System Solutions Towards High-Precision Visual Computing at Low Power

by

Venkatesh Kodukula

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved June 2023 by the
Graduate Supervisory Committee:

Robert LiKamWa, Chair
Chaitali Chakrabarti
John Brunhaver
Akshay Nambi

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

Efficient visual sensing plays a pivotal role in enabling high-precision applications in augmented reality and low-power Internet of Things (IoT) devices. This dissertation addresses the primary challenges that hinder energy efficiency in visual sensing: the bottleneck of pixel traffic across camera and memory interfaces and the energy-intensive analog readout process in image sensors.

To overcome the bottleneck of pixel traffic, this dissertation proposes a visual sensing pipeline architecture that enables application developers to dynamically adapt the spatial resolution and update rates for specific regions within the scene. By selectively capturing and processing high-resolution frames only where necessary, the system significantly reduces energy consumption associated with memory traffic. This is achieved by encoding only the relevant pixels from the commercial image sensors with standard raster-scan pixel read-out patterns, thus minimizing the data stored in memory. The stored rhythmic pixel region stream is decoded into traditional frame-based representations, enabling seamless integration into existing video pipelines. Moreover, the system includes runtime support that allows flexible specification of the region labels, giving developers fine-grained control over the resolution adaptation process. Experimental evaluations conducted on a Xilinx Field Programmable Gate Array (FPGA) platform demonstrate substantial reductions of 43-64% in interface traffic, while maintaining controllable task accuracy.

In addition to the pixel traffic bottleneck, the dissertation tackles the energy intensive analog readout process in image sensors. To address this, the dissertation proposes aggressive scaling of the analog voltage supplied to the camera. Extensive characterization on off-the-shelf sensors demonstrates that analog voltage scaling can significantly reduce sensor power, albeit at the expense of image quality. To mitigate this trade-off, this research develops a pipeline that allows application developers to adapt the sensor voltage on a frame-by-frame basis. A voltage controller is integrated into the existing Raspberry Pi (RPi) based video streaming pipeline, generating the sensor voltage. On top of that, the system provides a software interface for vision applications to specify the desired voltage levels. Evaluation of the system across a range of voltage scaling policies on popular vision tasks demonstrates that the technique can deliver up to 73% sensor power savings while maintaining reasonable task fidelity.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to Dr. Robert LiKamWa for his exceptional mentorship and unwavering guidance throughout my challenging yet rewarding graduate school journey. His rigorous training and insightful feedback have significantly contributed to my academic growth and professional development.

I am immensely grateful to my teammates, particularly Mason Manetta, for his invaluable collaboration and contributions to the Squint project. Additionally, I would like to extend my appreciation to Alexander Shearer, Van Nguyen, and Srinivas Lingtula for their involvement in the Rhythmic Pixel Regions project. Together, we have achieved significant milestones and advancements in our research.

I would also like to acknowledge the positive and enjoyable work environment fostered by my lab mates at Meteor Studio. Their camaraderie and engaging conversations have made my time there truly memorable.

I am thankful to the members of my Ph.D. committee, Dr. Chaitali Chakrabarti, Dr. John Brunhaver, and Dr. Akshay Nambi, for their valuable insights and feedback.

My heartfelt appreciation goes out to my friends and roommates for making my social life outside the lab vibrant and fulfilling.

I would like to extend my deepest gratitude to my parents, Latha Kodukula and Nageswararao Kodukula, for their unwavering support and for encouraging me to pursue my dreams since childhood. I am deeply grateful to my sister, Vandana Kodukula, for believing in me and for supporting my decision to pursue Ph.D.

Finally, I would like to acknowledge the School of Electrical, Energy, and Computer Engineering, the School of Arts and Media Engineering at Arizona State University, and the National Science Foundation for their academic and financial support, which has been instrumental in enabling me to pursue and complete my graduate studies.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

2 RHYTHMIC PIXEL REGIONS: MULTI-RESOLUTION VISUAL SENSING SYSTEM TOWARDS HIGH-PRECISION VISUAL COMPUTING AT LOW POWER 5

2.1 Introduction 6

2.2 Background and Related Work 11

2.3 Visual Computing with Rhythmic Pixel Regions 16

2.3.1 Developer-Specified Region Labels 17

2.3.2 Encoded Frame 17

2.3.3 Metadata: Per-Row Offset, Encoding Mask (EncMask) 18

2.3.4 A Case Study Around ORB-SLAM 19

2.4 Design 21

2.4.1 Rhythmic Pixel Encoder Architecture 22

2.4.2 Rhythmic Pixel Decoder Architecture 25

2.4.3 Developer Support for Rhythmic Pixel Regions 28

2.5 Implementation 32

2.5.1 FPGA-Based Encoder and Decoder Integration 32

2.5.2 Runtime 33

2.5.3 Workloads 34

2.6 Evaluation 39

2.6.1 The Use of Rhythmic Pixel Regions Is Flexible 39
2.6.2 The Use of Rhythmic Pixel Regions Is Memory Friendly... 40
2.6.3 The Hardware Extensions Are Lightweight and Scalable ... 42
2.7 Future Directions.. 44
3 SQUINT: A FRAMEWORK FOR DYNAMIC VOLTAGE SCALING OF IMAGE SENSORS TOWARDS LOW POWER IOT VISION 47
3.1 Introduction 48
3.2 Background and Motivation.................................. 52
3.2.1 Analog Readout Is The Bottleneck for Sensor Energy-Efficiency 52
3.2.2 Effects of Analog Voltage Variations on Image Sensor Circuitry 53
3.2.3 Analog Voltage Scaling for Energy-Efficiency 54
3.2.4 Motivational Observations 61
3.3 Design.. 62
3.3.1 Programmable Voltage Controller....................... 64
3.3.2 Developer Support.. 65
3.4 Implementation 69
3.4.1 RPi-based System Integration 69
3.4.2 Workloads 70
3.5 Evaluation 76
3.5.1 The Usage of Squint Is Flexible, Seamless, and Performant 76
3.5.2 The Usage of Squint Is Energy-Efficient 79
3.5.3 The Hardware and Software extensions Are Lightweight ... 81
3.6 Related Work 82

CHAPTER Page

3.7 Future Directions 85

4 CONCLUSION 87

BIBLIOGRAPHY 88

APPENDIX

A FRAME STACKS AND ENERGY MODEL 96

LIST OF TABLES

Table Page

1. Opportunities for Rhythmic Pixel Regions 9
2. System Components in the Video Pipeline 32
3. Vision Tasks and Benchmarks 34
4. Observed Statistics of Task and Benchmark 38
5. Resource Utilization for Different Encoder Designs 43
6. Characterized Image Sensors. Sensors Can Be Undervolted Way Beyond the Voltage Specification Mentioned in Their Datasheets. 54
7. System Components in the IoT Streaming Pipeline 69
8. Energy-Per-Pixel of Various Components in the Vision Pipeline. As Is Widely Acknowledged, Communication Cost Is At Least Three Orders of Magnitude More Than Compute Cost. 98

LIST OF FIGURES

Figure Page

1. Traditional Frame-Based Computing Captures and Processes Entire Frames. ROI-Based Computing Samples Regions of Interest, but at Uniform Spatial and Temporal Resolution. With Rhythmic Pixel Regions, Different Regions Are Captured at Different Spatio-Temporal Resolutions. 6
2. The Process of Encoding and Decoding. The Encoder Packs Pixels Within the Regions from the Original Image, Maintaining Raster-Scan Order. The Decoder Reconstructs Data from the Encoded Frame, Per-Row Offsets, and Encoding Mask. 16
3. For ORB-SLAM, Rhythmic Pixel Regions Can Discard Irrelevant Pixels Early in the Pipeline for Memory Efficiency, While Preserving Relevant Regions at Sufficient Resolutions for Accuracy. 18
4. System Design. The Rhythmic Encoder Decimates the Incoming Pixel Stream from Camera and Encodes Only Region-Specified Pixels into Memory. The Rhythmic Decoder Decodes the Pixels for Use with the Vision Algorithms. 22
5. Encoder Intercepts the Incoming Pixel Stream and Only Forwards Pixels That Match the Stride and Skip Specifications of Any Region. 24
6. Decoder Fulfills Pixel Requests from the Vision App in Two Steps. (a) The Pixel MMU Performs Address Translation to Fetch the Right Set of Pixel Regions from Encoded Frame. (B) The FIFO Sampling Unit Reconstructs the Original Pixel Regions from Encoded Pixel Regions and Metadata. 26

Figure Page

7. With Rhythmic Pixel Regions, Policies Can Define How Different Regions in the Image Can Be Captured at Non-Uniform Spatial and Temporal Resolutions. In This Illustrated Policy, Regions Shaded in Black Are Not Sampled. To Track Objects Entering/Leaving the Scene, the System Performs a Full-Frame Capture on a 'Cycle Length' Periodicity. 30
8. Rhythmic Pixel Regions Reduces Pixel Memory Traffic by Generating Sparser Pixel Streams and Reduces the Memory Footprint by Generating Smaller Frame Buffers. The Reduction Is More with Higher Cycle Lengths. 36
9. There Is a Trade-Off Between Cycle Length and Task Accuracy. In V-SLAM, We Observe High Standard Deviation. This Indicates Future Opportunity to Adaptively Reduce Cycle Length to Improve Accuracy for Scenes with High Motion. 36
10. Existing Systems Supply a Fixed Voltage to the Image Sensor Regardless of the Frame. Squint Adapts the Camera Supply Voltage per Frame for Energy-Efficiency. 48
11. Analog Voltage Variation Affects Pixel and ADC Circuits. A Lower Voltage (VDD) for Pixels Would Affect Transistor Settling Times, Resulting in Noise. A Lower Reference Voltage (VREF) for ADC Would Decrease the Dynamic Range, Resulting in Reduced Contrast. 52
12. Sensor Power Significantly Reduces by Lowering Its Supply Voltage. The Flat Portion Above 2.4 V Is Due to Strong Voltage Regulation Inside the Sensor. 56

Figure Page

13. Sensor's Brightness and Noise Levels Increase with Lower Analog Voltage. Increased Brightness Is Due to Decreased ADC Output Swing and Increased Noise Due to Elevated Shot Noise Levels. 57
14. Images Captured at Different Sensor Voltages and Their Histograms. Low Voltage Image Is Brighter and Grainier Than Its High Voltage Counterpart. This Is Also Reflected in the Shift in Mean and Variance Width in the Histogram. Images Appear More Greenish Because They Are Captured RAW Turning Off White Balance. 57
15. Brighter and Noisier Images Do Not Significantly Affect Task Fidelity of Vision Applications in Well-Lit Scenes. 58
16. High Fidelity Is Still Needed for Precise Object Detection in Challenging Scenarios. Low Fidelity Would Suffice Otherwise. 59
17. Phases of Squint. During Calibration, Squint Sweeps Voltage Supplied to the Camera to Determine the Undervolting Limit as Well as to Profile Noise, and This Information Is Stored in a Configuration File. During Execution, the System Uses the Configuration File to Translate Developer's High-Level Fidelity Input to Actual Low-Level Camera Voltage Using Standard Camera System Stack. 59
18. Different System Incarnations for Squint. In Server-Based, Vision Apps Running on Server Issue Voltage Request to the Voltage Controller, Whereas in Edge-Based, Vision Apps Run on the Edge Device Itself and Directly Issue Requests to the Voltage Controller. 63

Figure Page

19. Voltage Controller Comprises a DAC to Translate Digital Voltage Requests into Corresponding Analog Voltage, and a Voltage Buffer to Generate Sufficient Current Flow for Reliable Camera Operation. 65
20. Timing Different Voltages in Tandem with Camera Capture Schedule Leads to Different Fidelity Patterns Across the Frames and Within a Frame. 66
21. Our System Setup 69
22. Bright Light Voltage Schedules Generated by Different Policies for Entrance Detection Dataset. 73
23. Bright Light Voltage Schedules Generated by Different Policies for Person Tracking Dataset. 73
24. People Detection Task 76
25. Camera Pose Detection Task. 78
26. Power Trace for Different Sensor Voltage Modes. 80
27. Task: Visual SLAM; Benchmark: TUM Freiburg1-Xyz 97
28. Task: Visual SLAM; Benchmark: TUM Freiburg1-Floor 97
29. Task: Visual SLAM; Benchmark: TUM Freiburg2-360-Kidnap-Secret 97
30. Task: Human Pose Estimation; Benchmark: PoseTrack 2017 024575-Mpii . 97
31. Task: Human Pose Estimation; Benchmark: PoseTrack 2017 015301-Mpii . 98
32. Task: Face Detection; Benchmark: Chokepoint Dataset P2E-S5 98

Chapter 1

INTRODUCTION

The rapid advancement of visual computing systems has revolutionized the way we interact with the world through cameras, enabling a wide range of applications in the mobile devices, e.g., Oculus Quest, and Internet of Things (IoT). These systems have the remarkable ability to detect faces, understand spatial scene geometry, capture high-resolution images and videos, and provide security through video surveillance. With the continuous improvement of image sensor technology, such as higher resolutions and faster frame rates, the precision and capabilities of visual systems have significantly improved, opening up new possibilities for augmented reality, navigation, and other visually intensive applications.

While higher resolutions and faster frame rates enable high precision visual computing, they also pose challenges to visual systems on mobile and IoT devices in terms of energy-efficiency. Specifically, higher resolution captures result in high pixel traffic across memory interfaces, which hampers energy-efficiency. Additionally, higher resolution captures require more sensor power due to the costly analog readout process. To mitigate power consumption, existing systems often reduce the spatial resolution and frame rate of image capture, leading to energy savings but sacrificing visual precision and task accuracy. This tradeoff between power consumption and task fidelity highlights the need for innovative approaches that can enhance the capabilities of continuous mobile vision systems while optimizing energy efficiency.

This dissertation addresses the energy-efficiency challenge by proposing two novel approaches: Rhythmic Pixel Regions and Squint Imaging. These approaches aim to trasnform visual computing systems by offering improved precision, energy efficiency, and task adaptability, while taking into account the unique characteristics of different visual features and the power limitations of mobile and IoT devices.

The first approach, Rhythmic Pixel Regions, challenges the traditional paradigm of capturing and processing entire image frame streams at uniform spatial resolutions and frame rates. Instead, it introduces the concept of region-specific spatiotemporal resolutions, allowing for fine-grained configurability based on the properties of visual features. By leveraging encoded data representations, this approach enables the simultaneous capture of hundreds of rhythmic pixel regions, each with its own independently defined resolution. This selective configurability empowers developers to allocate higher spatiotemporal resolution to regions requiring precise visual features, such as augmented reality overlays or facial recognition, while allocating lower resolution to other parts of the frame. By dynamically adjusting the resolution based on task requirements, the rhythmic pixel region architecture eliminates the wasteful processing and memory traffic associated with unproductive pixels, leading to significant energy savings without compromising overall task accuracy.

The second approach, Squint, focuses on the dynamic voltage scaling (DVS) of image sensors to achieve energy savings. Image sensors are known to consume a significant amount of power in visual computing systems, and DVS allows for the adaptive variation of the analog voltage supplied to the sensor on a frame-by-frame basis. The Squint framework addresses the challenge of maintaining imaging fidelity in the face of voltage scaling by thoroughly characterizing the energy and fidelity implications on commercial off-the-shelf image sensors. This characterization reveals that, in most situations, the degradation in imaging fidelity caused by aggressive voltage scaling does not significantly impact the accuracy of modern neural network-based vision workloads. However, precise scene feature detection, especially in challenging conditions such as low-light or crowded scenes, still benefits from higher fidelity. To enable dynamic voltage scaling, the Squint framework introduces a lightweight and fully programmable voltage controller hardware interface, complemented by a runtime system that seamlessly integrates with vision applications. This integration empowers application developers to specify voltage schedules directly, allowing for precise control over power consumption while balancing imaging fidelity requirements.

By combining the Rhythmic Pixel Regions based architecture and the Squint framework, this dissertation presents a comprehensive solution to enhance the capabilities and energy efficiency of visual computing systems. The Rhythmic Pixel Regions architecture enable developers to finely tune spatiotemporal resolutions based on the specific requirements of different visual features, offering an unprecedented level of task adaptability while optimizing power consumption. Meanwhile, the Squint framework provides a dynamic voltage scaling mechanism for image sensors, enabling significant power savings without compromising overall imaging fidelity.

Through extensive evaluations, the proposed approaches have demonstrated remarkable improvements in energy efficiency, extending the battery life of mobile and IoT devices. By striking a delicate balance between task accuracy, power consumption, and imaging fidelity, these approaches pave the way for advanced mobile and IoT vision applications with prolonged operation times.

In summary, this dissertation introduces two groundbreaking approaches to address the challenges of energy efficiency and imaging fidelity in visual computing systems.

- Rhythmic Pixel Regions offers a new paradigm for selective spatiotemporal resolution configurations for AR/VR use-cases

- Squint enables dynamic voltage scaling in image sensors to leverage image quality - power trade-offs for IoT and XR use-cases

The rest of the dissertation is organized as follows. §2 presents Rhythmic Pixel Regions, §3 presents Squint framework, and §4 concludes this work.

Chapter 2

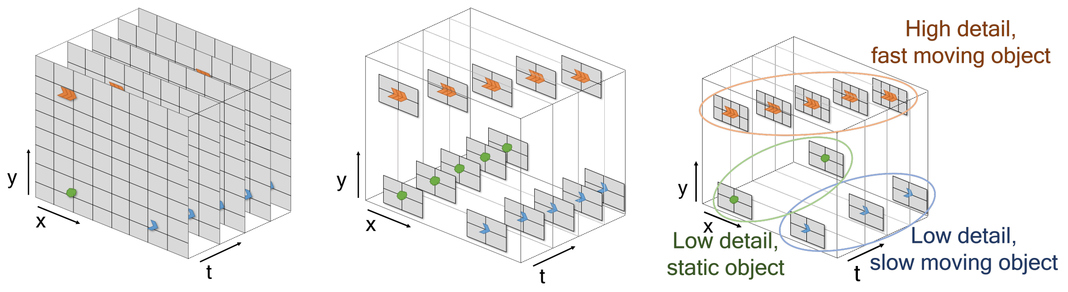
RHYTHMIC PIXEL REGIONS: MULTI-RESOLUTION VISUAL SENSING SYSTEM TOWARDS HIGH-PRECISION VISUAL COMPUTING AT LOW POWER

High spatiotemporal resolution can offer high precision for vision applications, which is particularly useful to capture the nuances of visual features, such as for augmented reality. Unfortunately, capturing and processing high spatiotemporal visual frames generates energy-expensive memory traffic. On the other hand, low resolution frames can reduce pixel memory throughput, but reduce also the opportunities of high-precision visual sensing. However, our intuition is that not all parts of the scene need to be captured at a uniform resolution. Selectively and opportunistically reducing resolution for different regions of image frames can yield high-precision visual computing at energy-efficient memory data rates.

