency, and task performance. This field has a vertical-oriented mindset connecting knowledge of sensor physics and electronics, analog and mixed-signal circuits, digital systems and architectures, OSs and programming languages, and end applications of computer vision and computational photography.

In this article, we survey this emerging area of SDI, and highlight key works across the hardware and software stack in the literature. We point out how this area intersects other popular areas of research including embedded computer vision, deep learning, computational photography, and digital hardware acceleration for imaging applications. However, we also point out the unique nature of SDI research, cross-cutting traditional boundaries to allow synergy among the stack.

In addition, we begin to discuss the growing issue of how to quantify performance and evaluate quality in this cross-disciplinary area. We survey existing metrics, tools, and benchmarks that are open source and available to the public. These tools can be used to help improve basic research, generate training datasets for machine learning applications, and model/analyze system efficiency in terms of latency and energy for the potential deployment of solutions. We highlight how this public infrastructure is distributed across the software–hardware stack from application programming interfaces (APIs) to image signal processors (ISPs) to hardware devices.

In the rest of this article, we dive deeply into the literature in this field. In Section III, we introduce research on programmable image sensors and their primary sensing mechanisms, as well as the associated algorithms that exploit these capabilities. In Section IV, we discuss how raw data from the image sensor are converted to visual images suitable for either human viewing or machine vision. Section V discusses the intersection of deep learning and SDI, which, in turn, motivates the various hardware and software research to accelerate imaging applications in Section VI. This includes works on hardware acceleration for imaging tasks/pipelines, OS/runtime configurations to enable flexibility, and languages and compilers to support programming these sensors. Finally, in Section VII, we outline tools and datasets available for SDI to enable faster research prototyping in this area. Section VIII wraps up our outlook on the field and provides a look at the future challenges and opportunities present in SDI.

II. WHY SOFTWARE-DEFINED IMAGING?

One of the major endeavors of this article is to coin the phrase "SDI," which we denote in this article by the acronym SDI, as an umbrella term and concept, which connects various strands of existing research. We do not seek to be overly prescriptive or limiting in our definition of SDI, but rather to coalesce around a common research agenda while having the flexibility to allow for offshoots and branches of interesting research potential. To help draw motivation for SDI, we actually first look at the research parallels in the communications community and how this enabled successful applications and high impact on those fields.

Software-defined radio (SDR) is now a mature field within communications [2]. It was originally coined in the 1980s, but a bulk of the research came in the 1990s and 2000s along with the growth of the telecom industry. SDR centers around the idea of a radio communication system where traditional hardware components, such as filters, mixers, filters, amplifiers, modulators/demodulators, and detectors, are instead augmented or replaced with software and embedded processing. This allows radio systems to be flexible and reconfigurable and to maximize their usage of the spectrum at beneficial tradeoff points for system design. SDR has spurred a wide variety of work [3], [4], [5], [6] and has been implemented in many modern communication standards [7], [8].

Similarly, our idea of SDI centers around the idea of the traditional hardware components and mechanisms of image sensors being replaced or augmented by software alternatives. The main reason for doing so is flexibility: allowing sensors to adaptively change their sensing behavior for new environments and application demands. For SDI, we focus on key sensor primitives, such as region-of-interest readout, exposure, frame rate, and quantization, since these are all programmable and implemented onboard the image sensor with dedicated hardware. SDI research aims to design the hardware–software stack from the sensor, ISP, compute architecture, and OS to allow for reconfigurability and programmability for these primitives, driven by the end applications. By doing so, we can actually achieve favorable tradeoffs between system efficiency and application-specific task performance for imaging and computer vision.

III. PROGRAMMABLE SENSOR MECHANISMS

Since the invention of CCD and CMOS image sensors in the late 20th century, there has been an increased emphasis on programmability in their design to enable user flexibility during capture. In this section, we will curate and discuss research works that highlight the development

of programmable camera sensors. In particular, we will delve into detailed discussions on frame rate, exposure, region of interest (ROI), and quantization, which are the most widely configured parameters in the image sensor. In addition to describing the hardware advances for these sensor mechanisms, we will also examine algorithms that exploit programmable sensor knobs for end-applications in imaging and machine vision.

We note that we analyze imaging systems for a single capture (e.g., conventional photography), multiframe or burst photography (e.g., high dynamic range (HDR) mode and panorama stitching), and video (e.g., continuous shooting mode). We do not differentiate between these modes throughout this article as modern systems typically can perform capture in all of these modalities. However, we do highlight when research has been designed and targeted for a specific application in mind.

A. ROI

The term ROI refers to the set of pixels within an image frame, which describes the target object. Typically, ROIs are defined in shape as rectangles on the image sensor due to ease of readout. However, a general ROI could be considered as any contiguous set of pixels that can be read from an image sensor. For a rectangular ROI, if the top-left corner pixel location of the object of interest is defined by $\{x, y\}$, and the width and height of the object are given by $\{w, h\}$, then the ROI is characterized by the vector $\{x, y, w, h\}$.

Before we start discussing the existing literature on ROI, let us first understand the principle of operation of a traditional CMOS image sensor. CMOS image sensors are built with a multitude of photodiodes with a small amplifier associated with every pixel and columnwise circuitry enabling further amplification and analog-to-digital conversion [9]. Photons impinging upon the photodiodes are converted to charge carriers once they arrive at the photoactive silicon region as per the photoelectric effect. The resulting accumulated charge is read out utilizing the amplifier (also called source follower) present at every pixel. The pixel array is arranged spatially in a row–column topology, and each row is read out at a time with every column in the current row undergoing parallel readout. In order to accommodate ROI readout (with sensor pixels outside of the ROI switched off), additional circuitry is required in general. An example is the use of skip logic [10], where the entire pixel array is divided into smaller clusters of 32 $\times$ 32 pixels. These are configured based on an external serial bit stream dictating which pixels to read out in the vertical and horizontal frame directions. The skipping pattern is scanned across the skip blocks constituting the skip logic and flip-flop shift registers where the status of the flip-flop determines whether the scanning signal is to be fed to the current shift register or the shift register of the succeeding block. In this way, only the ROI is readout while discarding the irrelevant frame information (see Fig. 1).

1) ROI Hardware: For conventional sensors, there has been ongoing research in both hardware and software programmability for ROIs. Some of the initial research proposed arbitrary ROI functionality [11], [12]. In particular, Dierickx et al. [11] looked at asynchronous spatiotemporal variable pixel readout. A true randomly addressable CMOS image sensor was proposed, wherein pixel readout could be varied both in the spatial domain and the temporal domain, i.e., any pixel located at any part of the image frame could be addressed and read out at any instance of time, depending on user needs [11]. In a similar vein, a 2048 $\times$ 2048 active pixel sensor (APS) was designed and fabricated with a nonintegrating direct pixel readout architecture featuring logarithmic light-to-voltage conversion [12]. This enabled random pixel accessibility spatially and temporally.

Sensors have also utilized adaptive pixel resolution to solve classical challenges with varying light. In normal image sensors, low-light conditions commonly result in underexposed and noisy images. In [13], an image sensor design was presented that featured on-chip pixel resolution variability for light-adaptive imaging. The proposed sensor design was shown to enhance the signal-to-noise ratio by summing the signals from neighboring pixels, thus mitigating the effects of noise in low-light images. The similar imaging technology was patented in 2000 that enabled spatial resolution variability depending on the light level [14]. These early works paved the way for developing power-efficient ROI-capable camera sensors.

Beyond just adaptive resolution, researchers also realized that ROIs can be leveraged for energy harvesting within the CMOS sensor. The primary vehicle for realizing an energy-harvesting scheme was the logarithmic response pixel circuit. It differed from run-of-the-mill image sensors in which their voltage changes as a logarithmic response of the amount of light hitting upon their pixels, as opposed to a linear response. In 2011, a CMOS image sensor design was presented with reconfigurable resolution capabilities [15], which exploited the logarithmic response pixel architecture for the purposes of energy harvesting for wireless imaging. In the proposed design, each pixel photodiode can be toggled to enter into either photosensing mode or energy-harvesting mode.

Finally, ROIs have been utilized to implement advanced sensing schemes, such as spatially variant resolution single-pixel imaging and compressed sensing. In lieu of using millions of pixels, single-pixel cameras multiplex incoming light measurements onto a single or few pixels using a spatial light modulator [16], [17]. The framework of compressed sensing allows single-pixel cameras to take far fewer measurements of the scene compared to conventional image sensors, and sparse optimization or machine learning techniques reconstruct the full scene in postprocessing [18]. Relevant to this article, the development in ROI technology has enabled on-chip single-pixel imaging without spatial light modulators [19], [20]. In [21], the problem of conventional cameras incurring

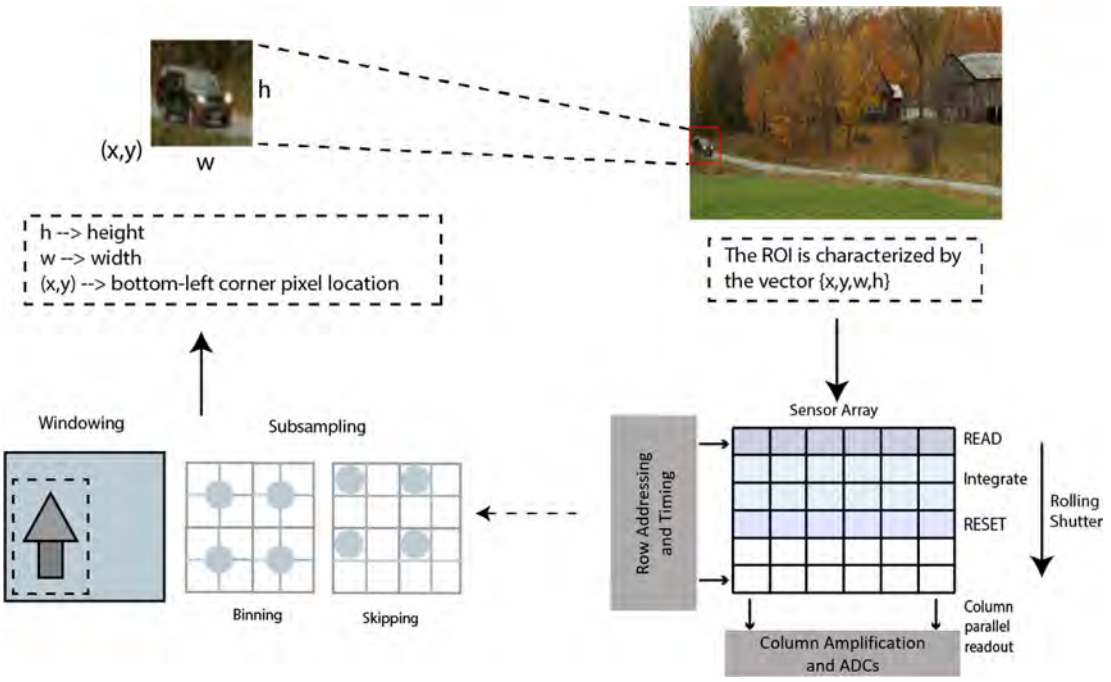


Fig. 1. ***Sensor pixels are switched off via binning or skipping strategies as per the detected ROI. This assists in improving the latency and energy efficiency by reducing the number of pixels to be read out. It also helps reduce the computation overhead associated with the ISP pipeline as fewer pixel operations will be required for fewer pixels.***

high processing costs in invisible wavelengths is addressed. The proposed technique enables stable image recovery by trading off the resolution of the non-ROI parts of the image, allowing image reconstruction with sub-Nyquist measurements. In a similar vein, a generalized assorted pixel (GAP) camera sensor design was proposed by Yasuma et al. [22], which allows controlling the trade-off between the spatial resolution, spectral detail, and dynamic range of an image postcapture. The complex array of color filters results in undersampled image channels—a similar problem to Shin et al. [21]—but the camera is accompanied by a novel algorithm that enables stable image recovery by suppressing aliasing effects.

2) ROI Selection Algorithms: Since the development of the ROI, many ROI-based applications have generated a lot of interest from the early 2000s. These applications include ROI-based object tracking and detection for computer vision and foveated imaging. Integration of these algorithms with ROI camera sensors implies tremendous power savings for both the camera readout and the downstream image processing due to data reduction and latency improvements. We highlight several ROI selection algorithms and how they opened up the avenue for energy-efficient solutions to various imaging applications.