To this end, we develop a visual sensing pipeline architecture that flexibly allows application developers to dynamically adapt the spatial resolution and update rate of different "rhythmic pixel regions" in the scene. We develop a system that ingests pixel streams from commercial image sensors with their standard raster-scan pixel read-out patterns, but only encodes relevant pixels prior to storing them in the memory. We also present streaming hardware to decode the stored rhythmic pixel region stream into traditional frame-based representations to feed into standard computer vision algorithms. We integrate our encoding and decoding hardware modules into existing video pipelines. On top of this, we develop runtime support allowing developers to flexibly specify the region labels. Evaluating our system on a Xilinx FPGA platform over three vision workloads shows 43 – 64% reduction in interface traffic and memory footprint, while providing controllable task accuracy.

2.1 Introduction

Through the lens of their cameras, mobile devices can visually observe a user's environment for detecting faces, understanding spatial scene geometry, and capturing images and videos. This has yielded a wide range of benefits, especially as image sensors have grown to support increasingly higher resolutions and frame rates. With such precision, it is now possible to position augmented reality (AR) overlays over users' faces or spatial living environments for social entertainment. Virtual AR media can also annotate physical environmental surfaces, including for navigational guidance for walking directions Google (2019). Wirelessly connected camera devices on the Internet-of-Things can provide home security, such as through video doorbells.



(a) Frame-based (b) ROI-based (c) Rhythmic pixels

Figure 1: Traditional Frame-Based Computing Captures and Processes Entire Frames. ROI-Based Computing Samples Regions of Interest, but at Uniform Spatial and Temporal Resolution. With Rhythmic Pixel Regions, Different Regions Are Captured at Different Spatio-Temporal Resolutions.

Unfortunately, visual systems on mobile systems are limited in their spatial precision, computational performance, and energy efficiency while performing continuous visual tasks. Mobile systems are constrained by their small form factors with limited battery sizes and heat management requirements. To reduce the power consumption, recently proposed systems reduce the spatial resolution and frame rate of image capture LiKamWa *et al.* (2013); Hu *et al.* (2018) to receive commensurate energy savings, especially by reducing the memory traffic of the DRAM-based frame buffers. Thus, resolution provides a tradeoff mechanism to dynamically configure systems for low power consumption or high visual task fidelity. However, downscaling or windowing a frame forces the application to reduce the resolution of the entire frame. Reducing the frame rate similarly reduces the temporal resolution of the entire frame stream. This reduced spatiotemporal resolution across the entire frame stream can lead to suboptimal visual precision.

This work aims to improve the capabilities of continuous mobile vision systems, based on a key insight: *The precision, performance, and efficiency of visual computing systems are limited by the current pattern of capturing and processing entire image frame streams at* ***uniform*** *spatial resolutions and* ***uniform*** *frame rates.* This assumption of frame-based computing (Fig. 1a) presents an inflexibly coarse granularity of tradeoff between task accuracy and energy efficiency. Most natural scenes do not have the same resolution needs across the entire image frame. Precise AR placement requires high spatial resolution for visual features on tracked surfaces, but would suffice with a relatively lower resolution for the rest of the frame. The detection and tracking of faces, hands, and objects could use a higher temporal resolution to capture quick motions, while the rest of a relatively static scene would suffice with a lower frame rate. Such tradeoffs are unavailable with the current model of frame-based computing.

To address this, we present a fundamental shift away from frame-based visual computing and towards ***rhythmic pixel regions*** (Fig. 1c), which we define as neighborhoods of pixels with region-specific spatiotemporal resolutions. Unlike visual computing based on a few Regions-of-Interest (ROIs) Iqbal *et al.* (2020), rhythmic pixel regions leverage encoded data representations that scalably allow for the capture of hundreds of regions, with independently defined spatiotemporal resolutions. By supporting the simultaneous capture of a diversity of rhythmic pixel regions, our visual computing architecture allows the developer to selectively specify regions where higher spatiotemporal resolution is needed and where lower spatiotemporal resolution will suffice (Table 1), e.g., dynamically guided by the properties of the visual features. The fine-grained configurability will allow developers to extend their existing visual computing algorithms and applications for high energy efficiency. This creates the illusion of high spatiotemporal resolution capture at low power consumption by eliminating the wasteful DRAM traffic of unproductive pixels.

To design the rhythmic pixel region abstraction and the architecture to support it, we introduce two hardware sensor data interfaces: an encoder to selectively reduce sets of pixels before they are stored in memory, and a decoder to reconstruct the pixels from memory into traditional frame-based addressing for standard application use. Together, the rhythmic pixel region encoder and decoder work to reduce the significant DRAM traffic of writing and reading visual data, leading to system energy savings. These interfaces, which integrate into the System-on-Chip, support existing and future high-resolution image sensors, allowing for a revolutionary upgrade to evolutionary mass-market-scale image sensors.

To evaluate our system, we design our implementation on a Xilinx ZCU102 FPGA SoC platform. We support various visual workloads, including hardware-

Table 1: Opportunities for Rhythmic Pixel Regions

	Trad. uniform frame-based vision	**Rhythmic pix region-based vision**
Spatial Resolution	If any part of the frame needs to be captured at a high resolution, e.g., to resolve complex texture or distant objects, the entire frame will need to be captured at a high resolution.	Regions with small, detailed, and/or distant features can be captured with the precision of high resolution. Frame regions with large, static, close visual features can be captured with the efficiency of low resolution.
Temporal Resolution	If any part of the frame needs to be captured at a high frame rate, e.g., to track substantial motion, the system will need to capture a sequence of entire frames of pixels at high frame rate.	Regions can be captured at different intervals. The entire frame can be scanned to update spatial understanding at a lower rate, e.g., 1 fps. Regions of moving objects/surfaces can be captured at higher rates, e.g., 60 fps.

accelerated neural network processing for face and object tracking, and OpenCV-based visual simultaneous localization and mapping (V-SLAM). Through our evaluation, we demonstrate the opportunity of rhythmic pixel regions to decrease pixel memory traffic by $43 - 64\%$ while only minimally degrading the visual task accuracy, e.g., only increasing absolute trajectory error of V-SLAM from 43 ± 1.5 mm to 51 ± 0.9 mm in our case study. Through reduction in memory traffic, rhythmic pixel region based techniques can significantly increase the energy-efficiency of battery-backed mobile systems.

In summary, we make the following contributions:

- To the best of our knowledge, we are the first to propose a visual computing paradigm where different parts of the scene are captured at different spatiotemporal resolutions – before the frame enters memory – for overall system energy-efficiency while respecting task needs.
- We develop two *lightweight and scalable* IP blocks – rhythmic pixel encoder and decoder – which decimates the incoming pixel stream while writing to memory and reconstructs the pixel stream on-the-fly while reading from memory.

- We develop a library and runtime to coordinate vision tasks with encoder/decoder operation.
- We augment our architecture and runtime support on top of an existing commercial mobile vision pipeline built around a FPGA platform. We evaluate the augmented architecture on a variety of vision tasks to demonstrate significant reduction in memory traffic with controllable accuracy loss.

2.2 Background and Related Work

A primer on vision pipelines: The ecosystem of visual computing sensors, devices, systems, and algorithms on mobile devices have rapidly evolved to provide high-performance platforms for multitude of vision tasks such as augmented reality on smartphones, tablets, and headsets Höllerer and Feiner (2004).

Image sensors collect digital readings of visual pixels in a frame Richard Szeliski (2010). The sensor sends values over a streaming MIPI interface, which enacts a serial transmission over multiple lanes MIPI Alliance (2020). The MIPI receiver inside the SoC receives the frame information from camera. In the sensor or on the system-on-chip, there is often an image signal processor (ISP) inserted into the visual computing pipeline, performing image improvement operations, e.g., white balance, and format changes, e.g., YUV conversion. Regardless of the placement and operation of the ISP, the visual hardware pipeline eventually writes the frame into DRAM and signals to the operating system that a frame is ready for readout from the memory.

Data movement across the off-chip MIPI and DDR interfaces entails significant energy consumption Ghose *et al.* (2018). While tasks such as AR could significantly benefit from high spatiotemporal resolutions, e.g., 4K at 60 fps, these resolutions generate high datarates across the camera and memory interfaces. For example, the system expends 2.8 nJ to move a pixel Pandiyan and Wu (2014); Ho *et al.* (2001); Raghunathan *et al.* (2003); Magen *et al.* (2004); Meindl *et al.* (2002) across the DDR interface, whereas it expends only 4.6 pJ for performing a multiply-and-accumulate (MAC) operation Hameed *et al.* (2010) around that pixel.

For AR, the software processes the frame through visual computing frameworks, extracting visual features to feed into SLAM algorithms Mur-Artal, Raúl, Montiel, J.

M. M. and Tardós, Juan D. (2015). These algorithms form a spatial understanding of the scene to estimate the pose of the camera. The pose of the virtual camera is precisely updated to the estimated pose of the physical camera. This allows the system to overlay virtual objects over the physical environment, achieving the AR illusion.

Multi-ROI sensors: Many image sensors are capable of selecting a region-of-interest (ROI) for readout ximea (2019). There are also sensors that allow for multiple ROIs to be read out Stemmer Imaging (2019). As region selection is performed at the sensor level, it offers efficiency and speed by reducing sensor readout time. However, there are significant limitations to adopting these sensors for mobile systems. In particular, the expressiveness of sensor-based region selection is limited by the footprint of additional circuitry. For example, in one such sensor, the region selection is limited to 4 regions, regions cannot overlap, and only full resolution and frame rate are available ximea (2019). In contrast, our support for rhythmic pixel regions provides extensive configurability and composability through the inherent scalability of the encoded data representation. This allows a much larger number of regions (hundreds), and grants each region independent resolution and rhythm/interval control. Moreover, we implement region selection in the SoC, which allows any conventional image sensor to employ multi-ROI benefits, including emerging high resolution and high framerate sensors.

Event-driven cameras: Event-driven cameras, also known as dynamic vision sensors, focus their sampling on pixels that change their value Gallego, Guillermo and Delbruck, Tobi and Orchard, Garrick and Bartolozzi, Chiara and Taba, Brian and Censi, Andrea and Leutenegger, Stefan and Davison, Andrew and Conradt, Jörg and Daniilidis, Kostas and others (2019). This allows for a significant reduction in sensor bit rate and allows microsecond time resolution. However, the circuitry is spatially expensive due to per-pixel motion detection module, reducing frame resolution, e.g., to 128 x 128 pixels. This limits the scalability of these sensors to support high resolutions. More fundamentally, the logic of deciding what pixels to read out is limited at hardware design time, disallowing high-level and/or semantic knowledge from governing the pixel selection process. Thus, while our work shares similar motivations and inspirations – reducing data rate for efficiency and performance – we uniquely allow the expressive ability to dynamically use knowledge of visual feature extraction needs to selectively sample pixels as needed.

Image/Video Compression: Decades of work in image processing has gone towards compressing images and videos to reduce the bit rate of storing and transmitting media. Many of these techniques are inspiring to this work. For example, JPEG and other image compression standards reduce information at spatial frequencies with less perceptual priority Richard Szeliski (2010). MPEG-H Part 2 / HEVC / H.265 reduce redundant information by leveraging estimated motion information from frame-to-frame Iaian Richardson (2004). However, such entropy-coding compression techniques require the frame – or multiple copies of the frame – in memory before compression can be done. This incurs the memory overhead of visual computing that rhythmic pixel regions strives to avoid; we perform encoding *before* the frame enters the DRAM. Traditional video codecs also employ sophisticated techniques such as discrete cosine transform (DCT) and motion compensation for data reduction, increasing their design complexity. In contrast, we use simple pixel discard based strategy for data reduction.

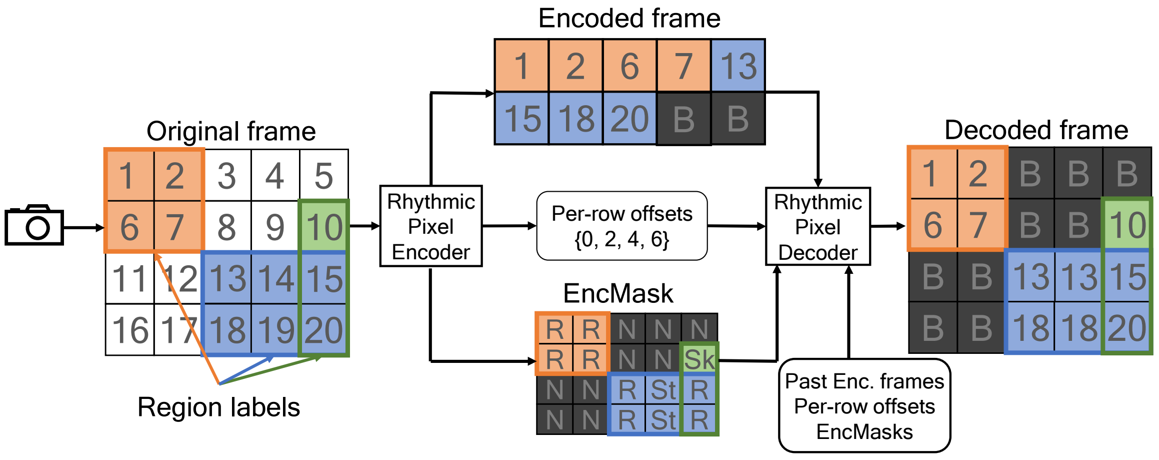
Foveated Rendering: To improve graphical rendering capabilities under limited computing resources, foveated rendering focuses rendering effort where users are likely to notice. Foveated rendering Wikipedia (2020) uses eye tracking to estimate user gaze and renders content near the gaze target at higher spatial resolutions. We apply similar motivation, increasing spatiotemporal resolution where it is needed, but with a distinctly different goal: to only capture necessary visual information. Among other differences, our work involves multiple simultaneous regions, as opposed to the singular region in foveated rendering.

Flexible spatio-temporal sampling algorithms: Computer vision researchers propose different algorithms that modulate different regions in an image with different spatio-temporal resolutions for computational photography and video prediction use-cases. Flexible voxels Gupta *et al.* (2010) proposes an efficient space-time sampling scheme that enables per-pixel temporal modulation and a motion-aware reconstruction scheme for artifact-free videography. More advanced techniques Reddy *et al.* (2011) have been proposed for space-time sampling and interpolation for video compressive sensing and high-speed imaging.

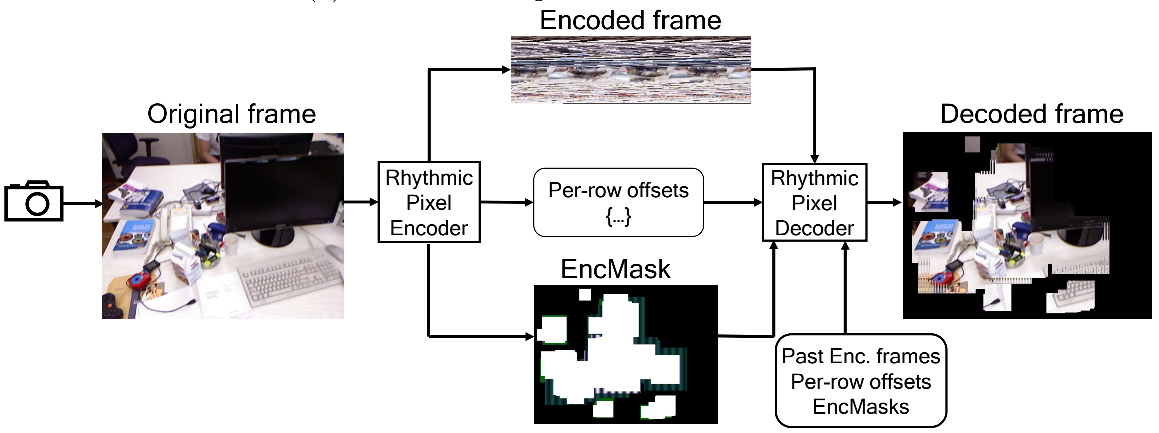
More recently, researchers proposed neural networks Lu *et al.* (2017) to generate high frame rate videos from low frame rate counterparts. These networks analyze spatiotemporal patterns across frames in a video to create interpolated frames. While we use similar spatiotemporal sampling mechanisms, we focus on providing the architecture and runtime support to enable spatiotemporal rhythmic capture on an embedded system.

Energy-efficient visual systems To reduce energy consumption and bandwidth utilization, some systems offload only interesting frames to the cloud, discarding the rest Chen *et al.* (2015). Determining which frames to discard itself is often an expensive task. To this end, other systems Naderiparizi *et al.* (2017) implement a dedicated hardware/software subsystem using an array of gating sensors and imagers.

Other hardware-software co-design techniques reuse computation for energy-efficient visual computing. Euphrates Zhu *et al.* (2018) 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 Buckler *et al.* (2018) exploits temporal redundancy between frames and uses motion compensation techniques to approximate CNN results. ASV Feng *et al.* (2019) applies stereo-specific algorithmic and computational optimizations to increase the performance with minimum error rate. Instead of approximating computation, our work primarily focuses on reducing the memory traffic of sensing by discarding pixels early in the vision pipeline. As such, rhythmic pixel regions can be complementary to the aforementioned visual processing techniques, with approximate computing filling in the visual gap of unsampled pixel data.



(a) Illustrative Representation of Dataflow



(b) Experimental Example Data with ORB-SLAM-Based Regions

Figure 2: The Process of Encoding and Decoding. The Encoder Packs Pixels Within the Regions from the Original Image, Maintaining Raster-Scan Order. The Decoder Reconstructs Data from the Encoded Frame, Per-Row Offsets, and Encoding Mask.

2.3 Visual Computing with Rhythmic Pixel Regions

The overarching concept of rhythmic pixel regions is to encode visual information captured by conventional sensors such that the encoded visual data: (*i*) meets real-time visual requirements specified by application developers, (*ii*) reduces the pixel memory throughput and footprint of pixel stream data into DRAM, and (*iii*) can be reconstructed into visual frames for application usage. Here, we introduce key data structures that specify the region labels, the encoded pixels, and associated meta-data, and how these data structures are used to perform pixel sampling and interpolation to satisfy visual needs. We also illustrate the effect of this new paradigm on a case study based on visual SLAM.

2.3.1 Developer-Specified Region Labels

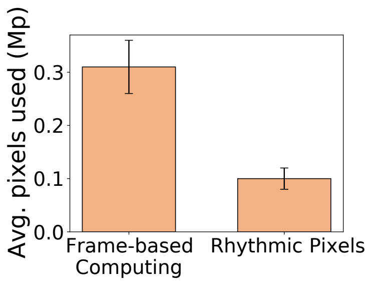
To mark the regions to be captured, as well as their spatiotemporal qualities, the system allows developers to specify a set of region labels (Fig. 2a) to define a capture workload. Each region label includes the following attributes:

- The coordinates of the top-left corner of the region
- The width and height of the region
- The stride resolution of the region, i.e., the density of pixels
- The skip rate, i.e., the time interval of consecutive sampling

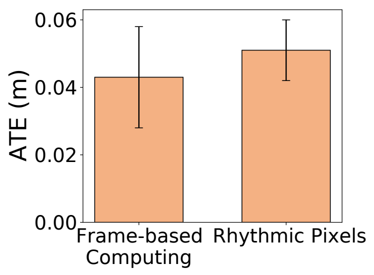
We do not yet support movement, resize, or other temporal dynamics of regions in this work.

2.3.2 Encoded Frame

The encoder uses the region labels to selectively store a reduced set of pixels into memory, forming the encoded frame. Pixels within any of the region labels are stored in original raster-scan order, while omitting any pixels that do not fall within regions, or are not within the rhythm of the stride or skip. This results in a tightly packed encoded frame (Fig. 2a).



(a) Pixels Captured



(b) Absolute Trajectory Error

Figure 3: For ORB-SLAM, Rhythmic Pixel Regions Can Discard Irrelevant Pixels Early in the Pipeline for Memory Efficiency, While Preserving Relevant Regions at Sufficient Resolutions for Accuracy.

Notably, traditional ROI-based computing typically adopts a different memory representation, storing each region of pixels as a grouped sequence in memory. For the small number of regions that ROI-based computing typically supports, this may suffice, but when scaled to hundreds of regions, this creates unfavorable random access patterns into DRAM and/or buffering of large portions of the frame into local SRAM when writing pixels into memory. Furthermore, overlapping regions will duplicate the storage of pixels that appear in multiple regions. Instead, by preserving raster scan order, the rhythmic encoded frame representation instead retains sequential write patterns. This allows for highly efficient scalability to raised number of regions with minimal resource overhead.

2.3.3 Metadata: Per-Row Offset, Encoding Mask (EncMask)

To service pixel requests from the vision applications, the decoder will need to translate pixel addresses in the original frame into pixel offsets in the encoded frame before retrieving the pixel value. However, this would limit decoder scalability, as the complexity of the search operation quickly grows with additional regions.

Thus, instead of using region labels, we propose an alternative method that uses two forms of metadata for the decoder to reconstruct the original pixel stream (Fig. 2a).

First, a per-row offset counts the number of encoded pixels prior to that row in the image. Second, an encoding sequence bitmask (EncMask) helps the decoder to reconstruct a pixel stream without the need for region labels. For each pixel in the original (pre-encoding) frame, the EncMask uses a two-bit status that indicates how a pixel is sampled in space and time:

N (`00`):	Non-regional pixel
St (`01`):	Regional pixel but strided
Sk (`10`):	Regional pixel but temporally skipped
R (`11`):	Regional pixel

Together, the per-row offset and EncMask allow the decoder to find the relevant pixel address in the encoded frame, and access the appropriate values from the current encoded frame or previous encoded frames to decode the pixel. We provide more information about this process in Section 2.4.2.

2.3.4 A Case Study Around ORB-SLAM

ORB-SLAM Mur-Artal, Raúl, Montiel, J. M. M. and Tardós, Juan D. (2015) is a popular real-time V-SLAM algorithm, highly centered around tracking visual “features”, creating a map of localized features, and finding matches to previously mapped features for continual positioning. Visual features – key for scene understanding – are scattered throughout a scene and carry different spatiotemporal needs. While running ORB-SLAM for augmented reality, for example, the feature extractor routinely detects several hundreds of features (e.g., 1500 features in a 1080p frame) at varying distances from the camera. This motivates the need to sample hundreds of regions around hundreds of features with need-specific resolutions. Features can be grouped into a smaller number of regions, but this reduces task accuracy and memory efficiency, as we demonstrate in §3.5.

In the context of rhythmic pixel regions, ORB features will be located in the decoded pixel streams. Regions covering the entire frame can be captured at a slower skip rate to maintain coverage of features as the device moves around the space. In between full captures, detected features in both full and partial captures can guide region label selection for the subsequent frame, defining regions around feature locations and with properties based on feature characteristics. Specifically, our policy uses the feature's "size" attribute to guide the width and height of the region. Along similar lines, it uses other feature attributes such as "octave" for stride and feature movement between frames for temporal rate. The encoder uses these feature-based region labels to selectively store pixels of interest and metadata into memory. The decoder uses these to reconstruct the frames for the algorithm to access and use for ORB-SLAM and further region selection.

We evaluate the efficacy over the TUM dataset Sturm *et al.* (2012) of 480p videos, capturing full frames every 10 frames and feature-based regions for other frames. As shown in Fig. 3, we find that using rhythmic pixel regions can eliminate the memory storage of 66% of the pixels of the original stream, while only increasing absolute trajectory error from 43 $\pm$ 1.5 mm to 51 $\pm$ 0.9 mm. We further explore the applicability of rhythmic pixel regions to visual workloads in our evaluation.

2.4 Design

The rhythmic pixel region architecture centers around the idea of: (*i*) encoding pixel streams to reduce the pixel data stored in memory, and (*ii*) decoding the pixel streams for vision application usage. Here, we outline the design aspirations that guide our architecture. We then describe the design of two hardware units – a rhythmic pixel encoder and a rhythmic pixel decoder – and their integration with the existing mobile SoC vision pipeline, as shown in Fig. 4. We also discuss the software runtime for vision application developers to leverage these hardware extensions through policy specification.

Design aspirations: We set four goals that guide the design of our hardware and software extensions.

- Lightweight: Our encoding/decoding techniques should be lightweight (unlike traditional video codecs) so that they can be designed and integrated with minimal resource overhead.
- Scalable: Our system should scale to support the capture of many regions with minimal resource overhead.
- Memory friendly: Our hardware extensions should perform encoding/decoding tasks with efficient memory access patterns and minimal DRAM traffic
- Flexible: Our runtime should allow developers to flexibly and independently specify resolution and update rate needs on a per-region basis.

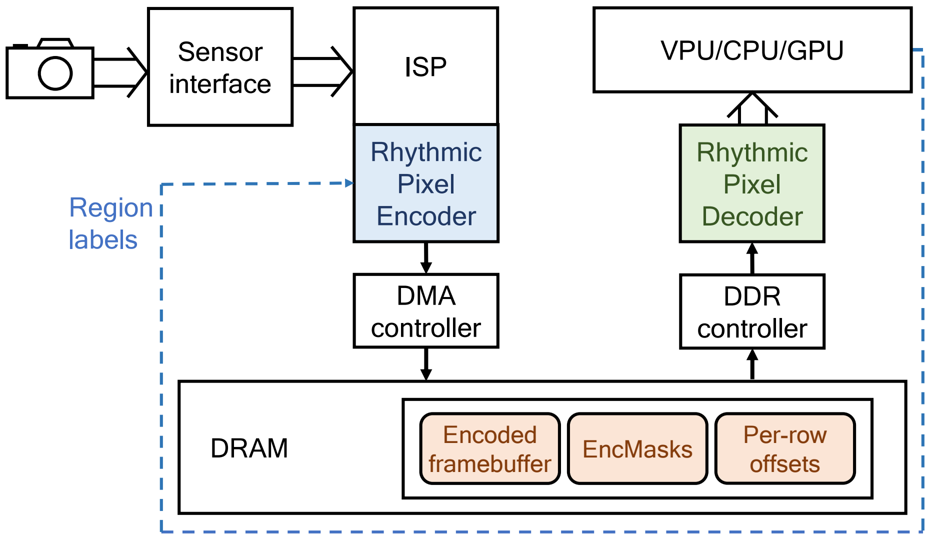


Figure 4: System Design. The Rhythmic Encoder Decimates the Incoming Pixel Stream from Camera and Encodes Only Region-Specified Pixels into Memory. The Rhythmic Decoder Decodes the Pixels for Use with the Vision Algorithms.

2.4.1 Rhythmic Pixel Encoder Architecture

The rhythmic pixel encoder module, shown in Fig. 5, intercepts the incoming pixel stream from an image sensor pipeline and uses developer-specified region labels to encode pixels into an encoded frame. The encoder also generates associated metadata (per-row offset is a count of the number of regional pixels in a row, and EncMask is the output of the region comparison process), which is stored with the encoded frame in DRAM. The app specifies region labels using the designed runtime support in Section 2.4.3.

We design our encoder as a fully streaming based module that avoids the need for partitions to store individually addressed regions. Instead, our encoder directly operates on a dense raster-scan based pixel stream, produces a sparse encoded stream based on the region labels, and writes the encoded stream to the DRAM, all with *on-the-fly* streaming operation.

2.4.1.1 Raster-Scan Optimized Sampling:

The sampling block operates on the pixel stream and decides whether to sample a pixel in space and time based on whether the pixel is in any of the regions or not. A naive approach could sequentially compare a pixel's location against every region label. This would be time consuming and would hinder pipeline performance. Although parallelizing the region label checking process would solve performance issues, we find that this exponentially increases the number of resources to support more regions, limiting encoder scalability.

Instead, our approach is to exploit the raster-scan patterns of the incoming pixel stream to reduce and reuse the work of the region search. The Sequencer keeps track of row and pixel location. For a given row, there is a smaller subset regions that are relevant – where the y-index of the pixel is inside of the y-range of the region and matches the vertical stride. The RoI Selector performs this search space reduction, i.e., converting from the region list to a sublist, once per row. Then, from pixel to pixel, the encoder's Comparison Engine only needs to check whether the x-index of the pixel is inside any of the regions in the sublist and matches the horizontal stride. As the sublist is much smaller than the original list, this design choice substantially reduces the level of region-based parallelism needed for the Comparison Engine, saving hardware complexity.

In addition, we perform further optimizations for the comparison engine based on spatial locality. Within a row, if we find that if a pixel belongs to a region, the Sampler can apply the same comparison result for the next *region width* number of pixels. We also simplify the process of finding relevant regions for a row through

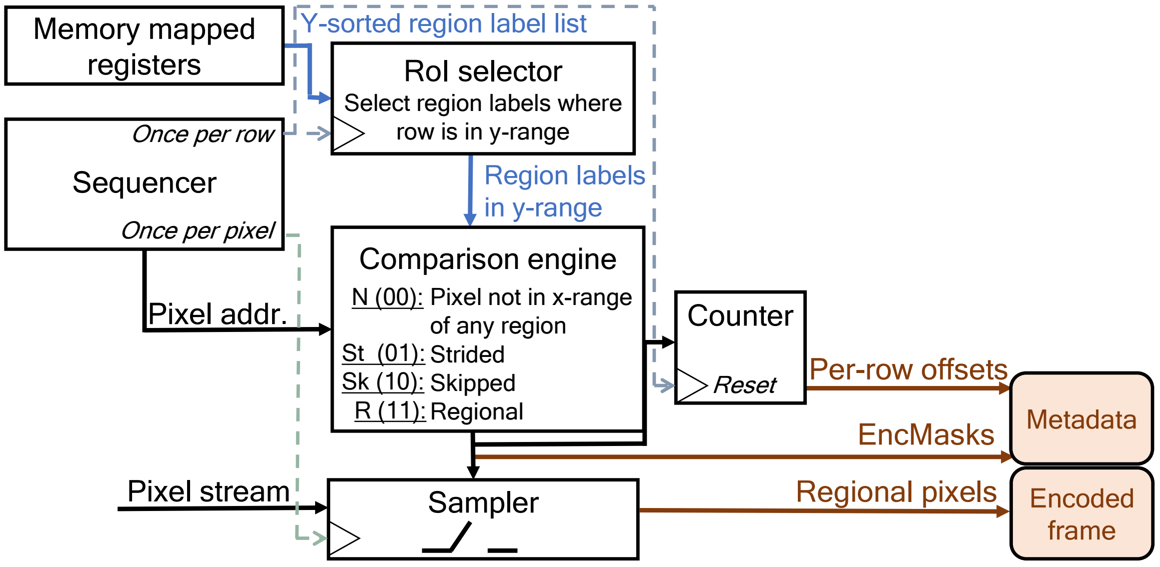


Figure 5: Encoder Intercepts the Incoming Pixel Stream and Only Forwards Pixels That Match the Stride and Skip Specifications of Any Region.

sorting the regions by their y-indices. This can be done by the rhythmic pixel region app runtime.

2.4.1.2 Integration with ISP Output

The placement of the encoder has implications on the efficiency of the visual capture. The most energy-efficient integration of the encoder would be on the sensor itself, reducing pixel traffic over the sensor interface and potentially reducing ISP computations. However, this would need a re-design of the ISP, which conventionally expects pixels in frame-based ordering in its algorithms. Thus, for this work, we instead integrate the encoder at the output of the ISP to seamlessly work with existing commercial ISPs.

As with typical ISP operation, the encoder collects a line of pixels before committing a burst DMA write to a framebuffer in the DRAM for efficient and performant memory transaction. Using a framebuffer allows asynchronous access to frame pixels. In the case of the rhythmic pixel architecture, the framebuffer also allows the system to collect multiple frames of data, which the decoder can use to extrapolate regional pixels over a sequence of frames for regions with temporal skip rate. The metadata (per-row offset and EncMask) of the frame is also stored alongside the framebuffer in DRAM. As the EncMask occupies 2 bits per pixel, we note that it occupies 8% of the original frame data (e.g., 500 KB for a 1080p frame). As explored in our evaluation, this amounts to a minimal memory overhead compared to the memory savings.

2.4.2 Rhythmic Pixel Decoder Architecture

The rhythmic pixel decoder fulfills pixel requests from the vision app, which seek groups of sequential pixels from a decoded framebuffer. While the app forms its request around pixel locations in the decoded address space, the decoder uses the metadata and encoded frames to translate decoded pixel addresses to the DRAM addresses of the encoded frame pixels. This request path is managed by a pixel memory management unit (PMMU), as described below and shown in Fig. 6.

The response path returns the pixel values to the vision application through a FIFO Sampling Unit. If required by spatial stride or temporal skip situations, the FIFO Sampling Unit interpolates encoded data to create decoded pixel values, which it provides to the requesting processor. Otherwise, the decoder returns the pixel value, or a black pixel, if the pixel was not within any of the specified region labels. As described below, the metadata is used to make such determinations.

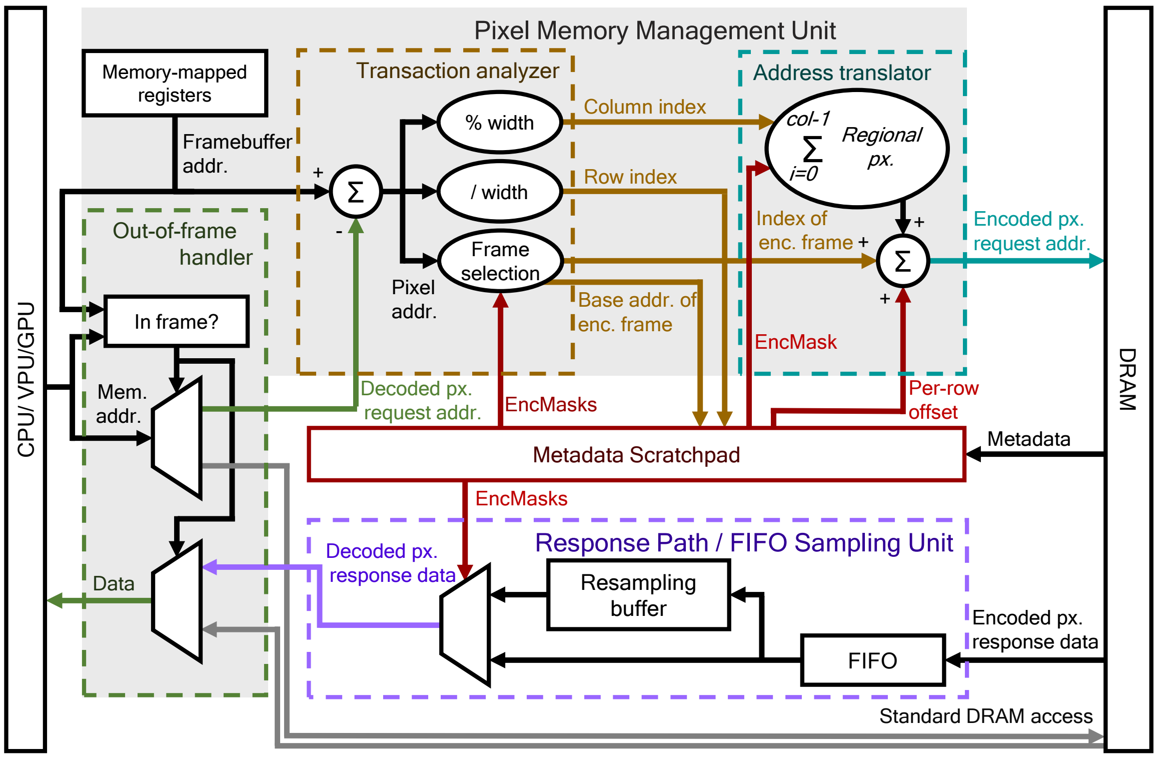


Figure 6: Decoder Fulfills Pixel Requests from the Vision App in Two Steps. (a) The Pixel MMU Performs Address Translation to Fetch the Right Set of Pixel Regions from Encoded Frame. (b) The FIFO Sampling Unit Reconstructs the Original Pixel Regions from Encoded Pixel Regions and Metadata.