One of the main tasks for any ROI algorithm is determining the salient regions of the image to be extracted in the ROI. In 2000, Hsung and Lun [23] advanced the ongoing research on ROI extraction algorithms by discovering that a wavelet-based ROI technique could be further improved in terms of computational complexity by replacing it with a simple interpolation method. It was also shown how the wavelet theory can be leveraged to devise a good sampling scheme. In 2002, Lin et al. [24] proposed an ROI detection algorithm that took into account a different variant of the traditional ROI camera sensor—one capable of reading out irregular ROI shapes. The algorithm showed promising results in the context of low-light, complex imagery. Furthermore, the algorithm fared well in instances where there had been connected objects in an image and in cases where an object had split into fragments.

Early ROI selection algorithms also utilized the Kalman filter, which adapts its internal matrices and state vector according to external measurements [25], [26], [27], [28]. In 2015, Zhang et al. [29] showed how to use frequency and space domain features for localizing regions of interest. The proposed algorithm is particularly effective for high-resolution remote-sensing images. With growing interest in neural network solutions, several deep learning-based object detection/tracking methods have been developed which offer state-of-the-art ROI-ing capability [30], [31], [32], [33].

For energy-efficient ROI algorithms, recent research has introduced the concept of adaptive subsampling where image sensors can powergate or turn off pixels outside of the ROI during readout to save power. Adaptive subsampling defines an ROI that localizes the part of the frame the user is interested in, and the pixels outside of the ROI are deemed redundant and discarded. In 2019, Mohan et al. [34] developed an algorithm that computed the heat map of incoming image frames and used Otsu's threshold to hone in on the object of interest. Along

the same lines, recent work leveraged a YOLO detector and a Kalman filter coupled together for predictive ROI tracking [35] to facilitate adaptive subsampling, the end goal being energy optimization via preemptive pixel inactivation outside of the ROI. Extending on this work, Iqbal et al. [36] showed that an efficient convolution operator (ECO) network coupled with the Kalman filter led to superior performance when accelerated on an FGPA. The work demonstrated how adaptive image sampling has tremendous potential in reducing computational complexity and processing speed.

ROI selection and extraction have been used widely since the early 2000s. A 2001 study conducted by Pietka et al. [37] showed an ROI extraction method for assistng radiologists in bone age estimation. A region-based segmentation algorithm was proposed by Blasco et al. [38] in 2007 for the industrial problem of detecting peel defects in citruses. This method exploited the fact that, in ROI-based algorithms, the contrast between different regions is of greater significance than the color of each pixel to help detect defects.

Closely tied to the notion of an ROI is foveated imaging, an image processing technique wherein pixel resolution varies spatially in a captured image frame. The spatial variation is guided by one or more parts of the image featuring the highest resolution, also known as fixation points. These fixation points correspond to the fovea (center of the eye's retina). A 2015 work leveraged foveated imaging for selecting visually interpretable elements from image frames using a network architecture called BubbleNet [39]. This work is geared toward applications such as visual mining and discovery, and could potentially be coupled with ROI-capable sensors for a combined hardware–software solution. Foveated imaging is also making a mark in the field of augmented reality (AR) and virtual reality (VR). AR/VR applications involve the integration of computer-generated artificial reality in the real world (AR) or the generation of a completely self-contained artificial reality (VR). Therefore, realistic and plausible image rendering is of paramount importance for these applications, as well as computational and energy efficiency to ensure satisfactory real-time performance. Notable foveated rendering techniques have been proposed in the past decade, which sacrifices resolution in the peripheries to optimize the system efficiency [40], [41], [42], [43], [44], [45]. Certain works also focus on foveated rendering based on eye tracking [46], [47].

B. Exposure

Exposure refers to the period of time during which the image sensor is allowed to be exposed to light. It is also known as shutter speed since it quantifies the length of time during which the shutter remains open during image capture. The chosen exposure setting should provide an optimal combination of the aperture dimension through which the sensor gets exposed to light, the shutter speed, and the ISO sensitivity (which determines how photosensitive the medium is) [48].

Exposure control has been of interest to researchers since the 1990s. Several HDR image sensors with exposure control have been proposed over the years [49], [50]. Different exposure strategies are adopted for serving different purposes. A slower shutter speed or longer exposure duration can be adopted to create the effect of blur in a captured photograph of a fast-moving object, thus lending a sense of movement in the still picture. Longer exposure duration is also useful in low-light conditions when greater light exposure is needed to compensate for the weak light source. In contrast, a short exposure time can be adopted when there is bright light or when the effect of blur is not desired. Exposure types include rolling [51], global [52], coded [53], two bucket [54], and so on.

Among these, coded imaging techniques, such as compressed sensing [16], coded exposure [53], or coded apertures [55], multiplex image measurements, and computationally recover the image. These techniques come with a multitude of advantages. In particular, coded exposure enables image deblurring while retaining high-frequency spatial details (see Fig. 2). For instance, researchers have utilized the coded exposure-based deblurring technique to eliminate spatially variant blur in images [56], [57]. Coded exposure has also been used to break down the tradeoff between the spatial and temporal resolution of cameras. In a paper by Liu et al. [58], a coded exposure-enabled camera was used to capture coded images, and a video sequence was successfully reconstructed from an individual coded image. Coded exposure has also found application in robotic platforms [59]. Given the success of coded exposure-based image recovery, researchers started designing image sensors with on-chip coded exposure capabilities [60], [61], [62], [63], [64], [65].

High-speed photography and optical flow-based applications require controlled temporal sampling. Programmable rolling shutters capture and combine spatial and temporal information [66] by controlling the readout timing and exposure length of each pixel row. The proposed readout architecture provides greater flexibility in sampling in the time dimension. In 2016, a new type of display-camera communication system called DisCo was proposed based on a rolling shutter image sensor [67], which receives a temporally modulated light signal as input and transforms it into a spatial flicker pattern. With a global model of the camera motion trajectory, motion deblurring has also been achieved with a rolling shutter camera [68]. In 2012, a novel rolling shutter camera-based barcode design was presented, which enabled robust barcode scanning from relatively long distances [69].

Primal-dual coding [70] is another coding scheme that dictates which light paths are to affect the image. A primal-dual coding video camera was presented by O'Toole and Kutulakos [71], which assisted in visualizing various light transport phenomena. In 2018, a new imaging technique called coded two-bucket (C2B) was proposed, which

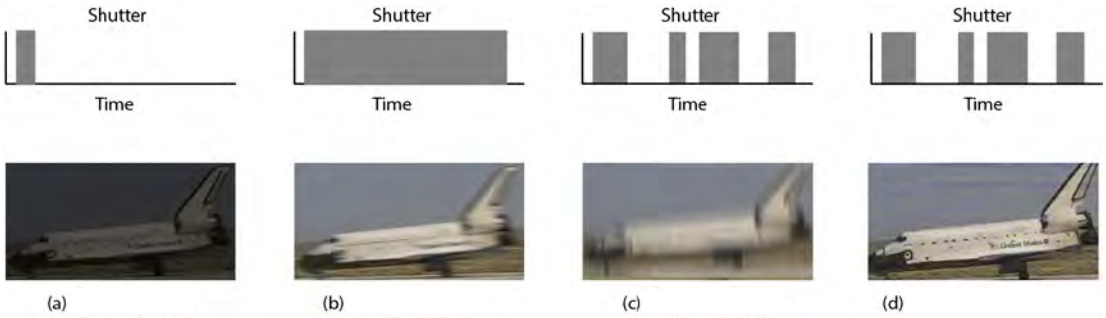


Fig. 2. *Conventional exposure-induced blurring effects are difficult to remove as high-frequency information is lost during deblurring via the deconvolution step. In contrast, software-defined coded exposure enables fairly accurate deblurring while preserving the high-frequency details. (a) Short exposure. (b) Longer exposure. (c) Coded exposure. (d) Coded exposure-based deblurring.*

features two buckets per pixel [54]. The incoming light signal is modulated at the pixel-level based on which bucket is triggered for integrating the signal.

C. Frame Rate/Readout

The frame rate is the temporal frequency at which image frames are processed/appear on a display (see Fig. 3). Hardware advances have been made over the years that allow frame rate configurability in image sensors. One line of research focuses on the fact that pixel sampling can accommodate frame rate variation in the same image sensor, improving energy efficiency and reducing off-chip image processing computations. Researchers have been interested in exploiting this since as far back as the 1990s. For instance, CMOS APSs were utilized for positioning signal processing circuitry on the imaging focal plane [72]. A cluster of digital shift registers could be configured by the user, and this rendered the resolution of the sensor to be user-dependent. The fully sampled frame rate was 30 Hz, and it went up for lower image resolutions. This work opened up the avenue for exploring image sensor designs, wherein the functionalities include pixel sampling-based frame rate programmability. In later years, Carnegie Mellon University published a work on their low-power, embedded computer vision platform [73]. The hardware platform includes a video first-in–first-out (FIFO) queue, which is crucial as it enables the camera to run at full speed and dissociates the pixel clock from the CPU clock. Since processing is dissociated from the pixel clock, the pixel clock does not need to be set to the slowest possible processing time. This enables higher frame rates. The work should be of particular interest for applications featuring surveillance/tracking of fast-moving objects. Other than this, a coded rolling shutter-based readout architecture was proposed by Gu et al. [66], which would allow the user to have greater flexibility in temporal domain image sampling. Works looking into scene information-based frame rate adaptation were also coming out. In 2010, a camera sensor was developed with sensor knobs that allowed spatial–temporal resolution adaptation postimage capture [74]. Scene information is considered while making decisions regarding resolution control.

In 2017, a work came out addressing the problems associated with continuous visual data processing on mobile devices [75]. The need for continuous information processing entails high computational complexity. To relieve mobile devices of such an intensive workload, the computational burden is offloaded to the cloud. A host of low-power sensors is placed as "gatekeepers" to the primary imager. These sensors continuously supervise the environmental stimuli. Only when incoming sensory data indicate an activity of interest for a running application, the primary imager is "woken up" to capture the image that is then transmitted to the cloud.

Other than this, sensor readout designs have also been proposed enabling both configurable SNR and configurable frame rate [76].

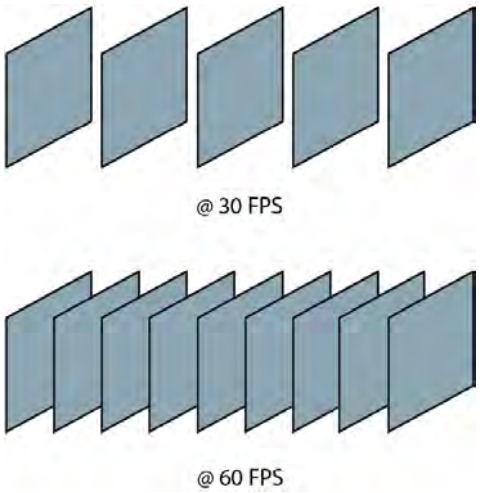


Fig. 3. *Software-defined frame rate control enables the user to select higher frame rates whenever the situation calls for it (when objects are moving very fast, for example). A higher frame rate basically means that we will be able to pack a greater number of frames within a 1-s duration. The figure visualizes that a greater number of frames are captured within the same duration at 60 frames/s than at 30 frames/s.*

Multiresolution solutions to motion-induced image blurring have also been investigated. Researchers discovered that implementing a multiresolution-capable sensor could facilitate frame rate programmability when required. Case in point—a spatial–temporal multiresolution image sensor design—had been proposed in 2007 [77]. Fast-moving objects filmed using low-frame-rate cameras may result in blurry images. High-speed imaging is required for capturing fast motion. However, it should be noted that high-frame-rate cameras come with their own share of problems such as large bandwidth requirements and high power consumption. In the proposed framework [77], a high frame rate is retained in the ROI localizing the fast-moving object, and a low frame rate is employed for the static background in the scene. The proposed CMOS sensor is designed to generate spatial–temporal multiple-resolution readouts utilizing two channels. One of the channels is reserved for low-frame-rate data at the highest possible resolution for static backgrounds, and the other channel is dedicated to high-frame-rate data at low spatial resolution for fast-moving objects in detected ROIs.

With the advances made in the deep learning community, algorithms are being developed exploiting programmable frame rates in image sensors. In particular, research in adaptive frame rate-based online object tracking has been gaining traction. On-the-go frame rate adaptation resulted in optimized energy savings without noticeable degradation in tracking precision in [78]. The optimum frame rate depends on motion speed, and this fact is leveraged in the study to improve energy savings via temporal sampling. Further work in the area optimized the frame rate by focusing on the energy consumption tradeoff between image capturing and tracking [79]. In the proposed method, scene information plays a role in picking the required parameters. Slowfast networks came out in 2019, which feature a slow pathway operating at a lower temporal resolution and a fast pathway operating at a higher temporal resolution [80]. The networks have shown promising results in video recognition tasks and have tremendous potential if integrated into SDI systems.