2.4.2.1 Pixel Memory Management Unit for Pixel Address Translation:

The PMMU works in the same spirit as a traditional memory management unit (MMU); while MMUs perform virtual to physical address conversion, our PMMU translates pixel transactions from decoded to encoded space. Similar to the exception handler of an MMU, the PMMU's Out-of-Frame Handler examines a memory request and determines if it is a valid pixel request, i.e., if the requested memory address is in the decoded framebuffer address space. The Out-of-Frame Handler forwards the transaction if it's a pixel-based one; otherwise, it will bypass for standard memory access.

Meanwhile, the Metadata Scratchpad loads the per-row offset and EncMask information pertaining to the transaction for the four most recent encoded frames. The Transaction Analyzer analyzes the EncMasks of the transaction and generates different sub-requests based on where the encoded pixels are present. Based on the stride (St) and skip (Sk) values of the EncMask bits, the pixel may be in the most recent encoded frame, or one of the recently stored encoded frames. Thus, similar to a virtual memory request, translation of these pixel requests creates sub-requests that are characterized by a base address (of the encoded frame), offset (row and column), and a tag index of which frame hosts the desired pixels.

These sub-requests are fed to a translator. For intra-frame requests, the base address remains the same, whereas for inter-frame requests, the translator modifies it to the appropriate base address. The per-row offset is read from the metadata. The column offset is the count of the number of full regional pixels from the start of the row until that pixel (The number of "11" entries in the EncMask). The translator sums this information to generate the new encoded pixel request corresponding to each sub-request, which will be sent to DRAM.

2.4.2.2 FIFO Sampling Unit:

A FIFO buffers data packets received from a pixel-based DRAM transaction. To prepare a pixel value to service the original request, the engine either dequeues pixel data from the FIFO, re-samples the previous pixel (in the case of stride), or samples a black pixel, based on the EncMask. The unpacker also relays the RLAST control signal to indicate the last transfer in the transaction for proper handshake with the processing unit for the memory request.

2.4.2.3 Integration with DDR Controller

We integrate the decoder module with the existing DDR controller inside the SoC. By doing so, the decoder can intercept memory traffic coming from any processing element and service requests.

2.4.3 Developer Support for Rhythmic Pixel Regions

From the point of view of the processing unit – CPU, visual processing unit, or GPU – the rhythmic pixel region architecture preserves the addressing scheme of the original frame-based computing through the decoded framebuffer address. This allows fully transparent use of existing software libraries and hardware accelerators, with no modification needed.

We develop runtime support to allow the developers to flexibly specify region labels. This consists of a `RegionLabel` struct and a `SetRegionLabels()` function for developers to set a list of regions. Region label lists can be set on a per-frame basis or persist across frames. A runtime service receives these calls to send the region label list to the encoder.

```
struct RegionLabel {
   int x, y, w, h, stride, skip;
};
SetRegionLabels(list<RegionLabel>);
```

2.4.3.1 Policy-Based Usage of Rhythmic Pixel Regions

Developers can build various policies that autonomously guide the region selection, in the similar spirit to issuing frame configurations with the Frankencamera API Adams *et al.* (2010) or Android's Camera API Android (2019). Policies can incur different system overheads, leading to system trade-offs. A policy should predict region progression with time as well as a region quality requirements to maximize task performance. A feature-based policy (such as that in our case study of Section 2.3.4) can use proxies, such as feature scale and feature displacement to estimate spatial and temporal resolutions of regions. Developers can also introduce improved application-specific proxies with other prediction strategies, e.g., with Kalman filters Iqbal *et al.* (2020).

The process of policy generation and modifying the app around that policy could be cumbersome for an app developer. To reduce the burden, we propose two tiers of developers

Policy Makers: The first tier of developers can specialize in policy development. They can write a wide variety of policies that employ a feature-based approach and/or sophisticated motion-vector based techniques, such as those found in Euphrates or EVA^2 to guide region selection.

Policy Users: The second tier of developers could select a policy directly from a pool based on their app needs.

This dichotomy of policy makers/users follows a typical stratified development paradigm, where high-level developers can simply employ domain-specific libraries (e.g., cuDNN, cuFFT, cuBLAS) that are developed to take advantage of architectural complexity (e.g., with CUDA) by a different set of low-level developers.

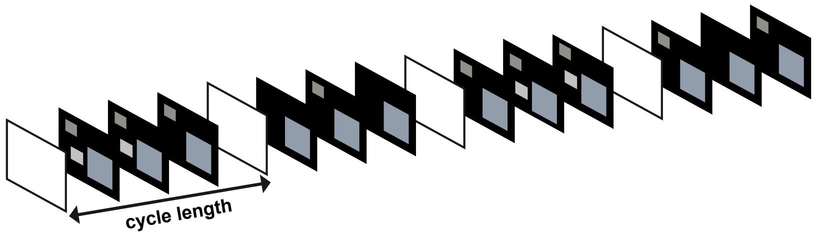


Figure 7: With Rhythmic Pixel Regions, Policies Can Define How Different Regions in the Image Can Be Captured at Non-Uniform Spatial and Temporal Resolutions. In This Illustrated Policy, Regions Shaded in Black Are Not Sampled. To Track Objects Entering/Leaving the Scene, the System Performs a Full-Frame Capture on a 'Cycle Length' Periodicity.

Example policy: As described in the case study, policies can center rhythmic pixel regions around features, ensuring that features are captured at sufficient resolution. We define an example policy (Fig. 7) around the concept of a *cycle length*, which is the number of consecutive frames between two full frame captures. These occasional full frame captures can provide contextual information of the entire scene at a slower temporal rate, while feature-based region tracking in the intermediate frames can provide continual coverage of important regions.

The visual features processed by the app – readily available in memory – can be used to determine the pixel regions for the next subsequent frame, centering around the features. For example, as discussed in §2.3.4, an ORB feature's "size" (or scale) attribute OpenCV (2015) can guide the width and height of its corresponding region, ensuring that the meaningful neighborhood of pixels around the feature is captured (with extra margin to allow for frame-to-frame feature displacement). Furthermore, the "octave"OpenCV (2015) attribute, which describes the texture of the feature, can determine the stride (resolution) parameter for each region. Through the displacement of matched features from frame to frame and measuring the displacement, the system can estimate the movement of a region. The policy can use this feature velocity to determine the temporal rate parameter of pixel regions, sampling fast moving regions more frequently and slow-moving regions less frequently.

We use and adapt this example policy to various visual workloads in our implementation and evaluation. We evaluate different cycle lengths, finding that as the cycle length increases, system efficiency improves, but the errors due to tracking inaccuracy also accumulate, and vice-versa. Cycle length thus becomes an important parameter to govern the tradeoff to meet application needs. The cycle length could also be adaptive, for example, by using the motion in the frame or other semantics to guide the need for more frequent or less frequent full captures. We envision that future policies developed by a wider community of policy makers could substantially improve the opportunities of rhythmic pixel regions.

Table 2: System Components in the Video Pipeline

Component	**Specification**
Camera	Sony IMX274, 4K @ 60 fps
ISP	Demosaic and Gamma correction, 2 Pixels Per Clock
CPU	ARM Cortex-A53 quad-core
GPU	ARM Mali-400 MP2
NPU	Deephi DNN co-processor
DRAM	4-channel LPDDR4, 4 GB, 32-bit

2.5 Implementation

2.5.1 FPGA-Based Encoder and Decoder Integration

Platform: We use Xilinx's reVISION stack platform Xilinx (2018) which implements a end-to-end video pipeline that emulates a standard mobile embedded vision pipeline. As shown in Table 7, we use Sony's IMX274 camera, a mobile class imager used in many smartphones for high fidelity capture, i.e., 4K @ 60fps. For the ISP, we use Xilinx's IP modules for performing demosaicing, gamma correction, and color-space conversion. These ISP blocks operate at a throughput of 2 pixels per clock for real-time performance. The processing subsystem comprises heterogeneous set of computing elements, similar to a mobile SoC. Specifically, the device contains an ARM Cortex-A53 quad-core CPU and an ARM Mali-400 MP2 graphics processing unit (GPU). In addition, we integrate a Deephi DNN co-processor Xilinx (2020b) into the FPGA fabric to emulate a neural network accelerator. Finally, the entire system is provisioned with 4 GB LPDDR4 DRAM in the processing subsystem, which we partially leverage for frame buffer capacity. The fully functional pipeline delivers real-time performance of up to 60 fps for video pass-through and up to 30 fps for certain vision tasks, such as face detection. We build our system around Xilinx's reVISION platform Xilinx (2018), which implements an end-to-end video pipeline with major components shown in Table 7.

Encoder and Decoder: We design our encoder IP module with Vivado HLS as a fully-streaming block with AXI-stream interfaces. In our encoder, input/output buffers are FIFO structures with a depth of 16. We find that this depth is enough to meet the 2 pixel-per-clock performance to match ISP performance and to avoid any pipeline stalls.

We also design our decoder with Vivado HLS, utilizing AXI memory-mapped interfaces on the input and output for integration with controllers and processing units. Our decoder operates within the timing budget without introducing extra latency. Both encoder and decoder functionally work in HLS simulations and the entire video pipeline passes Vivado FPGA post-layout timing. In addition to the hardware decoder, we design a software decoder using C++ and OpenCV. The software decoder runs in real-time for a 1080p video stream.

As Xilinx packages the system DDR controller as a part of its Zynq CPU IP module, we could not integrate the decoder between the DDR controller and the xPU as shown in Fig. 4. Instead, we integrate our decoder as a memory-mapped peripheral slave to Zynq SoC. In this integration, the system passes the pointers to the encoded frame and metadata to the decoder to reconstruct the original frame.

2.5.2 Runtime

Our runtime comprises a standard software stack with a user-space API, a kernel-space driver, and a set of low-level physical registers. We implement region parameters as registers in the encoder/decoder modules inside the SoC. Upon invoking any setter

Table 3: Vision Tasks and Benchmarks

Task	**Algorithm**	**Resolution**	**Benchmark**	**#Frames**
Visual SLAM	ORB-SLAM2 Mur-Artal, Raúl, Montiel, J. M. M. and Tardós, Juan D. (2015)	4K@ 30 fps	In-house dataset	6000
Pose estimation	PoseNet Cao *et al.* (2019)	720p@ 30 fps	PoseTrack 2017	3792
Face detection	RetinaNet 1996scarlet (1996)	SVGA@ 30 fps	ChokePoint dataset	22099

function from the application, the user-space API passes parameters to the kernel-space driver. The driver then writes these parameters to the appropriate registers in the hardware units over an AXI-lite interface.

2.5.3 Workloads

We study three widely-used vision tasks: (i) Visual SLAM, determining camera position with respect to its surroundings while constructing the map of surroundings. (ii) Human pose estimation, tracking person movement. (iii) Face detection, tracking faces over time. For each task, as shown in Table 3, we choose state-of-the-art algorithms with different input, memory, and computational requirements.

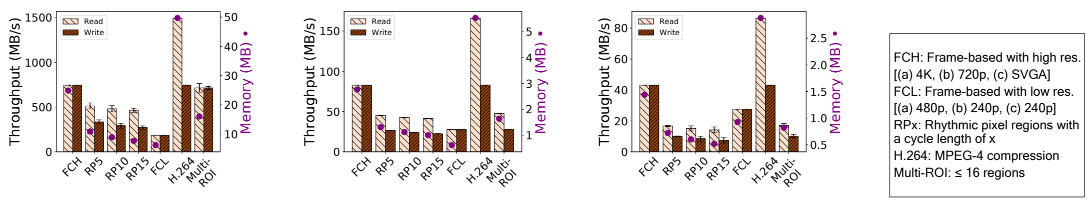
Notably real-time performance of the V-SLAM workload is not attainable on the FPGA's CPU due to the compute-intensive nature of ORB-SLAM2. Therefore, we resort to a simulation-based approach where we run the V-SLAM workload on a desktop computer, generate the region labels, and feed them to the encoder. That said, there are commercial V-SLAM IP cores Synopsys (2020); EETimes (2020) available in the market that offer real-time performance. We plan to integrate them into our FPGA platform for future studies.

Benchmarks: We use popular publicly available benchmarks for each of these tasks to evaluate their accuracy. Each of these datasets comprise videos with different visual attributes, including illumination variation and occlusion that mimic the real-time scenarios in the wild. In addition, these videos cover a wide range of settings, e.g., indoor/outdoor, crowded/dispersed, and fast/slow motion of objects making them realistic candidates for evaluating rhythmic pixel regions.

To evaluate the potential of rhythmic pixel regions on high-precision visual computing, we also evaluate visual SLAM on a 4K dataset. Since there are no ready-made 4K datasets available for visual SLAM, we create a dataset of 6000 frames spanning a total of 7 indoor video sequences with varying user movement. For portability, we use the 4K camera on a Microsoft Azure Kinect DK Microsoft (2020). We use an HTC Vive tracker setup to obtain the ground truth pose. The obtained ground truth pose has an offset compared to the original pose since the tracker is at a different location from the camera. We use the information from Kinect's IMU sensor to correct the offset.

For human pose estimation, we use the famous PoseTrack Max Planck Institute for Informatics, University of Bonn (2018) with 3792 frames across all video sequences. Finally, for face detection, we use the ChokePoint NICTA (2015) dataset which comprise 20 video sequences and 22,099 face images.

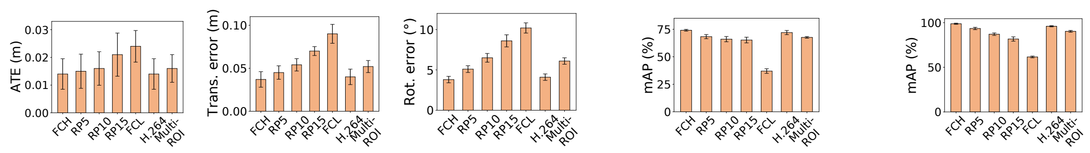
Baselines: We tested the workloads against the following baselines. (a) Frame-based computing: The system captures frames at high resolution (FCH: 4K for V-SLAM) or low resolution (FCL: 1080p for V-SLAM) (b) Rhythmic pixel regions (RPx): The system implements rhythmic pixel regions of cycle length "x" (c) Multi-ROI cameras: The system simulates off-the-shelf multi-ROI cameras (d) H.264: The system performs H.264 video compression with the "Baseline" profile and the "5.2 (2160p60)" level for the codec.



(a) Visual SLAM (b) Human Pose Est (c) Face Detection

Figure 8: Rhythmic Pixel Regions Reduces Pixel Memory Traffic by Generating Sparser Pixel Streams and Reduces the Memory Footprint by Generating Smaller Frame Buffers. The Reduction Is More with Higher Cycle Lengths.

As far as we've seen, commercial multi-ROI cameras only support up to 16 regions, likely due to architectural unscalability. For workloads that use more regions, we combine smaller regions into 16 larger regions through k-means clustering. To accurately represent the capabilities of multi-ROI cameras, we do not implement stride or skip adaptations. As an H.264 compression implementation is inaccessible on our FPGA board, we instead use a codec datasheet to form estimations Xilinx (2020a). As compression needs multiple frames to be stored in the memory, the pixel memory footprint and throughput scale accordingly.



(i) Traj Error (ii) Trans Error (iii) Rot Error (i) mAP (i) mAP

(a) Visual SLAM (b) Human Pose Est (c) Face Detection

Figure 9: There Is a Trade-Off Between Cycle Length and Task Accuracy. In V-SLAM, We Observe High Standard Deviation. This Indicates Future Opportunity to Adaptively Reduce Cycle Length to Improve Accuracy for Scenes with High Motion.

2.5.3.1 Metrics:

Here we discuss about different evaluation metrics.

Task Accuracy: We choose standard accuracy metrics from computer vision literature for all of our tasks. For visual SLAM, we use absolute trajectory error and relative pose error metrics as discussed in §2.3.4. For human pose estimation/face detection, we use the intersection over union (IoU) score as the metric. IoU measures the amount of overlap between the predicted and ground truth bounding boxes. A detection is a true positive (TP) if the IoU score is greater than a certain threshold; otherwise, it is considered as a false positive (FP). Final detection accuracy is the number of true positives among all detections, i.e., TP/(TP + FP), across all the frames, which is known as mean average precision (mAP).

Datarate and Memory Footprint: We build a throughput simulator which takes the region label specification per frame from the application and uses it to generate the memory access patterns of pixel traffic. The simulator counts the number of pixel transactions and directly reports the read/write pixel throughput in bytes/sec. For memory footprint, we measure the size of encoded frame buffers over time.

Overhead: We use Xilinx Vivado Xilinx (2019a) to determine the area and power overhead of our encoder and decoder modules. We use the resource utilization from the post-layout design as a proxy to report the area overhead; For power overhead, we use the numbers from Vivado power analysis tool.

Table 4: Observed Statistics of Task and Benchmark

Task	**Avg. Number of regions**	**Region size**	**Stride**	**Rate**
Visual SLAM	973	Min: 70x70 Max: 230x230	Min: 1 Max: 4	Min: 100 ms Max: 33 ms
Face detection	3	Min: 70x63 Max: 270x228	Min: 1 Max: 2	Min: 67 ms Max: 33 ms
Human Pose estimation	4	Min: 161x248 Max: 324x512	Min: 2 Max: 4	Min: 100 ms Max: 33 ms

2.5.3.2 Policy/Parameter Choices:

We outline our choices below.

Region selection: As outlined in §2.4.3, we use feature characteristics to guide the selection of region labels to evaluate our tasks. For V-SLAM, region size is derived directly from the feature size attribute, while spatial and temporal resolutions are derived from the octave attribute and feature displacement respectively. We use face trajectory for face detection and skeletal pose joints for human pose estimation for determining the regions. Spatial and temporal resolutions are calculated based on the region's size and motion, respectively.

Cycle length: We evaluate the effectiveness of the example cycle-based policy with cycle lengths of CL=5, 10, and 15.

2.6 Evaluation

2.6.1 The Use of Rhythmic Pixel Regions Is Flexible

Our runtime successfully allows apps to express region labels without any restrictions on the number, size, and resolution of the regions, as shown in Table 4. The size of these regions vary based on the semantics, e.g., nearness/farness of a face with respect to the camera. These regions are also sampled at different spatio-temporal resolutions based on their content.