D. Quantization

Image quantization entails the lossy compression of visual data facilitating the relaxation of memory and bandwidth constraints (see Fig. 4). The image data are represented in terms of bits, and a range of values is represented by just one quantized state. Work in image quantization algorithm development has been gaining momentum since the 1980s. In 1982, a landmark work was published in the field of image quantization [81]. The idea behind this study was to procure high-fidelity color images with small frame buffers. Color statistics were acquired from the first frame, and a colormap was chosen. Original pixels were then mapped to the nearest neighbor in the chosen colormap. In the following decade, quantization works featuring machine learning started making their mark. A clever idea for image quantization was presented, wherein a genetic algorithm was used in conjunction with the c-means clustering algorithm [82]. The method was particularly attractive as it was shown to converge to a global optimum. In the meantime, researchers were also invested in discovering ways to complement and fortify existing image quantization techniques. In [83], a peer group filtering (PGF) algorithm was proposed for image denoising. In this work, each pixel value in a group was substituted with the weighted peer representative average. Local statistics received after running PGF are utilized as weights in the quantization step to tone down colors in detailed parts of the frame. In 2007, the problem posed by the need for power-efficient hardware capable of handling high data traffic was addressed [84]. The adaptive quantization technique proposed in this article integrated a hardware-friendly algorithm with a digital CMOS image sensor. The quantization scheme involves boundary adaptation together with an online quadrant tree decomposition (QTD) processing. A later work presented a single-chip CMOS imaging sensor capable of image acquisition, as well as storage and compression [85]. To facilitate robust image compression, the authors exploited an adaptive quantization strategy based on online, least storage QTD processing preceded by fast boundary adaptation rule (FBAR), and differential pulse code modulation (DPCM). The demand for power-efficient hardware in real-life visual systems was also acknowledged in [86], where the proposed solution for energy-efficiency optimization was to tune the sensor quantization directly on the raw image for an end application—in this case, visual simultaneous localization and mapping (SLAM). Furthermore, the authors show that their proposed gradient-based image quantization scheme achieves high energy savings and high SLAM accuracy. In [87], a hybrid image compression framework based on vector quantization and DPCM was proposed as an ideal candidate for on-chip image compression.

There also exists a lot of work in the literature relating to hardware advancements in the field of image quantization. One notable work featured a power and resource-efficient hardware implementation of an image quantizer for capsule endoscopy applications [88]. The architecture is implemented using the CMOS 0.18-μm technology. In addition, a power-efficient CMOS sensor was developed that supports column-parallel single-slope/SAR (synthetic aperture radar) quantization scheme [89]. The architecture comprised of a 3-bit analog-to-digital converter (ADC) and an 8-bit approximation register (SAR) ADC. The distinguishing feature of this architecture is the selection of smaller SAR ADC reference voltages after the quantization of the first three most significant bits (MSBs).

E. High Dynamic Range Imaging

In computer vision, the term dynamic range indicates the change in intensity corresponding to the brightest detail in comparison to the darkest detail in an

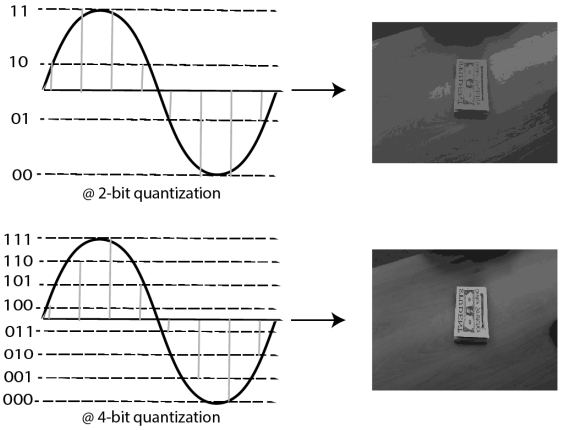


Fig. 4. ***Software-defined adaptive quantization allows the user to switch from a higher bit depth to a lower bit depth and vice versa depending on the situation. This figure shows the two different quantization representations of the same image. Lower bit depth representation introduces artifacts in the image, which may not be a problem for certain end applications and, therefore, can be adopted to optimize the overall energy efficiency of the system.***

image. The dynamic range of a CCD or CMOS sensor is the ratio of the maximum measurable signal and the noise floor [90]. Human visual systems have a very high-luminance dynamic range, thus allowing humans to perceive a great degree of contrast in a real-life scene. However, conventional image sensors are typically limited to 8-bit image channels, and the display devices are likewise constricted to a very small dynamic range, thus rendering the captured images unable to mimic the results that would have been obtained with human eyesight, as the digital images fail to capture the real-life brightness changes in the scene. HDR imaging technology has been developed over the years to circumvent the problem of a limited dynamic range of camera sensors. HDR imaging systems result in images of similar dynamic range and color to scenes observed by human eyes. As such, it has widespread applications in robotics, medical imaging, security, computer gaming, digital cinema/photography, and so on.

Over the years, researchers have innovated the existing image sensor designs to enable sensor-level HDR imaging. Changing the linear response of sensor pixels to logarithmic response has been shown to be an effective way of rendering HDR images [91], [92], [93], [94], [95], [96], [97]. Dynamic range programmability provides scope for improving the latency and energy efficiency of the overall imaging system. Keeping this in mind, we would like to point out that CMOS image sensors with linear-logarithmic hybrid pixel response with tunable dynamic range have already been proposed [98], [99]. Not only that, programmable pixel response has also been implemented in a digital pixel sensor along with dynamic range programmability in [100].

Single-shot HDR imaging is also of paramount importance as it significantly reduces the processing time owing to the input data being just a single image. In addition, it also ensures that there are no motion-induced ghosting artifacts that cannot be guaranteed in multiple-exposure multishot HDR images [101], [102], [103]. As such, research in HDR image acquisition with a single shot has been gaining momentum. In [104], a deep optical-based HDR imaging approach was proposed, where the encoder is modeled by the point spread function of the leans and the decoder is parameterized by a convolutional neural network (CNN). Similarly, other works of research are available in the literature, which performs single-shot HDR imaging [105], [106], [107], [108]. Very recently, a novel algorithm has been proposed for rendering single-shot HDR images by exploiting quantization noise information [109].

Zhao et al. [110] came out with a special kind of single-exposure HDR imaging sensor called a modulo camera. The sensor is able to receive unbounded radiance levels by retaining only the least significant bits as opposed to going into saturation mode. An unwrapping algorithm is employed to recover high radiance levels from a single modulus image. Modulo camera-based research is gaining traction in the vision community [111], [112].

IV. ISP PIPELINES

In Section III, we showed how recent research has enabled image sensors to configure the raw pixel values coming off the image sensor based on resolution/ROI, exposure, quantization, and frame rate. However, before these data can be usable, the traditional pipeline for processing images known as the image sensor processor (also called ISP) or ISP is employed. In this section, we describe the basics of the ISP and then proceed to show how SDI research has modified, augmented, and, in some extreme cases, removed the ISP to realize more flexibility and performance gains. Note that we will not highlight research that aims to make traditional ISPs better in their image quality, but rather look at work that reconceptualizes the ISP for the SDI paradigm.

A. ISP Pipeline Basics

The ISP unit processes the RAW data coming from the sensor to produce a complete image, typically in JPEG or PNG format. ISPs contain a series of signal processing stages that are designed to make the images appealing to human vision (see Fig. 5). While the precise stages of an ISP pipeline vary and are typically proprietary to the manufacturer, we describe a typical set of stages in the following.

1) Denoising: For RAW images, four sources of noise are dominant during sensing: *shot noise*, due to the physics of light sensing; *dark current* and *quantization noise* from the ADCs, and *read noise* from the readout electronics. Much research has gone into denoising algorithms, such as BM3D [113] and nonlocal means [114], to improve image SNR without blurring important high-frequency

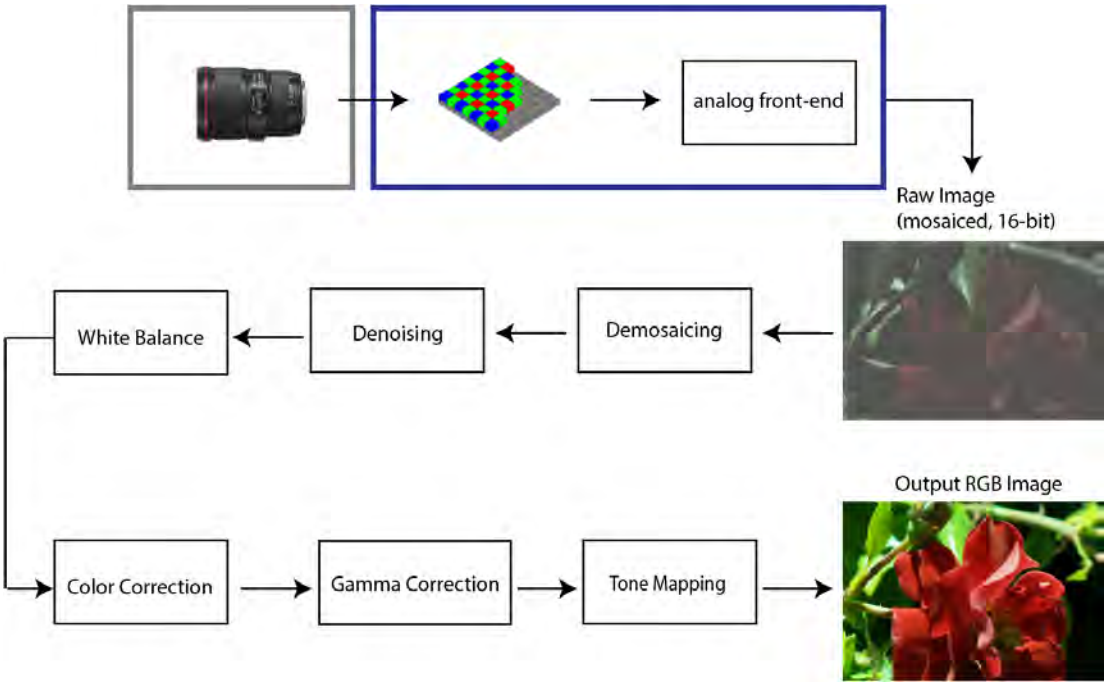


Fig. 5. ***Typical ISP pipeline that processes mosaiced raw images and converts them to usable images in the RGB color space. A typical ISP pipeline will first perform demosaicing followed by denoising and white balancing. Color correction, gamma correction, and tone mapping will, finally, afford the user the desired RGB image.***

image features, such as edges and textures. Denoising can be computationally expensive with up to 95% of resources in a highly optimized software pipeline [115]. Algorithm performance is particularly challenged in low-light scenarios where the shot noise can dominate other noise sources.

2) Demosaicing: The ISP transforms the RAW image into a conventional three-channel color image. Each pixel in the RAW image contains either red, green, or blue data arranged in the commonly used Bayer filter array. To fill the other two missing color values at each pixel, demosaicing interpolates this data for each pixel from neighboring pixels. Simple interpolation algorithms, such as nearest-neighbor or averaging, lead to blurry edges and other artifacts, so more advanced demosaicing algorithms use gradient-based information at each pixel to help preserve sharp edge details.

3) Color Transformations and Gamut Mapping: Color transformation stages translate the image into a color space that is visually pleasing. These transformations are local, per-pixel operations, given by a 3×3 matrix multiplication. The first transformations are color mapping and white balancing. Color mapping reduces the intensity of the green channel to match that of blue and red, and includes modifications for artistic effect. White balancing converts the image's color temperature to match that of the lighting in the scene. The matrix values for these transformations are typically chosen specifically by each camera manufacturer for esthetic effects.

Gamut mapping processes color values captured outside of a display's acceptable color range into acceptable color values. Gamut mapping is a nonlinear, per-pixel operation. Sometimes, images are transformed into non-RGB color spaces, such as YUV or HSV [116].

4) Tone Mapping: Tone mapping is a nonlinear, per-pixel function with multiple responsibilities. It compresses the image's dynamic range and applies additional esthetic effects. Typically, this process results in an esthetically pleasing visual contrast for an image, making the dark areas brighter, while not overexposing or saturating the bright areas. One type of global tone mapping called gamma compression transforms the luminance of a pixel p (in YUV space). However, most modern ISPs use more computationally expensive, local tone mapping based on contrast or gradient-domain methods to enhance image quality, specifically for high dynamic range scenes, such as outdoors and bright lighting.

5) Compression: The final stage of most ISP pipelines is image compression to help reduce storage and transmission requirements for the image data. The most common image compression standard is JPEG, which uses the discrete cosine transform quantization to exploit signal sparsity in high-frequency space. Other algorithms, such as JPEG 2000, use the wavelet transform, but the idea is the same: allocate more stage to low-frequency information and omit high-frequency information to sacrifice detail for space efficiency. This compression algorithm is typically

physically instantiated as a codec that forms a dedicated block of logic on the ISP.

B. Programmable ISPs

One of the main research thrusts in SDI is to allow the ISP stages to become programmable, thus enabling flexibility in the images being computed by the vision system. While there has been a lot of work on enhancing the ISP pipeline for improved visual quality including neural network-based improvements [117], [118], we will highlight works that enable better programmability and flexibility for vertically oriented imaging tasks. However, we will focus on algorithmic advances in this section, while software and infrastructure development, such as open-source camera pipelines, will be discussed in Section VI.

One of the key breakthroughs in ISPs was the realization that a unified optimization framework can be used instead of the different stages described earlier. In FlexISP, the authors propose this optimization and show improvements over classical algorithms for each of the stages [115]. Furthering this work, they show that these optimizations are known as proximal algorithms and design a programming language framework for easily writing algorithms [119].

New research has aimed to optimize application-specific quality and performance metrics with hardware-in-the-loop ISP pipelines. In [120], a covariance matrix adaptation evolution strategy (CMA-ES) is proposed to minimize a multiobjective optimization problem that encodes metrics for the downstream vision or imaging application. This optimization was shown to achieve superior performance than manual tuning or other automated ISP parameter optimizations. However, occasionally developers do not get access to the ISP hardware, as vendors typically only give a black-box abstraction of the ISP with parameter values as simple register values with ranges to the end user. To address this problem, recent work has introduced differentiable mapping between the parameter configuration space and evaluation metrics, using a CNN in an end-to-end fashion with imaging hardware in-the-loop [121]. This allows even black-box image processing pipelines to be efficiently optimized, reducing design time from months to hours. This pipeline is validated on experimental data for real-time display applications, object detection, and extreme low-light imaging.

C. Reduced ISP Pipelines

Beyond making ISPs more programmable, the other trend has been to reduce the computational processing of ISPs, in particular, for machine vision applications, where esthetic visual quality is not the main priority. This work aims to design minimal or even ISP-free camera pipelines to allow for improved latency, energy efficiency, and lowered system complexity.

One key paper in this area did extensive studies of the effects of different ISP stages for a suite of computer vision tasks [122]. The authors discovered that most of the color transforms were largely not influential for the end task performance and identified three main stages of the ISP: demosaicing, denoising, and gamma compression. Gamma compression was a surprising result, as it turned out that enhancing the local contrast of the images helped improve feature extraction and detection that computer vision tasks rely on for their processing. Following these insights, the authors propose a new ISP-free pipeline by subsuming these key stages directly into the mixed-signal image sensor itself and showed that they could potentially save 75% of energy by doing so [122].

Special attention has focused on obviating the need for demosaicing in machine vision tasks. Zhou et al. [123] observed that image gradient operations common to most vision feature extraction, such as SIFT, can be adapted to the RAW Bayer pattern, eliminating the need for demosaicing. Binary features [124], edge detection [125], and even scale-invariant feature transform/speeded up robust features (SIFT/SURF) [126] have been successfully implemented on Bayer images. From these studies, it has become clear that the need for high-quality demosaicing and color transformations is not critical for certain computer vision task performances. Indeed, many machine vision cameras utilize monochrome CMOS image sensors in this same line of reasoning.

Similarly, the extent to which vision algorithms need image denoising has been a topic of research in the community. Liu et al. [127] demonstrated that image denoising could help overcome performance degradation due to noise for semantic segmentation and image classification. Borkar and Karam [128] showed that deep CNNs can be particularly vulnerable to noise due to training and overfitting issues, and require modifications to be resilient at inference time. However, more recent work leveraging state-of-the-art neural radiance fields has shown robustness to noisy images, even outperforming 3-D reconstruction methods that utilized denoising for low-light images [129]. Furthermore, several papers are optimized directly on RAW images without explicit denoising [130], [131], [132], as described later in this section. As a result, there does not seem to be a clear consensus in the research literature on the necessity for image denoising for computer vision tasks.

The observations made by Buckler et al. [122] on the importance of gamma compression (which is a type of global tone mapping) for feature extraction have been corroborated by other researchers. Hansen et al. [133] confirmed that tone mapping was critical for image classification, especially for HDR images. Lu and Murmann [134] posited embedding gamma compression directly into the image sensor itself through log-gradients and show that this can potentially yield superior energy efficiency for vision tasks.

Finally, there have been several research studies for optimizing vision applications on RAW data directly. For instance, object detection, in particular, has shown

advances by adapting the algorithms to work on RAW data directly [130], enabled by the release of the PASCALRAW database from Stanford University [135]. Recent work has claimed superior performance on RAW images using learnable layers to adapt to this data directly [131]. Zhang et al. [132] skip the ISP directly for their histogram of gradient-based object detector. In addition, Christie et al. [86] showed the usefulness of RAW images for visual SLAM algorithms. RAW images have even been shown to be robust to adversarial or security attacks that can alter deep learning performance [136].

In conclusion, reducing or eliminating the ISP pipeline can lead to improved task efficiency for given applications. In particular, the focus on object detection, motivated by autonomous vehicles and navigation, has shown that the full ISP pipeline is unnecessary and can be modified without loss of performance. However, most of the work in this field is empirical in nature as it is difficult to prove when a certain ISP task, such as denoising or tone mapping, is essential for a given application.

V. DEEP LEARNING AND SDI

Modern applications in computer vision leverage recent advances in machine learning, particularly deep learning, which has revolutionized that field [137]. CNNs utilize a series of convolutional layers combined with nonlinearities and pooling/downsampling functions to extract features from images, and these networks are trained in either fully, semi-, or self-supervised fashion depending on the application. CNNs have become state-of-the-art for image and video classification, semantic segmentation, object detection and tracking, and even low-level tasks, such as image reconstruction, denoising, and processing. Thus, most SDI systems are now targeted for deep learning applications where their processed images are utilized by CNNs (or other variants of neural networks) for downstream tasks.

As a result, deep learning has also impacted SDI research in a variety of ways. In particular, deep learning has been used to augment or even replace the ISP pipeline, as described throughout Section IV. Differentiable ISP pipelines have been proposed to tune various ISP stages [117], [121], [138], [139]. The key is differentiability: making the end-to-end pipeline differentiable from image capture to vision task allow for joint optimization and the backpropagation of gradient information to update network layers. This differentiability has even extended to the realm of optical computing, allowing the codesign of optics and sensors with deep learning networks [104], [140], [141], [142].

However, the advances of deep learning have come at a significant cost with higher energy and power consumption. Deep neural networks can have millions of trainable parameters with hundreds of layers. Training these networks has been shown to have a huge energy and compute cost [143], but, surprisingly for deployed systems, inference costs dominate [144]. These costs have a direct impact on the field of SDI. While a primary focus for SDI has been on reconfigurability and enabling new sensing functionality, there has been an emerging interest to reduce the overall energy and compute costs for vision systems. For computer hardware, the main focus has been to transition from power-hungry graphics processing units (GPUs) to custom-designed hardware for neural network acceleration (see the survey paper by Sze et al. [145] for an overview of the field). In addition, the tiny machine learning (TinyML) community has grown to develop technology and solutions for low-power neural network performance on embedded microcontrollers and other distributed platforms [146], [147].

Another main limitation of deep learning with respect to SDI is the necessity of large amounts of training data for supervised learning, which is the dominant paradigm in deep learning to-date. While there exists large image and video datasets, including ImageNet [148] and YouTube-8M [149], corresponding RAW data for analyzing SDI are more difficult to store in large volumes. Furthermore, deep learning methods are subject to overfitting on their training data [150], and thus, testing a deep learning algorithm for data collected under different environments or camera parameters can yield domain mismatch and errors, as shown by Liu et al. [151]. However, as deep learning algorithms appear to be the main infrastructure behind modern computer vision, it is clear that SDI research must adapt to these limitations and be optimized for neural networks. In Section VI, we discuss hardware acceleration and software, many of which are targeted at deep learning processing. Furthermore, in Section VII, we outline a series of simulation and analysis frameworks that can make training and testing of deep learning algorithms potentially easier for SDI.

VI. HARDWARE ACCELERATION AND SOFTWARE

The key to SDI is programmability, which allows sensors to adaptively change their sensing behavior to meet application demands. To that end, we discuss, in this section, works that explore different hardware and software abstractions that enable the programmability and reconfigurability of different sensor primitives in the visual computing pipeline.

A. Hardware Acceleration

Here, we discuss works that propose domain-specific hardware accelerators from academia and industry for efficient image processing and computer vision. While some chips provide fixed functionality, others provide the ability to program the vision model and also adapt the vision model to suit different data flows. In addition to vision chips, we also discuss other hardware chips that enable computational photography use cases.

With CNNs becoming popular due to their superior task accuracy, different researchers proposed optimization techniques and design innovations for efficient and performant CNN inference. Specifically, researchers proposed

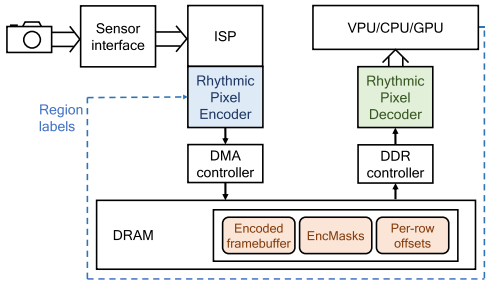


Fig. 6. ***Rhythmic encoder decimates the incoming pixel stream from the camera and encodes only region-specified pixels into memory. The rhythmic decoder decodes the pixels for use with the vision algorithms. The runtime sits on top of these hardware extensions and coordinates with the vision applications to govern the encoder/decoder operation [152].***

domain-specific accelerators with efficient dataflows to minimize the data movements across memory interfaces.

Eyeriss-v1 [153] proposes row-stationary dataflow that exploits local data reuse of filter weights and feature map activations in high-dimensional convolutions and also minimizes the data movement of partial sum accumulations. Even though the Eyeriss architecture provisions sufficient local memory for storing intermediate results, it still relies on DRAM to store most of the weights. Efficient inference engine (EIE) [154] eliminates the dependence on DRAM by having sufficiently large local SRAM and storing weights in that structure. It does so through deep compression for weights by pruning the redundant connections in the network and also by having multiple connections sharing the same weight.

Meanwhile, researchers proposed hardware implementations with support for reduced computing precision toward higher throughput and energy efficiency. Envision proposes dynamic-voltage-accuracy-frequency scaling (DVAFS) that not only reduces switching activity for low-precision computations by masking a configurable number of multiply-and-accumulate (MAC) units but also reuses the inactive arithmetic cells at reduced precision [155]. Thinker [156] uses bit-width adaptive computing, on-demand array partitioning, and memory-gathering techniques to substantially reduce redundant memory accesses. UNPU [157] has an architecture that reuses input feature maps and implements variable weight bit precision from 1 to 16 b for energy-efficiency.

Eyeriss-v2 [158] builds on top of the above works and specifically focuses on supporting diverse filter shapes and compressed domain processing for compact and sparse DNNs. This is achieved by introducing a highly flexible on-chip network that can adapt to different amounts of data reuse and bandwidth requirements of different data types. By doing so, Eyeriss-v2 can support a wide variety of compact and sparse DNN models, thereby improving the performance and energy efficiency of CNN inference. Complementary to Eyeriss-v2 and other works, more recently, Tianjic [159] and Simba [160] propose a decentralized hardware architecture where multiple cores communicate with each other through message passing over interconnects, without the need of global off-chip memory.

While CNN accelerators solely focus on optimizing data flows through maximal data reuse within a single component in the pipeline, there are other architecture works that deploy hardware–software techniques to reuse computation across different components in the vision pipeline. Euphrates [161] reuses the motion vector information from the ISP to extrapolate bounding box results for object detection to avoid extraneous vision processing on certain frames. EVA^2 [162] exploits temporal redundancy between frames and uses the activation motion compensation technique to approximate the CNN result. ASV [163] applies stereo-specific algorithmic and computational optimizations to increase the performance with a minimum error rate.