Algorithms/Apps are still reliable: With flexible region specification, the app now deals with only the pixels within the regions as opposed to the pixels in the entire frame. As shown in Fig. 9, we find that apps can still reliably perform their tasks with a slight accuracy loss compared to frame-based computing at high resolutions on uncompressed (FCH) or compressed (H.264) frames. Comparatively, frame-based computing at low resolutions (FCL) performs poorly, with significantly raised errors for all of the visual workloads.

For our workloads, we observe a trade-off between cycle length and accuracy. As shown in Fig. 9, while higher cycle lengths help discard more pixels, they also take a toll on the task accuracy. Moderate cycle lengths, e.g., CL=10, strike a reasonable balance between energy savings and task accuracy.

This raises a question: How much accuracy loss is acceptable? There is no set standard in the vision community, as it depends on the app context. To that end, the flexibility of policy choice allows developers to heuristically set accuracy expectations for their specific apps. Our evaluated workloads result in roughly 5% accuracy loss for moderate cycle lengths, i.e., CL=10. Future investigation into region selection policies and adapted algorithms could further improve accuracy.

We can also observe a higher standard deviation for raised cycle lengths, especially for V-SLAM. This indicates scene-based variability in accuracy loss. We analyze the scenes to find that the scenes reporting low accuracy loss are fairly static in nature, whereas the scenes with high accuracy loss have rapid scene motion. Other scenes in our benchmark resulted in an average accuracy loss. Ideally, in future integrations, this information could be leveraged, e.g., using accelerometer data and/or scene knowledge to guide the region selection policy.

2.6.2 The Use of Rhythmic Pixel Regions Is Memory Friendly

Discarding pixels relieves memory interfaces: Rhythmic pixel regions substantially reduce the data traffic across DDR interfaces, as shown in Fig. 8. With higher cycle lengths, the system discards more pixels, leading to further reduction in memory bandwidth. Specifically, we find that memory traffic decreases by 5-10% with every 5 step increase in cycle length.

Notably, we find that the work saving techniques such as search-space reduction in our encoder works for a broad spectrum of videos Our evaluated datasets, which contain different videos captured in the wild, comprise images with regions spread across the entire image as well as images with regions confined to a few areas within the image. The encoder saves work in both cases. In the former case, our encoder saves work by reducing the number of region comparisons for each row. In the latter case, the encoder saves work by skipping region comparison entirely for those rows where there are no regions.

The pixel memory throughput for the multi-ROI baseline is larger than that of rhythmic pixel regions for face detection and pose estimation workloads and substantially higher for visual SLAM. Video compression generates a substantially higher amount of memory traffic since it operates on multiple frames.

Discarding pixels reduces memory footprint: Rhythmic pixel regions not only relieve the memory interfaces but also helps reduce the pixel memory footprint of the vision pipeline, with similar trends, also shown in Fig.8. Specifically, with our paradigm, the average frame buffer size reduces by roughly 50% compared to frame-based computing. Frame buffers are storage intensive components in the vision pipeline; with smaller frame buffers, a system can not only reduce storage energy, but also potentially afford to store buffers locally inside the SoC itself, thereby reducing reliance on DRAM. Metadata produced by the encoder incurs minimal memory overhead. Specifically, EncMask and row offsets amount to 8% of pixel buffer storage for a 1080p frame.

Energy efficiency implications of rhythmic pixel regions: Memory efficiency begets energy efficiency; as is widely acknowledged in the computer architecture community, systems expend significant energy to move data in and out of memory. Based on first order modeling (see Appendix), with an assumption of 300 pJ to read a pixel and 400 pJ to write a pixel, the reduced interface traffic of rhythmic pixel regions reduces energy consumption by 18 mJ per frame for RP10 on V-SLAM at 4K and 30 fps. This reduces power consumption by 550 mW. The precise energy savings will depend on system characteristics that are highly specific to the device, operating system, and application usage pattern. We leverage the coarse model to contextualize the benefits of reducing pixel memory throughput.

Further energy reduction could result from deeper investigation in tuning vision algorithms and architectures to reduce their computational workload on lower resolution workloads, e.g., avoiding computation on zeroed pixels. Power reduction through sparse computing has been shown to be effective strategy Chen *et al.* (2019); Zhang *et al.* (2020); Gondimalla *et al.* (2019), which is a focus of future work.

2.6.3 The Hardware Extensions Are Lightweight and Scalable

Since real-estate is a precious resource on the SoC, our hardware extensions need to be lightweight and scalable to support a multitude of regions without taking too many resources.

Encoder scales well with number of regions: As shown in Table 5, the encoder is able to support additional regions without needing significantly more transistors. This is because our encoder utilizes a hybrid architecture that uses the processor for pre-sorting at the OS level and specialized architecture for shortlisting regions. This significantly reduces the number of comparisons needed, thereby favoring scalability. On the other hand, resource footprint for a fully-parallel based encoder design substantially increases with more regions to such an extent that they cause synthesis issues on the FPGA.

Decoder is agnostic to number of regions: As the decoder uses bitmasks (EncMask) instead of region labels, our decoder design is agnostic to the number of regions. Specifically, it needs 699 LUTS, 1082 FFs, and 2 BRAMs (18Kb) for 1080p decoding, regardless of the number of supported regions.

Encoder/decoder are performant: Encoder and decoder designs do not affect the pipeline performance. Our end-to-end system with encoder and decoder runs in real-time; our encoder meets the 2 PPC constraint of the capture pipeline. For

Table 5: Resource Utilization for Different Encoder Designs

Type	#Regions	Resources		
		#LUTs	#FFs	#BRAMs
Parallel	100	4644	5935	6
Parallel	200	8635	10935	6
Parallel	400	16251	20685	6
Parallel	1600	No Synth	No Synth	No Synth
Hybrid	100	942	1189	11
Hybrid	200	949	1190	11
Hybrid	400	944	1191	11
Hybrid	1600	952	1186	11

the decoder, since it intercepts every pixel transaction and appropriately modifies the transaction response, it will add a few clock cycles of delay when returning the response. We find that this delay is the order of a few 10s of ns and is negligible compared to the frame compute time – typically 10s of ms – of vision workloads.

Our alternative software decoder also runs in real time, consuming a few ms of CPU time for a 1080p frame where 30% of the pixels are regional pixels. The software decoder linearly scales in time to the amount of regional pixels.

Encoder/decoder are power-efficient: Our hardware extensions need to be power-efficient so as not to outweigh potential energy savings. Our encoder consumes 45 mW for supporting 1600 regions, which entails less than 7% of standard mobile ISP chip power (650 mW). Our decoder consumes < 1 mW of power. These power consumption estimates are based off of FPGA targets; power-efficiency improves for ASIC targets, which we will study as future work.

2.7 Future Directions

In this section, we first discuss the next research direction that we would like to explore deeply and then talk about other broad research avenues that we wish to study.

Rhythmic pixel regions through clock and voltage scaling on camera: Dynamic voltage and frequency scaling (DVFS) has been a popular thermal management technique for regulating temperature of microprocessors. Different from microprocessors, we would like to study the implication of clock and voltage scaling on camera readout. Our key insight is that the system could read frames at higher speeds by clocking the camera electronics faster. On top of that, we could apply sub-voltage thresholding to reduce the power consumption of analog power rails while keeping the digital power rails fully powered. Built on these insights, we bounce the clock and voltage of image sensor to modulate different pixel region readouts.

We intend to study this phenomenon around OnSemi's rolling shutter camera that has a programmable clock. We plan to modify clock frequency and voltage level using National Instruments Data Acquisition Control (NI DAC) and collect different images to study different region readout patterns.

Edge - cloud scenario: In this work, we explore the idea of rhythmic pixel regions for embedded vision, where both capture and processing happen on the same device. We will extend the idea of rhythmic pixel regions in edge - cloud scenario. We believe that edge - cloud use-cases will significantly benefit from early pixel discard as wireless data movement is at least three orders of magnitude more expensive than wired data movements. Specifically, in IoT kind of use-cases, the camera captures high-resolution data and sends all the data wirelessly to cloud. The cloud processing units perform vision and sends back the results to edge for display.

With rhythmic pixel regions, the system will send much lesser pixels, only the interesting ones, to the cloud. By doing so, we not only save the on the communication energy but also on the network latency. To realize this, we will build a low-power FPGA platform and place the encoder on the platform. We will place the decoder on the cloud which will decipher the encoded pixels and sends them to processing units for performing vision. The results of the vision will be sent back to the edge for guiding the selection of next pixel regions.

Camera to display pass-through: In this work, we did not explore the display element of the vision pipeline. Vision tasks such as AR would need a fully captured frame to be always sent to the display for overlaying digital animations on top of it. To this end, we will plan to design a dedicated data path which seamlessly sends data from camera to display without having to go through the memory. The raster-scan based design of mobile cameras and displays makes it possible to design such a memory-free data path. In parallel to this data path, we will have a vision data path which will perform localization on the camera frames to identify places of overlay. Based on the localization information, the rendering engine usually placed on the GPU will pull the appropriate content. Finally, we will mix the full frame from the first data path with the rendered animations from the second data path before pushing it to the display. **DRAM-less Computing:** The paradigm of rhythmic pixel regions significantly reduces the average size of the frame buffer. This presents an opportunity to store frame buffers in the local SoC memory when not dealing with full frame captures. By doing so, the system would be less dependent on DRAM, thereby significantly reducing data movement. In the future, we will study such an integrated memory-compute system and its effectiveness in reducing memory datarate.

Rhythmic Pixel Camera: In this work, we place our encoder after the camera capture inside the SoC. However, the MIPI interface which sends the pixels from camera to SoC is still burdened with data movement. To this end, we plan to study an integration of our encoder inside the camera module to reduce MIPI interface traffic for further energy savings.

Pixel Region Selection Policies: We will further explore potential policy tradeoffs of rhythmic pixel regions. We will study motion estimation techniques in guiding the region label selection. Machine learning, e.g., reinforcement learning, could also autonomously guide region selection. We believe these will unlock the potential of rhythmic pixel regions for high-precision visual computing with efficient memory use.

Chapter 3

SQUINT: A FRAMEWORK FOR DYNAMIC VOLTAGE SCALING OF IMAGE SENSORS TOWARDS LOW POWER IOT VISION

Energy-efficient visual sensing is of paramount importance to enable battery-backed low power IoT and mobile applications. Unfortunately, modern image sensors still consume hundreds of milliwatts of power, mainly due to analog readout. This is because current systems always supply a fixed voltage to the sensor's analog circuitry leading to higher power profiles. In this work, we propose to aggressively scale the analog voltage supplied to the camera as a means to significantly reduce sensor power consumption. To that end, we characterize the power and fidelity implications of analog voltage scaling on three off-the-shelf sensors. Our characterization reveals that analog voltage scaling reduces sensor power but also degrades image quality. Furthermore, the degradation in image quality situationally affects the task accuracy of vision applications.

We develop a visual streaming pipeline that flexibly allows application developers to dynamically adapt sensor voltage on a frame-by-frame basis. We develop a voltage controller that programmatically generates desired sensor voltage based on application request. We integrate our voltage controller into the existing RPi-based video streaming IoT pipeline. On top of this, we develop runtime support for flexible voltage specification from vision applications. Evaluating the system over a wide range of voltage scaling policies on popular vision tasks reveals Squint imaging can deliver up to 73% sensor power savings, while maintaining reasonable task fidelity.

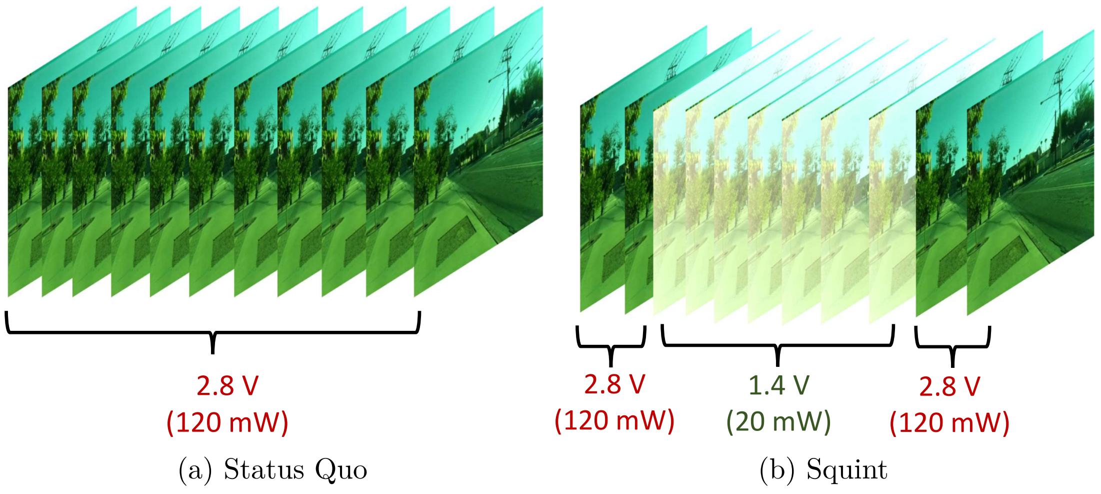


Figure 10: Existing Systems Supply a Fixed Voltage to the Image Sensor Regardless of the Frame. Squint Adapts the Camera Supply Voltage per Frame for Energy-Efficiency.

3.1 Introduction

Visual computing systems sense and perceive the world using cameras enabling a variety of IoT and mobile applications. IoT systems use cameras to perform tasks such as predicting irrigation patterns in AI-enabled farms Vasisht *et al.* (2021), detecting wildfires in forests Delbracio *et al.* (2021), and monitoring animal movements in wildlife sanctuaries Bick *et al.* (2021); Iyer *et al.* (2020). Energy-efficient sensing is of utmost importance for such IoT systems as they are deployed in the wild and typically operate on miniscule batteries. Energy-efficient sensing is also important for multi-camera mobile systems, such as augmented reality headsets, wherein several cameras, e.g., 18 cameras on Magic Leap v2 headset Tom's guide (2022), perform tasks such as world tracking, eye tracking, and body tracking, quickly draining the system's battery.

Unfortunately, modern image sensors are limited in their energy-efficiency due to analog readout and consume hundreds of milliwatts of power while performing continuous visual tasks. This is because existing systems (Fig. 10a) always supply a fixed voltage to the sensor's analog circuitry regardless of the frame. This work aims to improve the capabilities of visual computing systems by allowing them to dynamically vary the analog voltage supplied to the sensor on a frame-by-frame basis (Fig. 10b) for significant efficiency gains.

While dynamic voltage scaling of image sensors leads to large power savings, it also degrades the sensor's imaging fidelity due to reduced ADC output swing and increased pixel noise (§3.2). To assess fidelity issues, we characterize the energy and fidelity implications of analog voltage scaling of three popular off-the-shelf cameras. In addition to confirming the large sensor power savings created by aggressive voltage scaling, our characterization reveals a useful insight: degradation in imaging fidelity due to aggressive voltage scaling does not significantly affect the task accuracy of modern neural network based vision workloads in most situations. That said, we find that high fidelity is still sporadically needed to precisely detect scene features for tracking tasks, especially under challenging situations such as low-light and crowded scenes.

The idea to dynamically scale a component's voltage for reduced energy usage is inspired by the CPU power savings technique of dynamic voltage scaling (DVS) Weiser *et al.* (1994). However, unlike CPU's DVS, analog voltage scaling here introduces unique noise considerations, due to the direct reliance on the analog circuitry's fidelity for image readout. Sensor manufacturers have been moving towards designing sensors Image Sensors World (2021a,b,c); Choi *et al.* (2015b) with lower pixel supply voltage. These sensors need explicit design changes to overcome the fidelity issues caused due to lower pixel voltage.

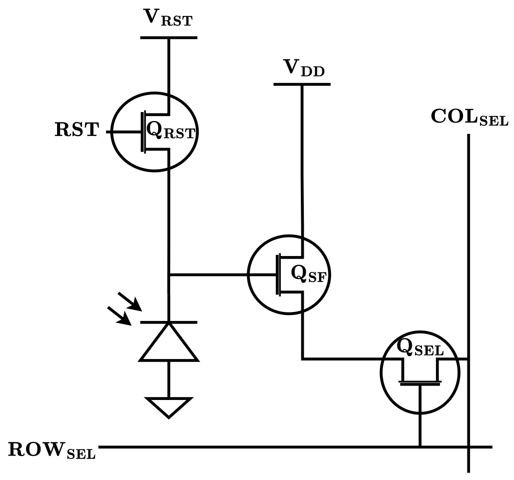
To design the system support (§3.3) to allow sensor dynamic voltage scaling, we introduce a simple voltage controller hardware interface that can programmatically generate desired sensor voltage requested by applications. On top of that, we introduce the Squint runtime that includes a software API allowing application developers to seamlessly specify voltage schedules directly from the vision applications on a frame-by-frame basis. These interfaces integrate well with existing and future visual computing systems, allowing for an easier upgrade.

To evaluate our system, we design our implementation (§3.4) around a RPi based streaming pipeline. We support various visual workloads, including neural network-based people detection and OpenCV-based camera pose estimation. Through our evaluation (§3.5) across a range of voltage scheduling policies, we demonstrate the opportunity of Squint based imaging to decrease sensor power by up to 73%, while only minimally degrading the visual task accuracy. Through reduction in sensor power, Squint imaging techniques can significantly extend the battery life of IoT and mobile systems.

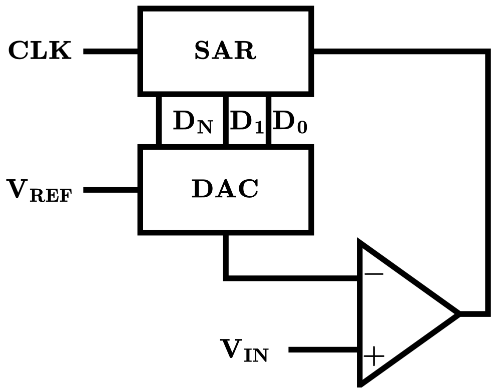
In summary, we make the following contributions:

- We explore the idea of dynamic voltage scaling for image sensors, where different frames are captured at different fidelities for overall system energy-efficiency, while respecting task needs.
- We characterize the energy and fidelity implications of analog voltage scaling on off-the-shelf commercial image sensors.
- We develop a lightweight and fully programmable voltage controller to generate desired analog supply voltage for the camera. On top of that, we develop a library and runtime to coordinate vision applications with voltage controller operation.

- We augment our hardware and software support on top of an existing commercial IoT streaming pipeline built around the RPi platform. We evaluate the augmented system on a variety of vision tasks to demonstrate significant reduction in sensor power with controllable accuracy loss.



(a) Pixel Design



(b) ADC Design

Figure 11: Analog Voltage Variation Affects Pixel and ADC Circuits. A Lower Voltage (VDD) for Pixels Would Affect Transistor Settling Times, Resulting in Noise. A Lower Reference Voltage (VREF) for ADC Would Decrease the Dynamic Range, Resulting in Reduced Contrast.

3.2 Background and Motivation

3.2.1 Analog Readout Is The Bottleneck for Sensor Energy-Efficiency

Modern IoT and mobile systems employ CMOS image sensors for video analytics and photography. CMOS image sensors are more efficient than their CCD counterparts, but still consume hundreds of milliwatts, due to their power-hungry analog readout LiKamWa *et al.* (2013).

Modern image sensors comprise three major elements: pixel array, analog readout, and digital logic. The pixel array converts light into voltages, consuming little energy Kitamura *et al.* (2012). Analog readout amplifies analog signals from the pixel array and converts them into digital values. Finally, digital logic maintains sensor operation, including timing generation and preparing data to send to the outside world.

Notably, the analog readout consumes 50%-75% of overall energy in recent sensor designs Kitamura *et al.* (2012); Choi *et al.* (2015a); Takayanagi *et al.* (2005). This is due to two factors: (i) Analog circuits need to operate all the time, making them power hungry compared to their digital counterparts which operate only on the active clock edge. (ii) Technology scaling improvements for energy efficiency have been much slower for analog components compared to digital.

3.2.2 Effects of Analog Voltage Variations on Image Sensor Circuitry

Existing systems supply higher analog voltage, typically 2.8 V, for high fidelity imaging Lin *et al.* (2004). Higher analog voltage (VDD) is needed to maintain sufficient drive strength of the photodiode, source follower, and bitline components in the pixel design, shown in Fig. 11a, for smoother charge transfer. Lowering analog voltage would weaken the drive strength of those components affecting transistor settling times Fossum and Hondongwa (2014); Lin *et al.* (2004), resulting in image noise. Lowering analog voltage would also slow down Image Sensors World (2021b,c) the charge transfer process. This would not significantly affect the imaging process if the photodiode accumulates enough charge, e.g., well-lit scenes. However, this would make the images dimmer in low-lit scenes where photodiodes do not accumulate enough charge.

Higher analog voltage (VREF) is also needed for high fidelity ADC operation for maintaining high dynamic range. Image sensors typically have a successive-approximation (SAR) ADC (Fig. 11b) that performs a binary search for generating

Table 6: Characterized Image Sensors. Sensors Can Be Undervolted Way Beyond the Voltage Specification Mentioned in Their Datasheets.

Camera system	**Sensor**	**Max. res**	**Voltage range (datasheet)**	**Actual lower limit**
RPi cam	Sony IMX219 Sony (2022b)	4K	2.8 +/- 10%	1.2 - 1.5
Pixycam	OmniVision OV5647 OmniVision (2022a)	1080p	2.8 +/- 10%	1.8
Python1300	OnSemi Python1300 OmniVision (2022b)	SXGA	2.8 +/- 10%	1.7

digital output by successively comparing input voltage (VIN) against reference voltage (VREF) Baker (2019). A higher VREF would enable higher precision output supporting a wider range of pixel intensity values leading to high dynamic range images. A lower VREF, on the other hand, would decrease the dynamic range forcing the pixels to saturate early and decreasing the overall image contrast.

While the exact pixel and ADC designs might vary for different image sensors, the fundamental underlying phenomenon of how the circuits behave at different analog voltage levels would remain the same. Furthermore, the underlying phenomena is agnostic to the shutter type, rolling or global, of the image sensor as the shutter type does not affect the charge transfer process and ADC operating mechanism.

3.2.3 Analog Voltage Scaling for Energy-Efficiency

Towards addressing the energy-efficiency bottleneck, we propose to scale analog voltage to promote camera energy efficiency. We study the energy and fidelity implications of voltage scaling by characterizing three different image sensors from three popular manufacturers, as shown in Table 6. These image sensors vary in terms of resolution, shutter type, and pixel designs, and are representative of cameras available in typical IoT and mobile systems. In this section, we mainly present the findings of RPi camera characterization; while other cameras revealed similar insights.

We observe that image sensor manufacturers usually take a conservative approach when it comes to specifying the recommended operating analog voltage for stable camera operation. They typically recommend a 10% tolerance band around nominal voltage in the datasheets. In reality, however, we find that we can undervolt the analog power rail to a value that is far beyond the recommended limit, as shown in Table 6, without affecting basic camera functionality. For instance, we can aggressively undervolt the RPi's analog power rail by up to 50% of its nominal value with the camera still streaming pixels.

The extent to which we can undervolt a camera strongly depends on sensor design and the sensor's sensitivity to analog voltage changes. Among the three sensors we characterize, we notice the RPi camera can be undervolted by a large extent compared to other ones. In addition to variability across cameras from different vendors, we also notice variability across cameras from the same vendor, as shown in Table 6. This is due to the silicon lottery Hennessey and Patterson (2018): One RPi camera can undervolt slightly better than the other and vice versa.

3.2.3.1 Energy Implications of Voltage Scaling

Aggressive undervolting leads to significant energy savings, as dynamic power quadratically varies with voltage ($P \propto V^2$). Since analog circuits are slow on technology scaling, dynamic power still dominates the overall power consumption by contributing to about two thirds of total power. On the other hand, static power, which is the less dominant power source in analog electronics, is largely unaffected by voltage scaling.

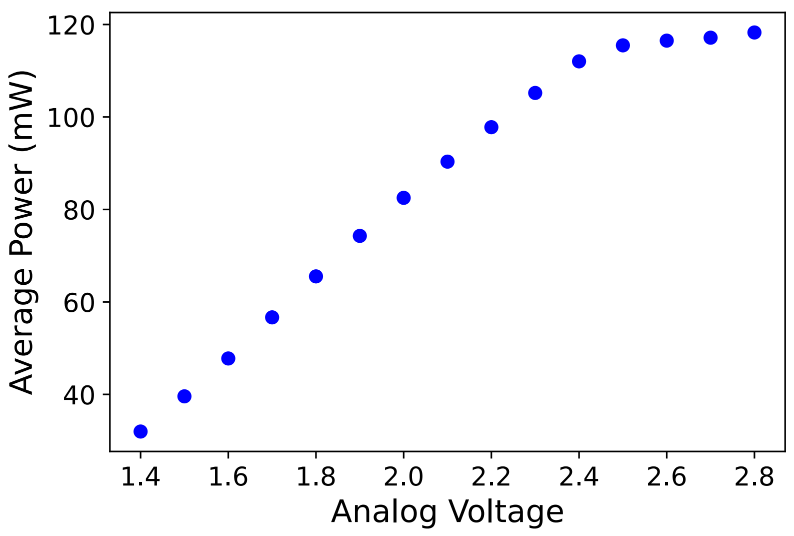
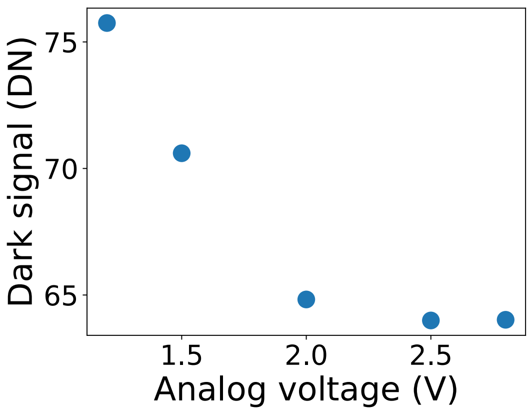


Figure 12: Sensor Power Significantly Reduces by Lowering Its Supply Voltage. The Flat Portion Above 2.4 V Is Due to Strong Voltage Regulation Inside the Sensor.

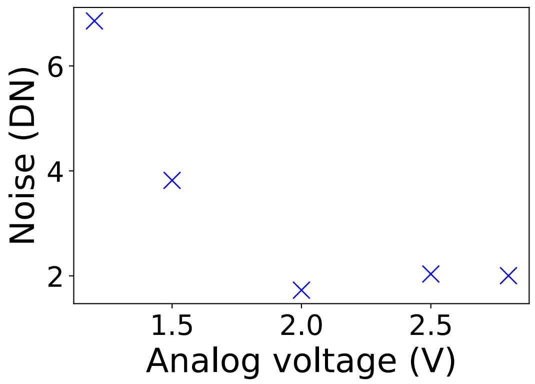
To understand how camera power varies with analog voltage scaling, we intercept the analog power rail of the RPi camera and instead connect it to an external variable power supply. To compute overall power, we measure the current drawn from different power rails, while we vary analog voltage. We find that undervolting the camera voltage by 50%, i.e. from 2.8 V to 1.4 V, yields more than 50% of camera power savings, as shown in Fig. 12.

3.2.3.2 Image Fidelity Implications of Voltage Scaling

While aggressively reducing analog voltage helps significantly reduce camera power, it also impairs image quality by making images brighter and noisier. Increased brightness is due to reduction in ADC dynamic range. That is, image sensors usually have a SAR ADC and lowering the ADC's reference voltage results in lowering the

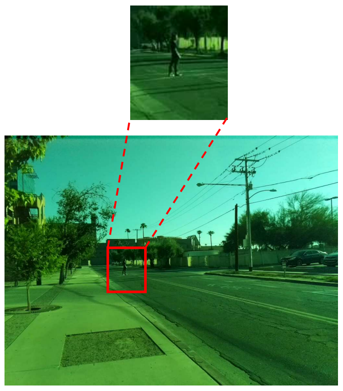


(a) Brightness

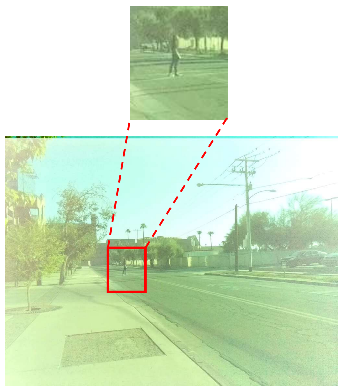


(b) Noise

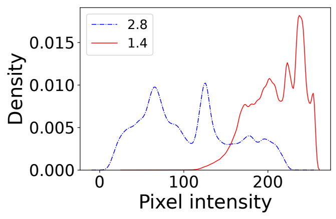
Figure 13: Sensor's Brightness and Noise Levels Increase with Lower Analog Voltage. Increased Brightness Is Due to Decreased ADC Output Swing and Increased Noise Due to Elevated Shot Noise Levels.



(a) Image (2.8 V)



(b) Image (1.4 V)



(c) Histograms

Figure 14: Images Captured at Different Sensor Voltages and Their Histograms. Low Voltage Image Is Brighter and Grainier Than Its High Voltage Counterpart. This Is Also Reflected in the Shift in Mean and Variance Width in the Histogram. Images Appear More Greenish Because They Are Captured RAW Turning Off White Balance.

output signal swing. This forces pixels to saturate early, brightening the image and decreasing the overall contrast.

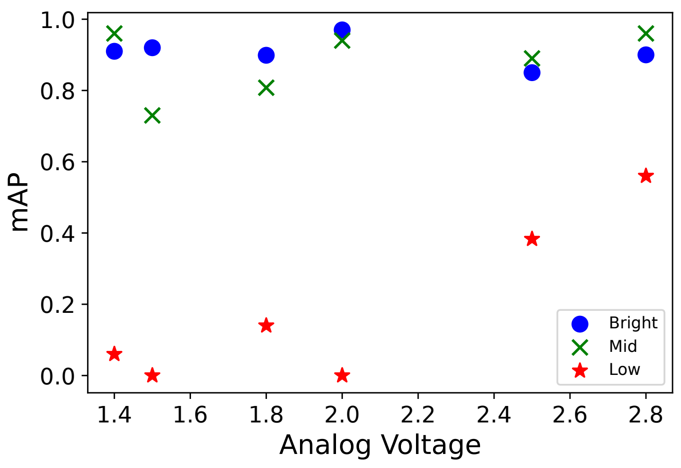
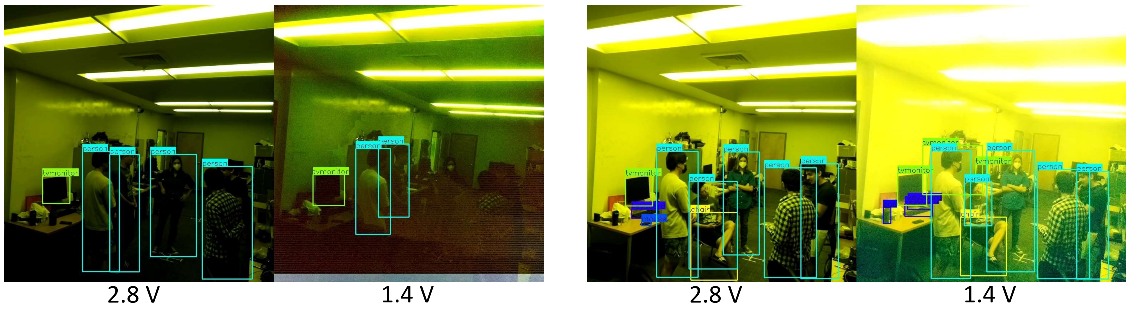


Figure 15: Brighter and Noisier Images Do Not Significantly Affect Task Fidelity of Vision Applications in Well-Lit Scenes.

Increased noise is due to more pixel shot noise triggered by reduction in pixel bias voltage as shot noise exponentially varies with the negative of bias voltage. In addition to brightness and noise artifacts, we also notice pixelation artifacts appearing in images. This is due to pixels randomly getting turned off possibly due to timing errors caused by insufficient supply voltage.

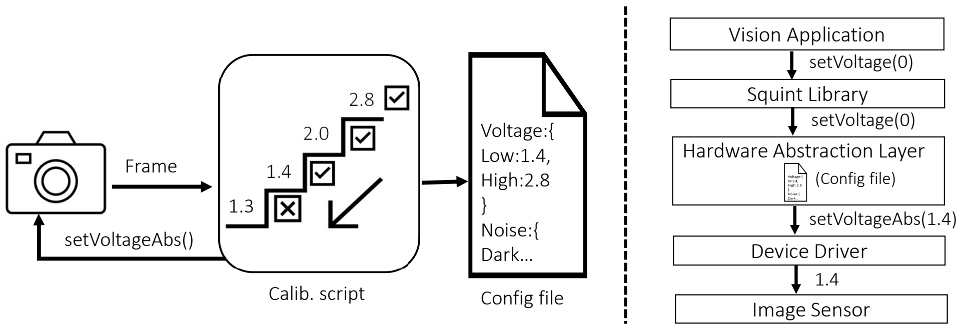
We experimentally validate these claims by measuring the standard dark signal Albert Theuwissen (2012) and temporal noise Albert Theuwissen (2011) of the RPi camera, while we vary sensor analog voltage. We capture 25 frames per voltage setting and average pixel values across frames to determine the average frame. We average the pixel values of the average frame to compute the dark signal. For temporal noise, we compute variance of pixel values across frames to determine the variance frame. Then we average all the values in the variance frame to compute the temporal noise.

As shown in Fig. 13, we notice the dark signal and the temporal noise substantially increase as we decrease the sensor voltage. We also noticed the sensor being insensitive



(a) Dimly-Lit (b) Well-Lit

Figure 16: High Fidelity Is Still Needed for Precise Object Detection in Challenging Scenarios. Low Fidelity Would Suffice Otherwise.



(a) Calibration (b) Execution

Figure 17: Phases of Squint. During Calibration, Squint Sweeps Voltage Supplied to the Camera to Determine the Undervolting Limit as well as to Profile Noise, and This Information Is Stored in a Configuration File. During Execution, the System Uses the Configuration File to Translate Developer's High-Level Fidelity Input to Actual Low-Level Camera Voltage Using Standard Camera System Stack.

to voltage changes until a certain point, which could be due to strong voltage regulation inside the sensor. The brightness and noise increase is also reflected in the histograms of actual images, as shown in Fig. 14. We can see that the peaks shift rightwards and the distribution spreads outwards as we switch from high voltage to low voltage.

On a side note, we can also see that the actual images in Fig. 14 appear more green because they are captured RAW by turning off most image signal processing (ISP) stages. Notably, we turn off automatic white balance (AWB) to avoid color based artifacts produced by AWB. AWB algorithms Ebner (2003, 2007); Cheng *et al.* (2014);

Van De Weijer *et al.* (2007) estimate color temperature of a scene based on average image intensity in order to determine appropriate gains that need to be applied for individual color channels of an image. As a result, AWB perceives brightness increase due to undervolting as change in scene brightness and apply more gain to the blue color channel resulting in bluish artifacts.

3.2.3.3 Task Fidelity Implications of Voltage Scaling

While aggressive undervolting impairs image fidelity by adding brightness and noise, we find that brightness and noise increases do not affect visual task fidelity in most situations. However, we see a noticeable degradation in task fidelity under challenging scene environments, e.g., low-light scenes, as shown in Fig. 15. This creates an opportunity to situationally adapt camera voltage based on scene illumination needs. That is, we can aggressively undervolt the camera most of the time for rapid power savings and occasionally bring the voltage back to nominal under poor lighting conditions.

The insensitivity of vision tasks to brightness and noise increase in most situations stems from the fact that neural networks running behind these tasks are usually trained to make themselves immune to intensity and noise changes Ziyadinov and Tereshonok (2022). That being said, vision tasks still need high fidelity for performing precise object detection, especially under challenging conditions such as dimly-lit scenes, as shown in Fig. 16a. For less challenging scenes, we observe that low-fidelity images provide comparable task performance to their high-fidelity counterparts, as shown in Fig. 16b. For this analysis, we experimentally capture images at different voltage settings from the RPi camera for three different lighting conditions. We then feed the images to a YOLO-based people detection task and compare the detection results against ground truth to determine accuracy.