More recently, rhythmic pixel regions (see Fig. 6) integrate a scalable hardware interface that selectively discards pixels before the camera pixel stream enters DRAM [152]. The hardware interface is augmented with a software runtime that runs on top of vision applications. The software runtime allows developers to flexibly specify region labels with independent spatiotemporal resolutions. Together these hardware and software interfaces result in significant memory traffic reduction, thereby enabling the energy efficiency of visual computing systems.

In parallel to academic chips, the industry came up with programmable vision processing units, popularly known as neural processing units (NPUs), for accelerating vision applications. Apple integrated an NPU inside their recent line of iPhones, which is used in conjunction with built-in 3-D sensors, primarily for the FaceID use case, i.e., unlocking the phone using the user's face.[1,2] Recently, NPUs have been more broadly used to run other vision tasks, such as semantic segmentation. Along similar lines, Google integrated an edge tensor processing unit (TPU) inside their recent Pixel smartphone, which can be programmed to execute a wide variety of vision tasks. Samsung[3] and Qualcomm[4] integrated NPUs for similar use cases in their recent Exynos and SnapDragon chipsets, respectively. Finally, Intel came up with Myriad2 VPU[5] that is aimed at other edge use cases, such as drone navigation, instead of smartphone-based vision.

In recent years, domain-specific hardware accelerators have been proposed to accelerate not only vision tasks but also imaging tasks. Specifically, for computational photography, Google integrated Pixel Visual Core into their Pixel smartphone product line.[6] This chip tightly interacts with the camera and configures the camera parameters, such as

[1]https://github.com/hollance/neural-engine
[2]https://support.apple.com/en-us/HT208108
[3]https://www.samsung.com/global/galaxy/galaxy-z-fold4/
[4]https://www.qualcomm.com/products/smartphones/mobile-a
[5]https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html
[6]https://en.wikipedia.org/wiki/Pixel_Visual_Core

exposure for use cases, such as HDR imaging. Samsung's smartphones also have a similar imaging core inside their Exynos chipset for computational photography use cases.

B. OS/Runtime Configurations

Here, we discuss the works that provide APIs for low-level control of mobile/embedded camera parameters and frameworks that enable seamless reconfiguration of camera parameters.

The FrankenCamera API provides full control to the programmer over different camera settings for each frame [164]. Specifically, FrankenCamera enables developers to capture a stream of camera images with fine-grained precision control. There are four key elements in FCam API: 1) shots; 2) sensors; 3) frames; and 4) devices. A shot bundles different camera parameters, such as gain, exposure, resolution, and frame rate, which completely describes the capture and postprocessing of a single image. The sensor is responsible for managing the entire pipeline. It takes a shot and pushes that to the request queue to capture a single image Alternatively, it can also capture a burst of shots. The sensor produces frames that can be retrieved using `getFrame` method from a queue of pending frames. Finally, devices help control the lens focus and flash settings to control the brightness of the scene. FCam also allows access to supplemental statistics such as histograms from ISP if available.

Android Camera2 API uses similar concepts as FCam for camera control on mobile devices.[7] As a next step to FCam, the Khronos group proposed the OpenKCam API standard with an aim to provide even more fine-grained control to developers on the entire capture pipeline [165]. OpenKCam offers controllability on different components in the capture pipeline. It provides control to change the color filter array (CFA) on the sensor side, whereas it provides control over focus distance, aperture, and focal length on the lens side. Along similar lines, on the flash side, it offers control over the flash duration activity, whereas it offers control over the resolution, ROI extraction, quality, and format on the output side. It also helps tune different knobs in the ISP, such as demosaicing quality, denoising quality, gamma correction, and color-space conversion. Finally, it also offers control over multiple sensors where there will be one master sensor controlling other sensors.

Meanwhile, there are also commercial image sensors with APIs for multi-ROI support [166]. However, due to hardware restrictions, these APIs allow the developer to configure only 16 ROIs, thereby limiting their flexibility. Furthermore, these APIs typically do not allow the ROIs to overlap and force the ROIs to be aligned in the same column. While a few tens of ROIs could be sufficient for simple tasks, such as face detection, tasks such as visual SLAM need several hundreds of ROIs for task fidelity.

In addition to low-level camera control APIs, there are frameworks that enable seamless reconfiguration of camera parameters. In particular, reconfiguring camera parameters, such as spatial resolution, incurs a significant amount of latency up to a few hundred milliseconds, thereby affecting pipeline performance. This is because the existing pipeline needs to undergo several sequential steps, such as flushing the existing frames, allocating memory to new frames, and starting the pipeline afresh. The Banner framework [1] avoids these time-consuming steps by allocating memory for the highest possible resolution so that any resolution image can be stored in that location. On top of that, Banner modifies the existing V4L2 framework to let it read only the actual resolution size number of pixels from memory whenever it needs to service request from a vision application.

C. Programming Languages and Compilers

In addition to domain-specific accelerators and domain-specific runtimes, researchers proposed domain-specific languages/compilers for building flexible and efficient imaging pipelines.

Rigel [167] is a framework that allows developers to express image processing pipelines in Lua programming language and compiles them into highly efficient hardware descriptions, which can be mapped onto an FPGA with minimal buffering. While there are a lot of high-level synthesis tools, they can work for lightweight image-processing tasks with the same amount of data flowing in and out of the pipeline components Rigel instead supports sophisticated vision tasks, such as feature extraction, which entails different input and output data rates in different pipeline components.

Darkroom [168] extends Rigel to support any hardware target. That is, Darkroom can compile high-level image processing code into line-buffered hardware pipelines for FPGA/application-specific integrated circuit (ASIC)/CPU targets. Darkroom has a simple programming model with different abstractions defined to express different image processing operations, such as convolution. Darkroom analyzes the code and comes up with a suitable scheduling policy to eliminate any pipeline hazards.

Meanwhile, Khronos came up with an OpenVX standard, which is an open standard API targeted at low-power and real-time embedded vision.[8] OpenVX lets developers express vision applications as a *graph* of image processing operations called nodes. These nodes can be mapped on any processing unit and can be coded in any programming language. Furthermore, these nodes could be fused to eliminate memory transfers, and processing can be tiled to keep data entirely in local memory for optimizing power and performance.

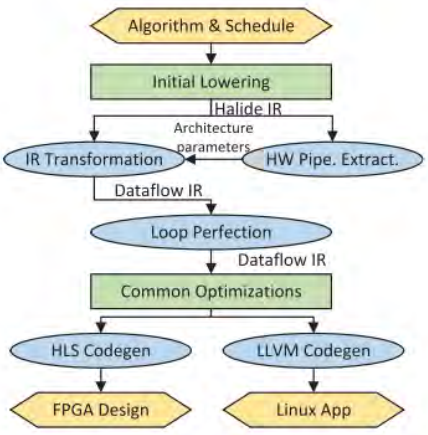
Halide [169] is a domain-specific language that allows developers to specify an algorithm along with a composable schedule for high-performance implementations.

[7] https://developer.android.com/reference/android/hardware/camera2/package-summary

[8] https://www.khronos.org/registry/OpenVX

```
UniformImage in(UInt(8), 2)
Var x, y
Func blurx(x,y) =  in(x-1,y) +    in(x,y) +    in(x+1,y)
Func out(x,y) = blurx(x,y-1) + blurx(x,y) + blurx(x,y+1)
```

(a)



(b)

Fig. 7. *(a) Simple 3 × 3 box filter implementation using Halide [169]. (b) Compilation flow [170]. Blue circles are new. Green rectangles are unchanged/existing Halide compilation passes. Yellow hexagons are the inputs and outputs of the compiler.*

Halide's compiler then stochastically searches over the space of schedules to find an optimal one for achieving high performance. It also takes into account different tradeoffs between locality and parallelism before generating hardware code. For instance, as shown in Fig. 7(a), a 3×3 unnormalized box filter can be simply expressed as a chain of two functions in x, y using Halide, without any extra data structures.

Manually specifying schedules for image processing pipelines could be cumbersome for nonexpert developers. To that end, Mullapudi et al. [171] proposed an algorithm that automatically generates high-performance schedules for Halide programs. The algorithm builds on top of function bounds analysis that is already present in the Halide compiler to automatically determine the right set of locality and parallelism-based program transformations.

To further reduce developer burden, more automation has been done in recent years through the deployment of differential programming principles and through the development of highly efficient libraries. Li et al. [172] extended Halide with reverse mode automatic differentiation to automatically optimize gradient computations for different programs, thereby reducing programmer burden. More recently Ahmad et al. [173] proposed DEXTER that automatically translates image processing libraries to Halide. Specifically, DEXTER parses a C++-based image processing function into a directed acyclic graph (DAG) of smaller stages, infers the semantics of each stage in the DAG expressed in a high-level IR, and, finally, compiles IR into executable Halide code.

VII. PUBLIC RESEARCH INFRASTRUCTURE

In this section, we discuss some of the available research tools that aid in the implementation of software-defined algorithms. Since vertically oriented solutions are rare and typically proprietary in the industry, researchers must cobble together various simulators, tools, and hardware to implement their solutions. As such, it is difficult to unify these tools under a common analysis framework, so we instead list them out thematically.

A. Camera APIs and Sensor Modeling

Several camera APIs have been discussed in Section VI, including the FrankenCamera API [164] and OpenKCam [165]. They provide control over different camera parameters; however, the number and type of camera parameters accessible are dependent on the camera manufacturer. In Table 1, we show the modifiable parameters from the FrankenCam API implemented on the N900 camera. The FCam API is available as open source.[9] Unfortunately, there is a dearth of available camera APIs that are user-friendly for research purposes, and more public toolkits should be released with documentation to investigate how to reconfigure image sensors.

Similarly, modeling sensor energy, such as the cost of reading out a set of pixels or how much energy a column ADC consumes, is difficult for researchers to ascertain. This is due to the varying design varieties and proprietary details that go into commercial imaging systems. Simple models have assumed per-pixel energy consumption based on available sensor specifications [134], [152], [174] or per-frame [175] and image sensor quantization energy consumption based on the ADC detection levels [86], [122]. An open-source sensor modeling tool has been provided by Iqbal et al. [36], where power efficiency has been modeled for a conventional mobile system based on numbers curated by Kodukula et al. [152]. The tool enables on-the-fly digital reconfiguration of sensor mechanisms, such as ROI, quantization, and noise, and computes sensor power consumption based on the aforementioned energy model, wherein the SoC-DRAM communication unit, storage, and computation per MAC are assigned per-pixel energy consumption numbers reflecting real-world data. This tool serves as an energy optimization framework

[9] http://fcam.garage.maemo.org/

Table 1 Example of Modifiable Parameters on FrankenCam API

	FrankenCam N900 Implementation
Exposure	38 - 2489140 µS
Gain	Analog & Digital 1.0 - 32
Frame Rate	0.4 - 30 fps
Resolution	160x120 2592x1968
Bit Depth	16, 24 bits
Format	DNG, JPEG
External Devices	Lens, Flash, Gyroscope

that determines optimal sensor configuration for energy-efficiency maximization.

B. ISP Simulators

The Image Systems Engineering Toolbox (ISET) for Cameras is one of the most comprehensive toolkits for assessing the quality of images, including handling consistent radiometric units for physical accuracy [176], [177]. ISET has a wide user base (500+ people) working across 24 different countries. The toolflow incorporates an extensible and open-source ISP pipeline for proper visualization of images. The simulated ISP processor includes several conventional techniques enabling basic camera functionality. Some of these techniques are estimating missing RGB values (demosaicing) and mapping RGB values to an internal color space for encoding and display (color balancing, color rendering, and color conversion). The proposed simulator also accommodates the inclusion of proprietary image-processing algorithms and facilitates the assessment of these algorithms under varying imaging conditions and sensor types. This simulator can be found open source[10] and integrations with physics-based rendering engines.[11]

For a more simplified ISP pipeline, OpenISP is an accessible version that contains a few tuning functions for hardware configuration and is suitable for teaching purposes [178]. The simulated ISP pipeline processes RAW images with various modules, including black level compensation, CFA interpolation, edge enhancement, hue/saturation/control, and brightness/contrast control. In contrast, to forward ISP simulators, such as ISET and OpenISP, Kim et al. [179] introduced a reversible ISP pipeline that takes processed JPEG or PNG images and converts them back to RAW images with minimal error. This is useful for scenarios where one does not have access to the RAW images or their metadata. This pipeline was later expanded to include both forward and reversible ISP operations by Buckler et al. [122], and both the reversible pipeline[12] and the expanded pipeline[13] are publicly available. Recently, a reversible ISP pipeline by Xing et al. [180] leveraged a differentiable JPEG compression simulator for superior performance.[14]

C. FPGA Acceleration

Field-programmable gate arrays (FPGAs) are semiconductor devices designed around configurable logic blocks. Their reconfigurable architecture provides higher flexibility than custom-designed ASICs while improving energy and area efficiency over a traditional microprocessor. Similar to ASICs, FPGAs are programmed using hardware description languages (HDL) that describe the structure and behavior of the desired electronic circuit. In the past,

[10]https://github.com/ISET/isetcam
[11]https://github.com/ISET/iset3d/wiki
[12]https://github.com/mbuckler/ReversiblePipeline
[13]https://github.com/cucapra/approx-vision
[14]Code available here: https://github.com/yzxing87/Invertible-ISP

Fig. 8. *Example of FPGA implementation flow using HLS tools presented in [36].*

the implementation of algorithms on an FPGA required the knowledge of HDL, such as Verilog or VHDL. However, high-level synthesis (HLS) tools have been developed in order to allow the deployment of algorithms by using high-level programming languages. In [181], an exhaustive list of current and abandoned FPGA HLS tools is presented. Some of these HLS tools have developed libraries and plugins specifically designed to accelerate custom vision tasks and applications. One example of these libraries is the ones provided by Xilinx. Vitis AI allows the acceleration of neural networks on Xilinx FPGAs, while the Vitis Vision library provides optimized kernels based on OpenCV.