3.2.4 Motivational Observations

To summarize, we have the following insights for image sensor voltage scaling:

- We can aggressively undervolt image sensors far beyond specified operating ranges in the datasheets.
- Aggressive undervolting results in significant camera power savings.
- Aggressive undervolting also makes images brighter and noisier due to the reduced upper bound of the analog voltage range
- Raised brightness and noise levels in images does not significantly affect vision task accuracy in most situations

These observations motivate the need for dynamic voltage scaling strategies for image sensors to save energy at sufficient task fidelity.

3.3 Design

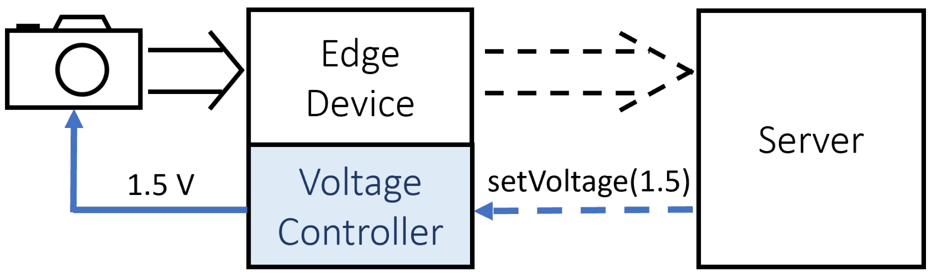
Design aspirations: We set three goals that guide the design of system support for sensor dynamic voltage scaling through our hardware and software extensions.

- Seamless: Vision applications should be able to reconfigure camera voltage on a frame-to-frame basis without any frame drops
- Lightweight: Our hardware extension should be lightweight with minimal system overhead
- Flexible: Our runtime should allow developers to flexibly specify camera voltage needs on a per-frame basis.

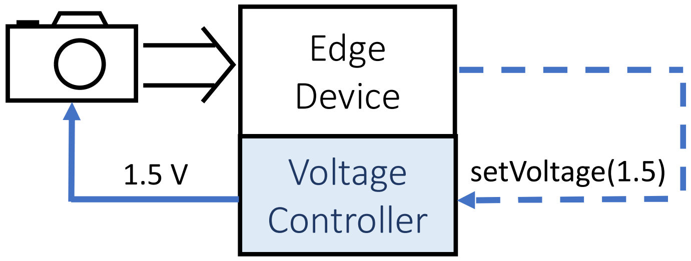
Calibration and Execution Phases: We propose two phases for Squint to seamlessly work with existing and future visual computing systems, as shown in Fig. 17.

1. To account for variations across sensors, the system goes through a one-time calibration phase to determine the actual voltage limits and noise profile of the image sensor.
2. The system then enters an execution phase to guide policy based voltage selection based on the calibration information.

(1) Calibration phase: During this phase, the system runs a script that automatically sweeps through different camera voltages and determines what set of voltages are usable. The script starts with the default nominal voltage specified in the datasheet and keeps decreasing the voltage in steps of one tenth of a volt until the camera freezes, at which point the camera voltage is restored to nominal. While it does so,



(a) Server-Based



(b) Edge-Based

Figure 18: Different System Incarnations for Squint. In Server-Based, Vision Apps Running on Server Issue Voltage Request to the Voltage Controller, Whereas in Edge-Based, Vision Apps Run on the Edge Device Itself and Directly Issue Requests to the Voltage Controller.

the script also captures several frames for noise assessment. The system commits the voltage limits and noise data into a configuration file, which could be stored in the hardware abstraction layer of the system stack.

(2) Execution phase: During this phase, the system uses the configuration file generated during the calibration step to translate the application's high-level fidelity requirements into actual set of voltages that needs to be applied to the camera. These voltages are sent to the voltage controller which enacts them on the camera using standard device-driver mechanisms.

System support for Squint: To efficiently run the calibration and execution phases, we build system support for Squint by extending the existing IoT pipeline with necessary hardware and software extensions. These extensions could be integrated into different system incarnations, such as server-based and edge-based, depending on where the vision application runs in the system, as shown in Fig. 18. The resulting Squint system architecture centers around the idea of adaptively scaling camera voltage for vision application usage. We describe the design of our voltage controller and its integration with the existing IoT streaming pipeline, as shown in Fig. 18. We also discuss the software runtime for vision application developers to leverage the voltage controller through policy specification.

3.3.1 Programmable Voltage Controller

The voltage controller module, shown in Fig. 19, intercepts the incoming voltage reconfiguration command from a server and produces appropriate analog voltage to power the camera. The vision application specifies voltage needs using the designed runtime support in §3.3.2.

We design a fairly simple voltage controller comprising a digital to analog converter (DAC) and a voltage buffer. The DAC programmatically takes desired voltage from vision application and converts that into equivalent analog voltage to power the camera. The DAC is supported by the voltage buffer which is a unity-gain amplifier that generates sufficient current for stable camera operation.

3.3.1.1 Integration with Edge Device

The voltage controller could be easily integrated as a standalone integrated circuit on top of the edge device. Alternatively, we could repurpose existing DVFS controllers

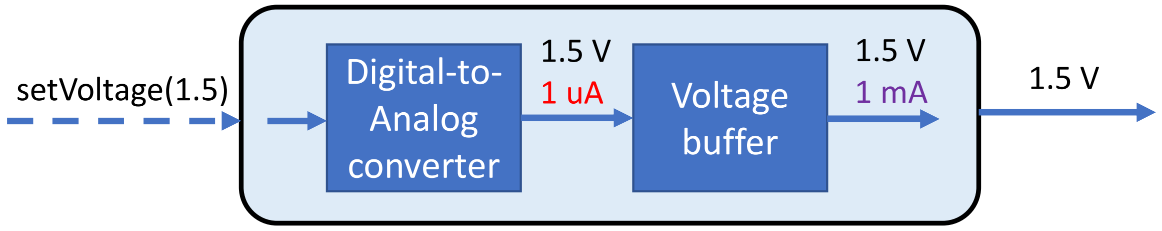


Figure 19: Voltage Controller Comprises a DAC to Translate Digital Voltage Requests into Corresponding Analog Voltage, and a Voltage Buffer to Generate Sufficient Current Flow for Reliable Camera Operation.

inside SoCs of edge devices to generate desired voltages for the camera. That would entail adding a few extra power modes to meet camera power supply requirements.

3.3.2 Developer Support

We develop runtime support to allow the developers to flexibly specify voltages. This consists of a `SetCamVoltage()` function for developers to set a voltage. Voltage can be set on a per-frame basis or persist across frames. A runtime service receives these calls to send the voltage to the voltage controller.

```
SetCamVoltage(float val, string spec="SNR/Rel/Abs")
```

In order to facilitate development, we provide multiple levels of abstraction for voltage specification. We provide sensor-agnostic ways to configure voltage such as through "SNR" and relative voltage specification ("Rel"), allowing developers to be unconcerned with the intricacies of imaging and power trends. Additionally, we also provide sensor-specific ways to configure voltage to allow developers to directly set absolute ("Abs") camera voltages.

For relative voltage specification, we let developers specify their desired voltage in [0, 1] range with 0 denoting the low voltage limit and 1 denoting the high voltage limit,

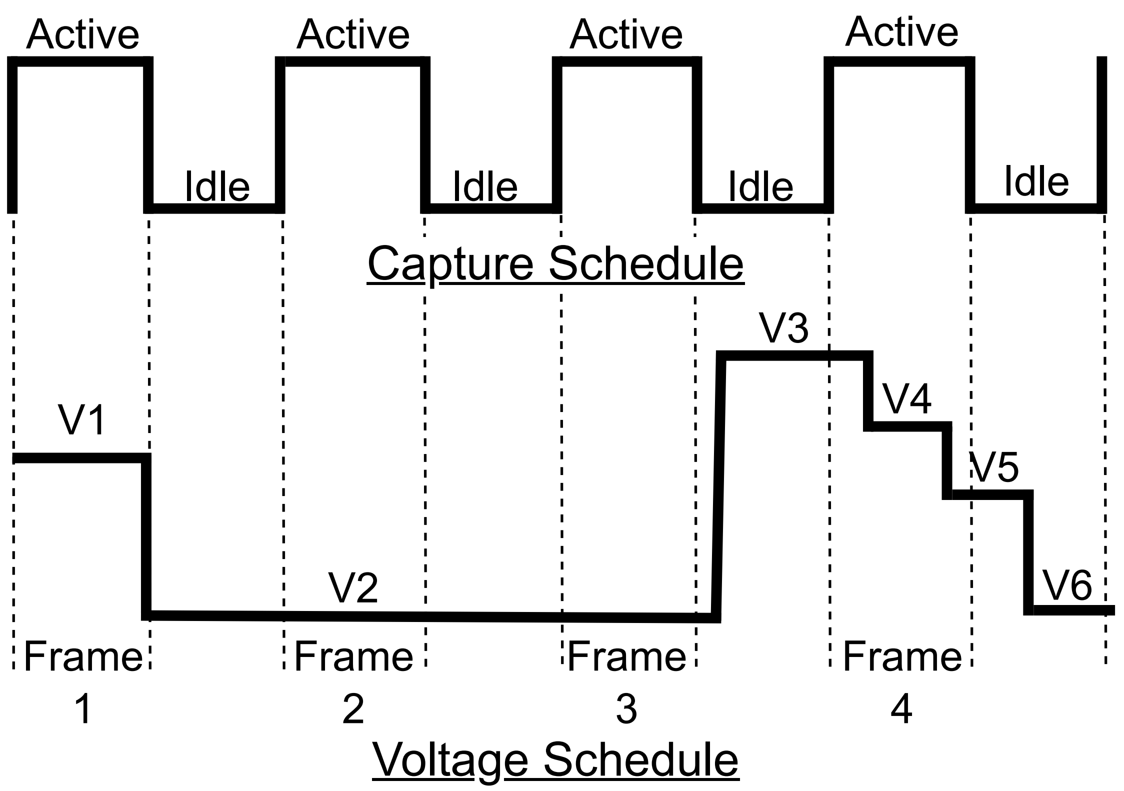


Figure 20: Timing Different Voltages in Tandem with Camera Capture Schedule Leads to Different Fidelity Patterns Across the Frames and Within a Frame.

regardless of the camera. Our decision to abstract this information is grounded in the observation of the monotonicity of imaging (Fig. 13) and power (Fig. 12) trends.

Our runtime service translates the specification into appropriate hardware voltage with the help of the configuration file generated during the calibration phase of the system. The final voltage is then enacted to the sensor hardware through device driver, similar to how it is done in standard Android and iOS mobile systems architecture.

3.3.2.1 Integration with CameraAPI and Sysfs Interface

Our voltage configuration method could be easily integrated as part of standard CameraAPI. The CameraAPI is an application programming interface that allows software developers to interact with a device's camera hardware, including controlling camera settings such as exposure, focus, and zoom, in a platform agnostic manner. By integrating voltage control into the existing CameraAPI, developers can easily configure voltage through a familiar interface.

```
VideoCapture cap(0);
cap.set(cv::CAP_PROP_EXPOSURE, 0.0167);
cap.set(cv::CAP_PROP_VOLTAGE, 0.5);
```

The above snippet shows an example integration using the popular OpenCV-based Linux CameraAPI. In this approach, voltage can be defined as a camera property and set to a desired value, similar to other camera settings such as exposure. This method of integration can also be adapted to Android and iOS cameraAPIs.

Alternatively, voltage control can be integrated as part of sysfs interface to allow developers to configure and query sensor voltage through a standardized interface controlled by the operating system. This approach involves modifying the camera device driver to create voltage as a sysfs file node, which can then be exported to userspace using the sysfs interface. Developers can also directly specify voltage scheduling policy, instead of voltage, to enable automatic voltage control by OS, reducing developer burden.

```
echo 0.5 > /sys/devices/camera/sensor/voltage
echo "Perf" > /sys/devices/camera/sensor/policy
```

3.3.2.2 Voltage Scheduling Policies

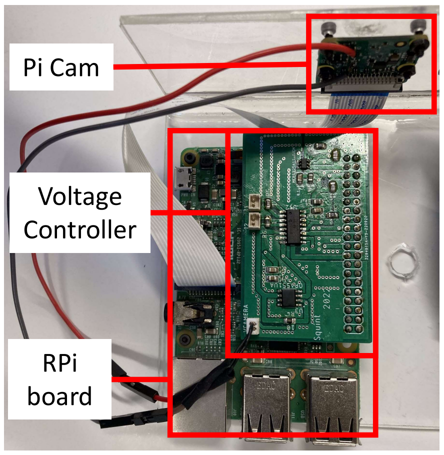
Developers can build various policies that autonomously guide the voltage configuration, in the similar spirit to issuing voltage configurations with the Linux DVFS API on desktop systems and Android's Frequency API on mobile systems. Policies can incur different system overheads, leading to system trade-offs.

A policy should predict voltage progression with time as well as image quality requirements to maximize task accuracy. A simple policy could occasionally operate a camera at high voltage to detect frames and undervolt the camera for tracking the detected objects across frames. Developers can also introduce improved application-specific proxies with other prediction strategies, e.g., with optical flow based voltage reconfiguration.

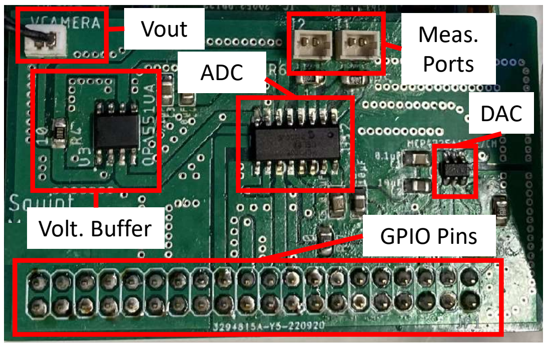
Notably, depending on how the voltage schedule is aligned with the camera's capture schedule, different fidelity granularities could be achieved within a single image, as shown in Fig. 20. Modern cameras employ a rolling shutter mechanism that exposes and streams one line of pixels at a time. For coarse-grained fidelity control, the system should sustain the voltage for the entirety of the active frame capture duration. On the other hand, for fine-grained fidelity control, the system could apply different voltages to different parts of the frame.

Table 7: System Components in the IoT Streaming Pipeline

Component	Specification
Camera	RPi Cam v2.1, Sony IMX219, 4K @ 60 fps
Edge device	RPi v 3B, quad-core Cortex-A53, 1GB RAM
Network	300 Mbps Ethernet
Server	Intel Xeon CPU E5-1620, 32 GB RAM



(a) Squint



(b) Voltage Controller

Figure 21: Our System Setup

3.4 Implementation

3.4.1 RPi-based System Integration

Platform: We build our IoT system (Fig. 21a) as a gstreamer pipeline around RPi to be representative of a standard video streaming system. As shown in Table 7, we use RPi v2 camera Sony (2022a) as the capture device, connected to a RPi v3B. The RPi forms the edge device that consumes the incoming pixel stream from the camera and compresses the stream into a compact representation using standard H264 compression techniques. We run a real-time streaming protocol (RTSP) server Micheal

Promonet (2022) on the Pi that transmits the compressed pixel stream from the edge device over Ethernet to the server. Finally, we utilize a server class Intel Xeon CPU desktop that decompresses the compressed pixel stream for performing video analytics, e.g., people detection. The entire pipeline runs in real-time at 24 fps.

Voltage controller: We design our voltage controller (Fig. 21b) as a fully programmable module using off-the-shelf components. Specifically, we use Microchip's MCP4725 12-bit DAC Microchip (2022) as our digital-to-analog converter and Texas Instruments OPA551PA operational amplifier Texas Instruments (2022) to build the voltage buffer that generates sufficient current for stable camera operation. We integrate the voltage controller as a standalone module, powered and interfaced through the I2C interface of the Pi. The voltage controller takes voltage configuration commands from the Pi and generates the desired voltage to power the camera's analog voltage rail.

Runtime: We implement our runtime as a filesystem interface (§3.3.2) using a stateless networking service protocol. Specifically, we implement a RESTful API Wikipedia (2022); Fielding (2000) to handle voltage configuration requests from the vision applications running on the server. A simple web service running on the Pi forwards these voltage configuration requests to the voltage controller over the I2C interface.

3.4.2 Workloads

We study YOLO-based people detection that localizes and tracks people in a scene to allow for counting analytics and potential security responses. This would be representative of multiple IoT video analytics applications such as counting people entering and leaving a trail entrance and analyzing shoppers' buying activity for automated checkout. Additionally, we perform a preliminary investigation of OpenCV marker based camera pose estimation that is the backbone for augmented reality tracking tasks. This would be representative of multi-camera AR systems such as Microsoft Hololens.

3.4.2.1 Benchmarks

Since existing datasets contain images taken at nominal camera voltage and the undervolting image artifacts are difficult to simulate, we construct our own dataset based on real image captures at different camera voltages. Specifically we stage our dataset representing two real-life scenarios:

- Entrance detection: people entering and leaving a scene
- Person tracking: people randomly standing around in a room holding casual discussions in places such as office cafeterias

Our dataset contains variations in lighting, object density, and object proximity, making it a viable proxy for real life scenarios. We recruit five people and take their consent for staging the dataset. We instruct the participants to move in specific patterns that emulate the entrance detection and the person tracking in cafeteria scenarios mentioned above. During the movements, we vary the ambient light using a controllable analog light source. Specifically, each capture consists of six minutes with two minutes of high light, then two minutes of medium light, and finally two minutes of low light.

We use our experimental RPi camera to stage the dataset at different voltages, while the participants emulate the scenarios. In addition to the experimental camera, we also use a ground truth camera that always operates at nominal voltage for ground truth captures. We place the experimental and ground truth cameras in close proximity so that they see the same scene with negligible parallax. Overall, the dataset includes 73,066 images. We plan to release the dataset at the time of publication.

For the marker based pose estimation, we construct a limited dataset using the same setup as above. For this workload, we place the marker at three different distances and three different orientations with respect to the camera.