Iqbal et al. [36] introduced an HLS toolflow for accelerating an object tracking algorithm that uses adaptive subsampling. Fig. 8 shows an example of their implementation flow using HLS tools. This flow allows developers to deploy algorithms on Xilinx platforms while using familiar programming languages, such as C++ and Python, and high-level domain-specific frameworks, such as TensorFlow and OpenCV. Vitis takes advantage of the OpenCL standard to generate application kernels using an array of accelerated libraries. These kernels are hosted on an ARM or x86-based processor, which communicates via a PCIe port with the device, a physical collection of hardware resources onto which the kernels are executed. The Xilinx Runtime Library (XRT) is responsible for the communication between the application code and the application kernels during runtime. In this example, an evaluation board (ZCU102) that contains two ARM processors and a ZYNQ UltraScale+ MPSoC FPGA is shown. This pipeline is available as open source.[15]

D. Energy Estimation for Deep Learning Accelerators

For many SDI researchers, it is critical to model not only the sensing cost but also the computing cost for the end application. While this is very task-dependent, researchers have introduced several tools to estimate the hardware cost for accelerating vision workloads, particularly those of CNNs and deep learning. Yang et al. [182], [183] introduced energy modeling for deep neural networks based on their layers and sparsity, whose method is available for use on their website.[16] This modeling work was further extended to hardware accelerators [184].[17] However,

[15]https://github.com/oiqbal95/fpga_adaptive_tracking
[16]https://energyestimation.mit.edu./
[17]Tool available here: http://accelergy.mit.edu/

we caution readers that all these tools are design and technology-dependent, and further design and modeling are needed to determine the energy costs for one's own solution.

VIII. CONCLUSION

In this article, we show how the field of SDI has advanced over the past several decades. Unique to this body of literature is the combined emphasis on hardware–software codesign, where computer hardware and sensors are built alongside the accompanying software and algorithms that control them. This is also the main reason for the interdisciplinary nature of the field, requiring knowledge spanning optics and image sensing physics [9], [185], computer engineering and architecture [186], programming languages [187], OSs, and machine learning/computer vision [137], [188] to fully leverage these vertically integrated imaging platforms.

For recent research, there has been a plethora of new papers appearing in major conferences and journals, including IEEE Computer Vision and Pattern Recognition (CVPR) Conference, International Conference on Computational Photography (ICCP), IEEE TRANSACTIONS ON COMPUTATIONAL IMAGING (TCI), Siggraph, Mobisys, International Symposium on Computer Architecture (ISCA), Architectural Support for Programming Languages and Operating Systems (ASPLOS), and many more. Beyond research, there has been the adoption of standards and consortiums, such as OpenVX and Khronos group, aiming to standardize development for these image-sensing interfaces. Finally, organizations such as TinyML help champion machine learning and computer vision at the edge, which will leverage SDI research significantly in the industry and commercial sectors. We encourage readers to seek out these resources and organizations to further engage with this growing field.

There are many exciting avenues for new research for SDI, particularly as machine learning and computing on the edge become more prevalent in our sensing devices. For instance, the rise of machine learning has allowed for hardware–software codesign for end-to-end systems, including systems that optimize both the optics and back-end algorithms for enhanced performance [174], [189]. New sensors are still emerging every year in the industry. Event-based cameras [190] are offering attractive tradeoffs in terms of energy consumption and frame rate/latency, in exchange for sparsely sampling the visual scene and still achieving remarkable performance on end vision tasks, such as SLAM, HDR, and object detection. Quanta image sensors using jot pixels are able to achieve single photon sensing with Mpixel resolutions (1.1-μm-pixel pitch) and 1000+ fps frame rates [191] and have been started to be proposed for imaging applications [192].

In conclusion, we identify the opportunity for a research community to coalesce around SDI, build infrastructure and support for pursuing the advancement of science and technology, and realize applications in the commercial industry and other sectors. We hope that this survey article provides a suitable foundation of knowledge with which the reader can explore further and deeper pertaining to their own interests. ■

Acknowledgment

The authors thank Ravi Sivalingam and Evgeni Gousev for discussions surrounding this topic. They also thank Adrian Sampson for giving useful feedback on an early version of this draft.

REFERENCES

[1] J. Hu, A. Shearer, S. Rajagopalan, and R. LiKamWa, "Banner: An image sensor reconfiguration framework for seamless resolution-based tradeoffs," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2019, pp. 236–248.

[2] J. Mitola, III, "Software radios: Survey, critical evaluation and future directions," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 8, no. 4, pp. 25–36, Apr. 1993.

[3] "Cognitive radio. An integrated agent architecture for software defined radio," Ph.D. thesis, KTH-Roy. Inst. Technol., Stockholm, Sweden, 2000.

[4] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 25–30.

[5] A. Katidiotis, K. Tsagkaris, and P Demestichas, "Performance evaluation of artificial neural network-based learning schemes for cognitive radio systems," *Comput. Electr. Eng.*, vol. 36, no. 3, pp. 518–535, 2010.

[6] F. M. Ghannouchi, "Power amplifier and transmitter architectures for software defined radio systems," *IEEE Circuits Syst. Mag.*, vol. 10, no. 4, pp. 56–63, Nov. 2010.

[7] A. Abidi, "The path to the software-defined radio receiver," *IEEE J. Solid-State Circuits*, vol. 42, no. 5, pp. 954–966, May 2007.

[8] L. Schiavone, N. Browne, and R. North, "Joint tactical radio system—Connecting the gig to the tactical edge," in *Proc. MILCOM IEEE Mil. Commun. Conf.*, Oct. 2006, pp. 1–6.

[9] S. Jayasuriya, "Image sensors," in *Computer Vision: A Reference Guide*. New York, NY, USA: Springer, 2020, pp. 1–5.

[10] O. Schrey, J. Huppertz, G. Filimonovic, A. Bussmann, W. Brockherde, and B. J. Hosticka, "A 1k × 1k high dynamic range CMOS image sensor with on-chip programmable region-of-interest readout," *IEEE J. Solid-State Circuits*, vol. 37, pp. 911–915, 2002.

[11] B. Dierickx, D. Scheffer, G. Meynants, W. Ogiers, and J. Vlummens, "Random addressable active pixel image sensors," in *Proc. SPIE*. Princeton, NJ, USA: Citeseer, 1996, Art. no. 262512.

[12] D. Scheffer, B. Dierickx, and G. Meynants, "Random addressable 2048 × 2048 active pixel image sensor," *IEEE Trans. Electron Devices*, vol. 44, no. 10, pp. 1716–1720, Oct. 1997.

[13] Z. Zhou, B. Pain, and E. Fossum, "A CMOS imager with on-chip variable resolution for light-adaptive imaging," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 1998, pp. 174–175.

[14] Z. Zhou, E. R. Fossum, and B. Pain, "Integrated sensor with frame memory and programmable resolution for light adaptive imaging," U.S. Patent 6 057 539, May 2 2000.

[15] M. K. Law, A. Bermak, and C. Shi, "A low-power energy-harvesting logarithmic CMOS image sensor with reconfigurable resolution using two-level quantization scheme," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 2, pp. 80–84, Feb. 2011.

[16] M. F. Duarte et al., "Single-pixel imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 83–91, Mar. 2008.

[17] W. L. Chan, K. Charan, D. Takhar, K. F. Kelly, R. G. Baraniuk, and D. M. Mittleman, "A single-pixel terahertz imaging system based on compressed sensing," *Appl. Phys. Lett.*, vol. 93, no. 12, pp. 121105-1–121105-3, Sep. 2008.

[18] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[19] T. Sonoda, H. Nagahara, K. Endo, Y. Sugiyama, and R.-I. Taniguchi, "High-speed imaging using CMOS image sensor with quasi pixel-wise exposure," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, May 2016, pp. 1–11.

[20] L. Spinoulas, K. He, O. Cossairt, and A. Katsaggelos, "Video compressive sensing with on-chip programmable subsampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2015, pp. 49–57.

[21] Z. Y. Shin, H. S. Lin, T.-Y. Chai, X. Wang, and S. Y. Chua, "Programmable single-pixel imaging," in *Proc. 13th Int. Conf. Sens. Technol. (ICST)*, 2019, pp. 1–6.

[22] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar, "Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum," *IEEE Trans. Image Process.*, vol. 19,

no. 9, pp. 2241–2253, Sep. 2010.

[23] T.-C. Hsung and D. P. K. Lun, "New sampling scheme for region-of-interest tomography," *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 1154–1163, Apr. 2000.

[24] H. Lin, J. Si, and G. P. Abousleman, "Knowledge-based hierarchical region-of-interest detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 4, May 2002, p. 3628.

[25] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *Proc. Workshop Motion Video Comput.*, 2002, pp. 169–174.

[26] L. Marcenaro, M. Ferrari, L. Marchesotti, and C. S. Regazzoni, "Multiple object tracking under heavy occlusions by using Kalman filters based on shape matching," in *Proc. Int. Conf. Image Process.*, vol. 3, 2002, p. 3.

[27] X. Li, K. Wang, W. Wang, and Y. Li, "A multiple object tracking method using Kalman filter," in *Proc. IEEE Int. Conf. Inf. Autom.*, Jul. 2010, pp. 1862–1866.

[28] D. Y. Kim and M. Jeon, "Data fusion of radar and image measurements for multi-object tracking via Kalman filtering," *Inf. Sci.*, vol. 278, pp. 641–652, Sep. 2014.

[29] L. Zhang, B. Qiu, X. Yu, and J. Xu, "Multi-scale hybrid saliency analysis for region of interest detection in very high resolution remote sensing images," *Image Vis. Comput.*, vol. 35, pp. 1–13, Mar. 2015.

[30] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Feb. 2015, pp. 3119–3127.

[31] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4293–4302.

[32] H. Fan and H. Ling, "SANET: Structure-aware network for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Feb. 2017, pp. 42–49.

[33] Y. Song et al., "VITAL: Visual tracking via adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8990–8999.

[34] D. Mohan, S. Katoch, S. Jayasuriya, P. Turaga, and A. Spanias, "Adaptive video subsampling for energy-efficient object detection," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, 2019, pp. 103–107.

[35] O. Iqbal et al., "Design and FPGA implementation of an adaptive video subsampling algorithm for energy-efficient single object tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 3065–3069.

[36] O. Iqbal, V. I. T. Muro, S. Katoch, A. Spanias, and S. Jayasuriya, "Adaptive subsampling for ROI-based visual tracking: Algorithms and FPGA implementation," *IEEE Access*, vol. 10, pp. 90507–90522, 2022.

[37] E. Pietka, A. Gertych, S. Pospiech, F. Cao, H. K. Huang, and V. Gilsanz, "Computer-assisted bone age assessment: Image preprocessing and epiphyseal/metaphyseal ROI extraction," *IEEE Trans. Med. Imag.*, vol. 20, no. 8, pp. 715–729, Aug. 2001.

[38] J. Blasco, N. Aleixos, and E. Moltó, "Computer vision detection of peel defects in citrus by means of a region oriented segmentation algorithm," *J. Food Eng.*, vol. 81, no. 3, pp. 535–543, 2007.

[39] K. Matzen and N. Snavely, "BubbLeNet: Foveated imaging for visual discovery," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1931–1939.

[40] A. Patney et al., "Towards foveated rendering for gaze-tracked virtual reality," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.

[41] A. Patney et al., "Perceptually-based foveated virtual reality," in *Proc. ACM SIGGRAPH Emerg. Technol.*, 2016, pp. 1–2.

[42] E. Turner, H. Jiang, D. Saint-Macary, and B. Bastani, "Phase-aligned foveated rendering for virtual reality headsets," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Mar. 2018, pp. 1–2.

[43] X. Meng, R. Du, M. Zwicker, and A. Varshney, "Kernel foveated rendering," in *Proc. ACM Comput. Graph. Interact. Techn.*, vol. 1, no. 1, 2018, pp. 1–20.

[44] O. T. Tursun et al., "Luminance-contrast-aware foveated rendering," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.

[45] X. Meng, R. Du, and A. Varshney, "Eye-dominance-guided foveated rendering," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 5, pp. 1972–1980, May 2020.

[46] H. Kim et al., "Eye tracking based foveated rendering for 360 VR tiled video," in *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 484–486.

[47] A. Changwani and T. Sarode, "Low-cost eye tracking for foveated rendering using machine learning," in *Proc. IEEE Int. Conf. Cloud Comput. Emerg. Markets (CCEM)*, Sep. 2019, pp. 32–39.

[48] J. Bernacki, "Automatic exposure algorithms for digital photography," *Multimedia Tools Appl.*, vol. 79, pp. 1–26, Jan. 2020.

[49] O. Yadid-Pecht and E. R. Fossum, "Wide intrascene dynamic range CMOS APS using dual sampling," *IEEE Trans. Electron Devices*, vol. 44, no. 10, pp. 1721–1723, Oct. 1997.

[50] M. Mase, S. Kawahito, M. Sasaki, Y. Wakamori, and M. Furuta, "A wide dynamic range CMOS image sensor with multiple exposure-time signal outputs and 12-bit column-parallel cyclic A/D converters," *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2787–2795, Dec. 2005.

[51] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich, "Synchronization and rolling shutter compensation for consumer video camera arrays," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Aug. 2009, pp. 1–8.

[52] Y. P. Wang, "Imaging apparatus comprising image sensor array having shared global shutter circuitry," U.S. Patent 8 629 926, Jan. 14, 2014.

[53] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: Motion deblurring using fluttered shutter," in *Proc. SIGGRAPH*, 2006, pp. 795–804.

[54] M. Wei et al., "Coded two-bucket cameras for computer vision," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 54–71.

[55] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.*, vol. 26, no. 3, p. 70, 2007.

[56] Y.-W. Tai, N. Kong, S. Lin, and S. Y. Shin, "Coded exposure imaging for projective motion deblurring," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2408–2415.

[57] W. Xu and S. McCloskey, "2D barcode localization and motion deblurring using a flutter shutter camera," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Jan. 2011, pp. 159–165.

[58] D. Liu, J. Gu, Y. Hitomi, M. Gupta, T. Mitsunaga, and S. K. Nayar, "Efficient space-time sampling with pixel-wise coded exposure for high-speed imaging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 248–260, Feb. 2014.

[59] K. Park, S. Shin, H.-G. Jeon, J.-Y. Lee, and I. S. Kweon, "Motion deblurring using coded exposure for a wheeled mobile robot," in *Proc. 11th Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, 2014, pp. 665–671.

[60] Y. Luo, J. Jiang, M. Cai, and S. Mirabbasi, "CMOS computational camera with a two-tap coded exposure image sensor for single-shot spatial–temporal compressive sensing," *Opt. Exp.*, vol. 27, no. 22, pp. 31475–31489, 2019.

[61] J. Zhang, T. Xiong, T. Tran, S. Chin, and R. Etienne-Cummings, "Compact all-CMOS spatiotemporal compressive sensing video camera with pixel-wise coded exposure," *Opt. Exp.*, vol. 24, no. 8, pp. 9013–9024, 2016.

[62] Y. Luo and S. Mirabbasi, "Always-on CMOS image sensor pixel design for pixel-wise binary coded exposure," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sep. 2017, pp. 1–4.

[63] N. Sarhangnejad et al., "5.5 dual-tap pipelined-code-memory coded-exposure-pixel CMOS image sensor for multi-exposure single-frame computational imaging," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 102–104.

[64] J. Zhang, Y. Suo, C. Zhao, T. D. Tran, and R. Etienne-Cummings, "CMOS implementation of pixel-wise coded exposure imaging for insect-based sensor node," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Dec. 2015, pp. 1–4.

[65] T. Hirata, H. Murata, H. Matsuda, Y. Tezuka, and S. Tsunai, "7.8 A 1-inch 17Mpixel 1000fps block-controlled coded-exposure back-illuminated stacked CMOS image sensor for computational imaging and adaptive dynamic range control," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 120–122.

[66] J. Gu, Y. Hitomi, T. Mitsunaga, and S. Nayar, "Coded rolling shutter photography: Flexible space-time sampling," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, Mar. 2010, pp. 1–8.

[67] K. Jo, M. Gupta, and S. K. Nayar, "DisCo: Display-camera communication using rolling shutter sensors," *ACM Trans. Graph.*, vol. 35, no. 5, pp. 1–13, 2016.

[68] S. Su and W. Heidrich, "Rolling shutter motion deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1529–1537.

[69] G. Woo, A. Lippman, and R. Raskar, "VRCodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Nov. 2012, pp. 59–64.

[70] N. Sarhangnejad, H. Lee, N. Katic, M. O'Toole, K. Kutulakos, and R. Genov, "CMOS image sensor architecture for primal-dual coding," in *Proc. Int. Image Sensor Workshop (IISW)*, 2017, pp. 1–4.

[71] M. O'Toole and K. N. Kutulakos, "Visualizing light transport phenomena with a primal-dual coding video camera," in *Proc. ACM SIGGRAPH Emerg. Technol.*, p. 1, 2014.

[72] R. Panicacci, S. E. Kemeny, L. H. Matthies, B. Pain, and E. R. Fossum, "Programmable multiresolution CMOS active pixel sensor," in *Proc. SPIE*, vol. 2654, 1996, pp. 72–79.

[73] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh, "CMUcam3: An open programmable embedded vision sensor," in *Proc. Int. Conf. Intell. Robots Syst.*, 2007, pp. 1–34.

[74] M. Gupta, A. Agrawal, A. Veeraraghavan, and S. G. Narasimhan, "Flexible voxels for motion-aware videography," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2010, pp. 100–114.

[75] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan, "Glimpse: A programmable early-discard camera architecture for continuous mobile vision," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2017, pp. 292–305.

[76] J.-E. Park, D.-H. Lim, and D.-K. Jeong, "A reconfigurable 40-to-67 dB SNR, 50-to-6400 Hz frame-rate, column-parallel readout IC for capacitive touch-screen panels," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2305–2318, Oct. 2014.

[77] J. Choi, S.-W. Han, S.-J. Kim, S.-I. Chang, and E. Yoon, "A spatial–temporal multi-resolution CMOS image sensor with adaptive frame rates for moving objects in the region-of-interest," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2007, pp. 502–618.

[78] Y. Inoue, T. Ono, and K. Inoue, "Adaptive frame-rate optimization for energy-efficient object tracking," in *Proc. Int. Conf. Image Process. Comput. Vis., Pattern Recognit. (IPCV)*, 2016, p. 158.

[79] Y. Inoue, T. Ono, and K. Inouer, "Situation-based dynamic frame-rate control for on-line object tracking," in *Proc. Int. Jpn.-Afr. Conf. Electron., Commun. Computations (JAC-ECC)*, 2018, pp. 119–122.

[80] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Feb. 2019, pp. 6202–6211.

[81] P. Heckbert, "Color image quantization for frame buffer display," *ACM Siggraph Comput. Graph.*, vol. 16, no. 3, pp. 297–307, 1982.

[82] P. Scheunders, "A genetic C-means clustering algorithm applied to color image quantization," *Pattern Recognit.*, vol. 30, no. 6, pp. 859–866, 1997.

[83] Y. Deng, C. Kenney, M. S. Moore, and B. Manjunath, "Peer group filtering and perceptual color image quantization," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 4, Aug. 1999, pp. 21–24.

[84] C. Shoushun, A. Bermak, W. Yan, and D. Martinez, "Adaptive-quantization digital image sensor for low-power image compression," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 1, pp. 13–25, Jan. 2007.

[85] S. Chen, A. Bermak, and Y. Wang, "A CMOS image sensor with on-chip image compression based on predictive boundary adaptation and memoryless QTD algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 4, pp. 538–547, Apr. 2011.

[86] O. Christie, J. Rego, and S. Jayasuriya, "Analyzing sensor quantization of raw images for visual SLAM," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 246–250.

[87] Y. Wang, A. Bermak, A. Bouzerdoum, and B. Ng, "FPGA implementation of a predictive vector quantization image compression algorithm for image sensor applications," in *Proc. 4th IEEE Int. Symp. Electron. Design, Test Appl.*, Jan. 2008, pp. 431–434.

[88] K. Wahid, S.-B. Ko, and D. Teng, "Efficient hardware implementation of an image compressor for wireless capsule endoscopy applications," in *Proc. IEEE Int. Joint Conf. Neural Netw. IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 2761–2765.

[89] F. Tang, D. G. Chen, B. Wang, and A. Bermak, "Low-power CMOS image sensor based on column-parallel single-slope/SAR quantization scheme," *IEEE Trans. Electron Devices*, vol. 60, no. 8, pp. 2561–2566, Aug. 2013.

[90] Y. Bandoh, G. Qiu, M. Okuda, S. Daly, T. Aach, and O. C. Au, "Recent advances in high dynamic range imaging technology," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2010, pp. 3125–3128.

[91] S. Kavadias, B. Dierickx, D. Scheffer, A. Alaerts, D. Uwaerts, and J. Bogaerts, "A logarithmic response CMOS image sensor with on-chip calibration," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 1146–1152, Aug. 2000.

[92] M. Loose, K. Meier, and J. Schemmel, "A self-calibrating single-chip CMOS camera with logarithmic response," *IEEE J. Solid-State Circuits*, vol. 36, no. 4, pp. 586–596, Apr. 2001.

[93] G. Storm, R. Henderson, J. E. D. Hurwitz, D. Renshaw, K. Findlater, and M. Purcell, "Extended dynamic range from a combined linear-logarithmic CMOS image sensor," *IEEE J. Solid-State Circuits*, vol. 41, no. 9, pp. 2095–2106, Sep. 2006.

[94] H. Y. Cheng, B. Choubey, and S. Collins, "A high-dynamic-range integrating pixel with an adaptive logarithmic response," *IEEE Photon. Technol. Lett.*, vol. 19, no. 15, pp. 1169–1171, Aug. 2007.

[95] H.-Y. Cheng, B. Choubey, and S. Collins, "An integrating wide dynamic-range image sensor with a logarithmic response," *IEEE Trans. Electron Devices*, vol. 56, no. 11, pp. 2423–2428, Nov. 2009.

[96] D. Kim and M. Song, "An enhanced dynamic-range CMOS image sensor using a digital logarithmic single-slope ADC," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 10, pp. 653–657, Oct. 2012.

[97] M. Bae, B.-S. Choi, S.-H. Jo, H.-H. Lee, P. Choi, and J.-K. Shin, "A linear-logarithmic CMOS image sensor with adjustable dynamic range," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5222–5226, Jul. 2016.

[98] M. Vatteroni, D. Covi, and A. Sartori, "A linear-logarithmic CMOS pixel for high dynamic range behavior with fixed-pattern-noise correction and tunable responsivity," in *Proc. SENSORS*, Oct. 2008, pp. 930–933.

[99] M. Vatteroni, P. Valdastri, A. Sartori, A. Menciassi, and P. Dario, "Linear–logarithmic CMOS pixel with tunable dynamic range," *IEEE Trans. Electron Devices*, vol. 58, no. 4, pp. 1108–1115, Apr. 2011.

[100] A. Kitchen, A. Bermak, and A. Bouzerdoum, "A digital pixel sensor array with programmable dynamic range," *IEEE Trans. Electron Devices*, vol. 52, no. 12, pp. 2591–2601, Dec. 2005.

[101] S. Mann and R. Picard, "On being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures," in *Proc. IST Annu. Meeting*, vol. 48, Mar. 1996, pp. 1–7.

[102] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *Proc. ACM SIGGRAPH Classes*. New York, NY, USA: ACM, 2008, p. 31, doi: 10.1145/1401132.1401174.

[103] T. Mitsunaga and S. K. Nayar, "Radiometric self calibration," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 1999, pp. 374–380.

[104] C. A. Metzler, H. Ikoma, Y. Peng, and G. Wetzstein, "Deep optics for single-shot high-dynamic-range imaging," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1375–1385.

[105] C. Aguerrebere, A. Almansa, Y. Gousseau, J. Delon, and P. Musé, "Single shot high dynamic range imaging using piecewise linear estimators," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, May 2014, pp. 1–10.

[106] K. Hirakawa and P. M. Simon, "Single-shot high dynamic range imaging with conventional camera hardware," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1339–1346.

[107] V. G. An and C. Lee, "Single-shot high dynamic range imaging via deep convolutional neural network," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, 2017, pp. 1768–1772.

[108] A. K. Johnson and C. Jiji, "Single shot high dynamic range imaging using histogram separation and exposure fusion," in *Proc. 5th Nat. Conf. Comput. Vis., Pattern Recognit., Image Process. Graph. (NCVPRIPG)*, 2015, pp. 1–4.

[109] A. Bhandari and F. Krahmer, "HDR imaging from quantization noise," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2020, pp. 101–105.

[110] H. Zhao, B. Shi, C. Fernandez-Cull, S.-K. Yeung, and R. Raskar, "Unbounded high dynamic range photography using a modulo camera," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, Jul. 2015, pp. 1–10.

[111] F. Lang, T. Plötz, and S. Roth, "Robust multi-image HDR reconstruction for the modulo camera," in *Proc. German Conf. Pattern Recognit.* Cham, Switzerland: Springer, 2017, pp. 78–89.

[112] C. Zhou et al., "UnModNet: Learning to unwrap a modulo image for high dynamic range imaging," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–12.

[113] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[114] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 490–530, 2005.

[115] F. Heide et al., "FlexISP: A flexible camera image processing framework," *ACM Trans. Graph.*, vol. 33, no. 6, p. 231, 2014, doi: 10.1145/2661229.2661260.

[116] R. E. W. C. Rafael Gonzalez, *Digital Image Processing*, 4th ed. London, U.K.: Pearson, 2017.

[117] E. Schwartz, R. Giryes, and A. M. Bronstein, "DeepISP: Toward learning an end-to-end image processing pipeline," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 912–923, Jan. 2019.

[118] Z. Liang, J. Cai, Z. Cao, and L. Zhang, "CameraNet: A two-stage framework for effective camera ISP learning," *IEEE Trans. Image Process.*, vol. 30, pp. 2248–2262, 2021.

[119] F. Heide, S. Diamond, M. Nießner, J. Ragan-Kelley, W. Heidrich, and G. Wetzstein, "ProxImaL: Efficient image optimization using proximal algorithms," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–15, 2016.

[120] A. Mosleh, A. Sharma, E. Onzon, F. Mannan, N. Robidoux, and F. Heide, "Hardware-in-the-loop end-to-end optimization of camera image processing pipelines," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Feb. 2020, pp. 7529–7538.

[121] E. Tseng et al., "Hyperparameter optimization in black-box image processing using differentiable proxies," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–27, 2019.

[122] M. Buckler, S. Jayasuriya, and A. Sampson, "Reconfiguring the imaging pipeline for computer vision," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 975–984.

[123] W. Zhou, L. Zhang, S. Gao, and X. Lou, "Gradient-based feature extraction from raw Bayer pattern images," *IEEE Trans. Image Process.*, vol. 30, pp. 5122–5137, 2021.

[124] O. Losson and L. Macaire, "CFA local binary patterns for fast illuminant-invariant color texture classification," *J. Real-Time Image Process.*, vol. 10, no. 2, pp. 387–401, Jun. 2015.

[125] A. Aberkane, O. Losson, and L. Macaire, "Edge detection from Bayer color filter array image," *J. Electron. Imag.*, vol. 27, no. 1, 2018, Art. no. 011006.

[126] A. Trifan and A. J. Neves, "On the use of feature descriptors on raw image data," in *Proc. 5th Int. Conf. Pattern Recognit. Appl. Methods*, 2016, pp. 655–662.

[127] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 1–7.

[128] T. S. Borkar and L. J. Karam, "DeepCorrect: Correcting DNN models against image distortions," *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 6022–6034, Dec. 2019.

[129] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "NeRF in the dark: High dynamic range view synthesis from noisy raw images," *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, May 2022, pp. 16190–16199.

[130] A. Omid-Zohoor, C. Young, D. Ta, and B. Murmann, "Toward always-on mobile object detection: Energy versus performance tradeoffs for embedded HOG feature extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 5, pp. 1102–1115, May 2018.

[131] W. Ljungbergh, J. Johnander, C. Petersson, and M. Felsberg, "Raw or cooked? Object detection on RAW images," 2023, *arXiv:2301.08965*.

[132] X. Zhang, L. Zhang, and X. Lou, "A raw image-based end-to-end object detection accelerator using hog features," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 1, pp. 322–333, Jan. 2022.

[133] P. Hansen et al., "ISP4ML: The role of image signal processing in efficient deep learning vision systems," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, 2021, pp. 2438–2445. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICPR48806.2021.9411985

[134] Q. Lu and B. Murmann, "Improving the energy efficiency and robustness of TinyML computer vision using log-gradient input images," 2022, *arXiv:2203.02571*.

[135] A. Omid-Zohoor, D. Ta, and B. Murmann. (2014). *Pascalraw: Raw Image Database for Object Detection*. [Online]. Available: http://purl.stanford.edu/hq050zr7488

[136] Y. Zhang, B. Dong, and F. Heide, "All you need is RAW: Defending against adversarial attacks with camera image pipelines," *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 323–343.

[137] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[138] S. Diamond, V. Sitzmann, F. Julca-Aguilar, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Towards

end-to-end image processing and perception," *ACM Trans. Graph.*, vol. 40, no. 3, pp. 1–15, 2021.

[139] E. Tseng et al., "Differentiable compound optics and processing pipeline optimization for end-to-end camera design," *ACM Trans. Graph.*, vol. 40, no. 2, pp. 1–19, 2021.

[140] J. Chang and G. Wetzstein, "Deep optics for monocular depth estimation and 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2019, pp. 10193–10202.

[141] N. Robidoux, L. E. G. Capel, D.-E. Seo, A. Sharma, F. Ariza, and F. Heide, "End-to-end high dynamic range camera pipeline optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2021, pp. 6297–6307.

[142] G. Wetzstein et al., "Inference in artificial intelligence with deep optics and photonics," *Nature*, vol. 588, no. 7836, pp. 39–47, 2020.

[143] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 1–43, 2022.

[144] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," 2021, *arXiv:2109.05472*.

[145] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[146] P. Warden and D. Situnayake, *TinyML: Machine Learning With Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA, USA: O'Reilly Media, 2019.

[147] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "TinyML: Current progress, research challenges, and future roadmap," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Nov. 2021, pp. 1303–1306.

[148] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Aug. 2009, pp. 248–255.

[149] S. Abu-El-Haija et al., "YouTube-8M: A large-scale video classification benchmark," 2016, *arXiv:1609.08675*.

[150] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2011, pp. 1521–1528.

[151] Z. Liu, T. Lian, J. Farrell, and B. A. Wandell, "Neural network generalization: The impact of camera parameters," *IEEE Access*, vol. 8, pp. 10443–10454, 2020.

[152] 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 *Proc. 26th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2021, pp. 573–586.

[153] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2015, pp. 1–12.

[154] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, 2016, pp. 1–12.

[155] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[156] S. Yin et al., "A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications," in *Proc. Symp. VLSI Circuits*, 2017, pp. C26–C27.

[157] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[158] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss V2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, pp. 292–308, 2019.

[159] L. Deng et al., "Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation," *IEEE J. Solid-State Circuits*, vol. 55, no. 8, pp. 2228–2246, Aug. 2020.

[160] Y. S. Shao et al., "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 14–27.

[161] Y. Zhu, A. Samajdar, M. Mattina, and P. Whatmough, "Euphrates: Algorithm-SoC co-design for low-power mobile continuous vision," 2018, *arXiv:1803.11232*.

[162] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, "EVA^2: Exploiting temporal redundancy in live computer vision," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, 2018, pp. 1–14.

[163] Y. Feng, P. Whatmough, and Y. Zhu, "ASV: Accelerated stereo vision system," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 643–656.

[164] A. Adams et al., "The Frankencamera: An experimental platform for computational photography," in *Proc. SIGGRAPH*, 2010.

[165] Khronos. (2023). *OpenKCam Overview*. [Online]. Available: https://www.khronos.org/openkcam

[166] ximea. *Multiple ROI Cameras*. Accessed: Sep. 2021. [Online]. Available: https://www.ximea.com/support/wiki/allprod/Multiple_ROI

[167] J. Hegarty, R. Daly, Z. DeVito, J. Ragan-Kelley, M. Horowitz, and P. Hanrahan, "Rigel: Flexible multi-rate image processing hardware," *ACM Trans. Graph.*, vol. 35, no. 4, p. 85, 2016.

[168] J. Hegarty et al., "Darkroom: Compiling high-level image processing code into hardware pipelines," *ACM Trans. Graph.*, vol. 33, no. 4, p. 144, Jul. 2014.

[169] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, "Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," *ACM SIGPLAN Notices*, vol. 48, no. 6, pp. 519–530, 2013.

[170] J. Pu et al., "Programming heterogeneous systems from an image processing DSL," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 3, p. 26, 2017.

[171] R. T. Mullapudi, A. Adams, D. Sharlet, J. Ragan-Kelley, and K. Fatahalian, "Automatically scheduling halide image processing pipelines," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–11, 2016.

[172] T.-M. Li, M. Gharbi, A. Adams, F. Durand, and J. Ragan-Kelley, "Differentiable programming for image processing and deep learning in halide," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–13, 2018.

[173] M. B. S. Ahmad, J. Ragan-Kelley, A. Cheung, and S. Kamil, "Automatically translating image processing libraries to halide," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 1–13, 2019.

[174] H. G. Chen et al., "ASP vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 903–912.

[175] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2013, pp. 69–82.

[176] J. E. Farrell, F. Xiao, P. B. Catrysse, and B. A. Wandell, "A simulation tool for evaluating digital camera image quality," *Proc. SPIE*, vol. 5294, pp. 124–131, Dec. 2003.

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

[178] Cruxopen. *Cruxopen/OpenISP: Image Signal Processor*. [Online]. Available: https://github.com/cruxopen/openISP

[179] S. J. Kim, H. T. Lin, Z. Lu, S. Süsstrunk, S. Lin, and M. S. Brown, "A new in-camera imaging model for color computer vision and its application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2289–2302, Dec. 2012.

[180] Y. Xing, Z. Qian, and Q. Chen, "Invertible image signal processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Feb. 2021, pp. 1–12.

[181] R. Nane et al., "A survey and evaluation of FPGA high-level synthesis tools," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 10, pp. 1591–1604, Oct. 2016.

[182] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, "A method to estimate the energy consumption of deep neural networks," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, 2017, pp. 1916–1920.

[183] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5687–5695.

[184] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2019, pp. 1–8.

[185] E. Hecht, *Optics, 5E*. India: Pearson Education, 2002.

[186] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, The Netherlands: Elsevier, 2011.

[187] A. V. Aho and J. D. Ullman, *Principles of Compiler Design*. Reading, MA, USA: Addison-Wesley, 1977.

[188] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Germany: Springer, 2010.

[189] Y. Wu, V. Boominathan, H. Chen, A. Sankaranarayanan, and A. Veeraraghavan, "PhaseCam3D—Learning phase masks for passive single view depth estimation," in *Proc. IEEE Int. Conf. Comput. Photography (ICCP)*, Jun. 2019, pp. 1–12.

[190] G. Gallego et al., "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2020.

[191] J. Ma, S. Masoodian, D. A. Starkey, and E. Fossum, "Photon-number-resolving megapixel image sensor at room temperature without avalanche gain," *Optica*, vol. 4, no. 12, pp. 1474–1481, Dec. 2017. [Online]. Available: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-4-12-1474

[192] S. Ma, S. Gupta, A. C. Ulku, C. Bruschini, E. Charbon, and M. Gupta, "Quanta burst photography," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–79, 2020.