3.4.2.2 Baselines

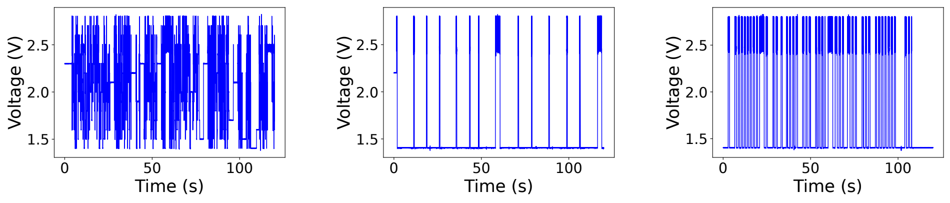
We test our workloads against the following baselines:

- Squint (Sq-X): The system supplies the desired voltage to the camera based on the vision application's fidelity needs. Here "X" denotes the policy used for voltage scaling. Notably, the system operates the camera at extreme voltages, i.e., 2.8 V and 1.4 V, for RPi camera, since there is no incentive to operate in between, as discussed in §3.2..
- Reduced resolution (RR): Instead of modulating voltage, the system changes capture resolution based on the fidelity needs. Analogous to Squint, the system configures the camera at 1080p for high fidelity and at 480p for low fidelity.

3.4.2.3 Metrics

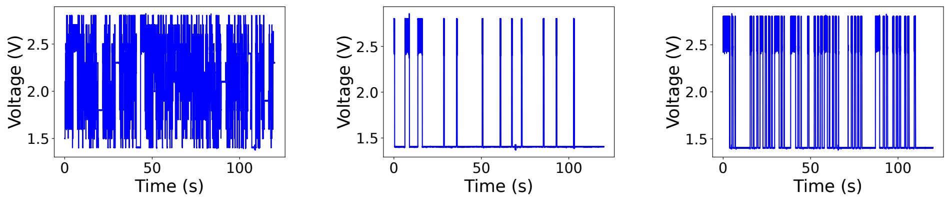
Here we discuss different evaluation metrics.

Task accuracy: We use the standard metrics from computer vision literature to evaluate the workloads. For the people detection workload, we use intersection over



(a) Random (b) Lightutil (c) On-Demand

Figure 22: Bright Light Voltage Schedules Generated by Different Policies for Entrance Detection Dataset.



(a) Random (b) Lightutil (c) On-Demand

Figure 23: Bright Light Voltage Schedules Generated by Different Policies for Person Tracking Dataset.

union (IoU) as the metric. IoU measures the amount of overlap between estimated and ground truth detections. A detection is considered a true positive (TP), if the IoU score is above a certain threshold. Otherwise, it is considered a false positive (FP). The final detection accuracy is obtained by determining the number of true positives among all the detections, i.e., TP/(TP + FP), for each frame, which is known as mean average precision (mAP). The final mAP is obtained by averaging individual mAP scores across all frames. On the other hand, for marker-based camera pose estimation workload, we use translational error and rotational error as the metrics. These are derived by computing the L2 norm of estimated pose against ground truth pose.

Power: We sample different camera power rails every 0.1 ms to measure their respective current draws. We multiply the current draw with the voltage of each rail and sum them over all the rails to get the sensor power. We analyze the sensor power readings across different samples to report power trends.

Overhead: We measure the voltage reconfiguration time of our API by measuring the time difference between voltage reconfiguration request sent by the vision application and acknowledgement notification received from the Pi's web service. We measure the reconfiguration time across multiple voltage requests to report trends. On the other hand, to report power overhead of the voltage controller module, we use worst case power estimates from the component datasheets.

3.4.2.4 Policies/Parameter Choices

We evaluate the following policies, in a similar spirit as CPU's DVFS policies Dehmelt (2014); Elgebaly and Sachdev (2007).

Performance (Perf): The system sets the sensor voltage to the highest possible setting, providing maximum task performance but potentially consuming more power and generating more heat. This is representative of status-quo.

Powersave (Pwr): The system sets the sensor voltage to the lowest possible setting to save power.

Random (Rand): To demonstrate the effectiveness of flexible voltage reconfiguration, we map a random policy whereby the system generates a random number from 1.4 to 2.8, and supplies that voltage to the image sensor.

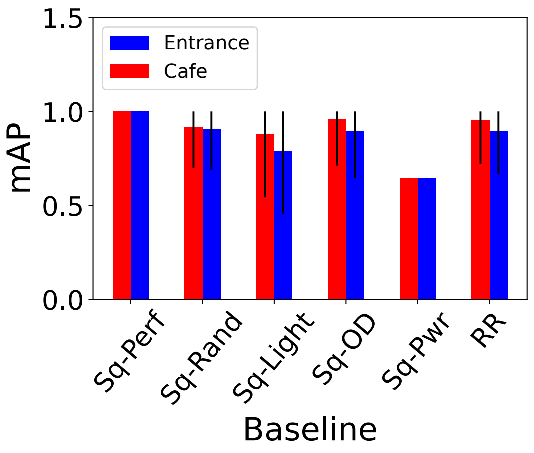
Lightutil (Light): The system uses ambient light sensor's reading to determine the optimal voltage setting for the current workload. The system uses a lighting - voltage lookup table that is obtained from sensor characterization to guide the voltage selection process.

For our evaluation, the system occasionally captures a high fidelity image, computes its average pixel intensity, and uses it as a proxy to estimate the ambient lighting of a scene. If the light estimate is greater than a certain threshold, the system supplies 1.4 V to the sensor; otherwise it supplies 2.8 V to the sensor, as guided by our characterization curves.

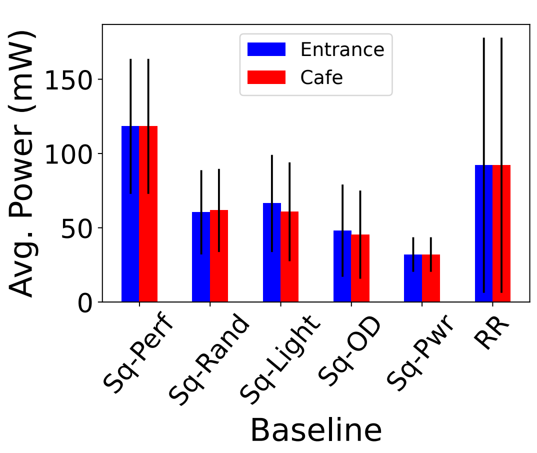
On-Demand (OD): The system sets the sensor voltage to the lowest possible setting when the scene is idle and scale the voltage up as scene complexity increases. Specifically, the system computes IoU score across detection results of previous frames to determine the staticness/dynamism of a scene's objects and uses the IoU score to appropriately determine the desired sensor voltage.

A larger IoU score across frames indicates that the scene is fairly static. In such scenarios, the system supplies 1.4 V to the camera to continue to track the existing objects. On the other hand, a lower IoU score across frames indicates large motion in the scene, including people entering and exiting the scene. In such scenarios, the system supplies 2.8 V to the camera for high quality detections.

Distance based (Dist): We use a distance based policy for marker based camera pose estimation workload, whereby the system adapts voltage based on proximity of marker with respect to the camera. The system supplies a low voltage (1.4 V) to the camera when the marker is close to the camera, and supplies a high voltage (2.8 V) when the marker is located far from the camera for precision.



(a) Task Accuracy



(b) Power

Figure 24: People Detection Task

3.5 Evaluation

3.5.1 The Usage of Squint Is Flexible, Seamless, and Performant

Our runtime flexibly allows apps to configure full swing of sensor voltages on a frame-to-frame basis without any restrictions. The voltage schedule for random policy shown in Fig. 22 indicates that the API is flexible enough to randomly configure sensor voltage anywhere from 1.4 V to 2.8 V. On the other hand, the system operates the image sensor at extreme voltages, either 1.4 V or 2.8 V, for Lightutil and On-Demand policies, as shown in Fig. 22, because there is no power/performance incentive to operate at an intermediate voltage, as mentioned in Section 2.

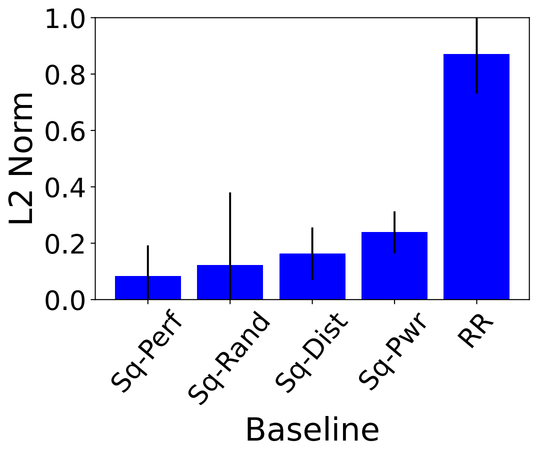
We find that the system chooses a different voltage schedule for different policies for a given scenario. For instance, we can see On-Demand policy schedule has a denser schedule than Lightutil, because it's motion adaptive, and motion changes are frequent in both the entrance detection and the office cafeteria scenes. Lightutil on the other hand has a sparser schedule because it's light adaptive and light changes are less frequent in the scenes.

In addition to variation across policies, we find that the same policy selects a different voltage schedule for different scenarios. For instance, we can see On-Demand having a denser schedule for the entrance detection scenario (Fig. 22) compared to the person tracking in cafeteria scenario (Fig. 23), as there is large motion involved in the former due to people constantly entering and leaving the scene as opposed to the cafeteria one where people are fairly static.

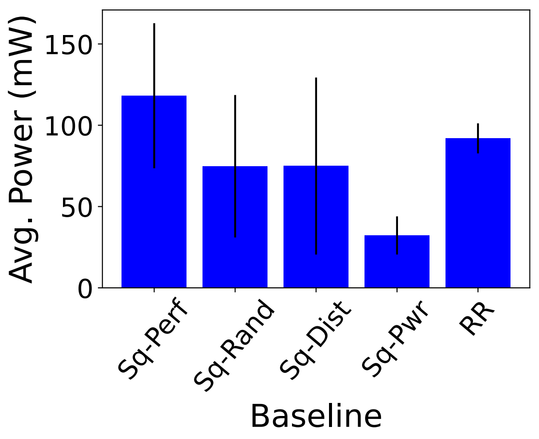
While the runtime configures different voltages for different policies, we find that Squint is able to stream frames seamlessly without any noticeable frame drops. By doing so, Squint is able to provide a consistent real-time pipeline performance of 24 fps for all the evaluated workloads.

Notably, we find that voltage configuration requests executed while the sensor is actively capturing an image results in different parts of the frame possessing different qualities. This is due to rolling shutter sensor operation whereby different lines of pixels are read at different voltage configurations. We notice such artifacts during the dataset collection with more artifacts appearing when rapidly changing voltages. This opens up an opportunity to design more fine-grained voltage scheduling policies that can sample different parts of a frame, as mentioned in §3.3.2. We plan to study the fidelity effects of such policies deeply in a future work.

Algorithms Are Still Reliable We find that task accuracy is fairly maintained for people detection workload, while policies choose different voltage schedules during system execution, as shown in Fig. 24a. Notably, On-Demand has slightly better accuracy than Lighutil because it's adaptive to both light and motion in the scene. Random policy has comparable performance to Lightutil because the system randomly



(a) Task Accuracy



(b) Power

Figure 25: Camera Pose Detection Task.

modulates the sensor supply voltage with an average operating voltage landing in the middle. At this average intermediate voltage, the frame quality is comparable to the frame quality at a nominal voltage. Lower accuracy for Powersave policy is due to continuous low voltage operation, regardless of scene dynamics. On the other hand, reduced resolution (RR) baseline also provides consistent accuracy as neural networks, by design, are typically trained to perform well at low input resolutions, e.g., 300x300 for YOLO.

From Fig. 24a, we also notice that the accuracy varies quite a lot within a policy as indicated by the large error bars. This variation is mainly because of varying scene dynamics pertaining to occlusion, object proximity, and contrast difference in the dataset. Specifically, we notice the difficult scene situations such as people occluding one another, people standing far from the camera, and natural contrast differences due to non-uniform illumination in the room result in lower task accuracy, and vice-versa.

For marker based pose estimation workload, we find the accuracy is also maintained for distance based policy, "Sq-Dist", as the system adapts sensor voltage based on the marker's proximity with respect to the camera, as shown in Fig. 25a. On the other hand, reduced resolution baseline adapts sensor resolution based on marker distance from the camera. We can see that reduced resolution baseline has a slightly larger error than Squint, mainly because higher resolution does not help with pose tracking in dimly-lit office environments unlike Squint based imaging which helps with tracking by adding brightness to the signal.

3.5.2 The Usage of Squint Is Energy-Efficient

We find that Squint imaging leads to significant camera power savings, up to 73% for our evaluated policies, due to aggressive undervolting. The actual amount of power saved directly depends on the voltage schedule chosen by a policy. A sparser voltage schedule with more low voltage operation, such as the one with Lightutil policy under "Bright" light, results in more savings, whereas a denser voltage schedule with more high voltage operation, such as the one with On-Demand policy under "Bright" light, results in relatively less savings. However, if we consider all lighting conditions, Lightutil has an overall higher power consumption than On-Demand, as shown in in Fig. 24b. This is because Lightutil always selects high voltage under low-light conditions, whereas On-Demand selects both high and low voltage based on scene dynamics. We observe similar trends for camera pose estimation workload as can be seen from Fig. 25b.

Fig. 26 shows the experimental power trace corresponding to two different voltages. As we can see, for a given supply voltage, the image sensor alternates between active and idle states with different timing and power profiles for each of those states. While the system switches the sensor's supply voltage, we notice only the power profiles of active/idle states getting shifted keeping their timing intact. Decreasing the active

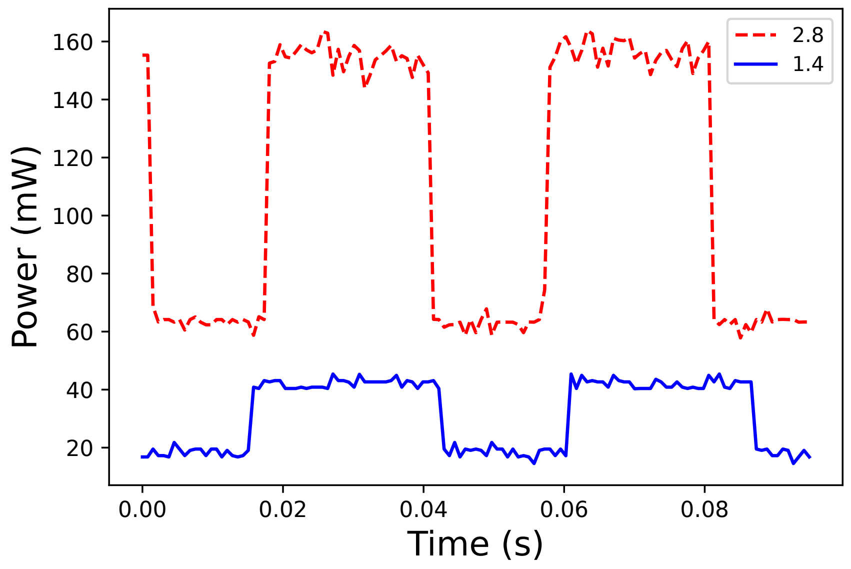


Figure 26: Power Trace for Different Sensor Voltage Modes.

state duration through aggressive standby techniques LiKamWa *et al.* (2013) would lead to even more savings and we will deeply study such mechanisms as future work.

Substantial camera power savings would significantly extend the battery life of battery-backed IoT systems. Specifically, for ultra-low power IoT systems such as the ones built around Cortex-M processors ARM (2022), sensing contributes to a major chunk of system energy. In such systems, a large sensor power savings resulting from Squint imaging techniques could potentially extend the battery life from a few hours to a few days. Such low power profile systems could be powered from energy harvesting sources, e.g., solar cells, and this would eventually pave the way for battery-less systems.

Meanwhile, we find that reducing capture resolution (RR) baseline also results in noticeable sensor power savings. This is because the sensor samples fewer pixels at lower resolutions, thereby reducing burden on analog readout. Image sensors typically adopt a binning mechanism for image subsampling, whereby low-resolution images are achieved by averaging pixels over a window. This means all pixels in the imaging array are activated, typically at a higher resolution, but only a few of them are readout after an averaging process based on the desired resolution. Since readout is far more power hungry than pixels, reducing resolution helps with energy-efficiency. Our voltage scaling techniques could be combined with resolution adaptation techniques to compound energy savings.

3.5.3 The Hardware and Software extensions Are Lightweight

Voltage controller is power and area efficient: Our voltage controller consumes < 1 mW of power during system operation, which is negligibly small in comparison to the camera power savings. We built our voltage controller as a standalone PCB module mainly with three components, which could be easily integrated into existing camera systems. Miniaturization efforts such as building the voltage controller into an integrated circuit form factor using the state-of-the-art fabrication technology would lead to much higher power and area efficiency.

Voltage configuration API is performant: We find that our API takes about 10 ms for switching the camera voltage. Since 10 ms is within a frame period (33 ms), the API enables the system to seamlessly switch sensor voltages on a frame-by-frame basis.

3.6 Related Work

Dynamic voltage scaling in CPUs: The idea to dynamically scale a component's voltage has been extensively used in microprocessors since the early 1990s Weiser *et al.* (1994); Chandrakasan *et al.* (1992); Horowitz (1993). In case of CPUs, undervolting helps conserve processor power Nakai *et al.* (2005) and cools down the chip, but at the cost of performance. Overvolting, on the other hand, helps increase processor performance allowing for higher frequency operation. CPU voltage scaling is usually combined with clock frequency scaling for power and performance gains, and the mechanisms to enable such gains are popularly referred to as DVFS Rabaey *et al.* (1996); Kim *et al.* (2008) in the architecture community.

Similar to CPU undervolting, we undervolt image sensors to conserve power. Furthermore, while undervolting CPUs degrades application's performance, undervolting image sensors degrades application's imaging fidelity. Finally, unlike CPU voltage scaling techniques which adapt the voltage of *digital* circuits, our sensor voltage scaling techniques adapt the voltage of *analog* circuits for significant power savings.

Sensors for power efficiency: Designing image sensors that can operate on lower analog power supplies has been an emerging trend in the image sensors community to promote sensor's power efficiency. Samsung recently designed two such imagers Image Sensors World (2021c,b) with different resolution and pixel sizes Image Sensors World (2021a), which use 2.2 V supply to power all their analog circuitry. Since lowering pixel voltage causes fidelity issues stemming from decreased voltage swing and backflow of charge into photodiodes, several circuit level changes have been done to compensate for those issues.

Many researchers in academia designed power-efficient sensors as well. Choi et al Choi *et al.* (2015b) designed a dual voltage-mode camera that can operate in high-fidelity imaging mode (3.3 V) and low-power vision mode (0.9 V) to save power. Osawa et al Osawa *et al.* (2018) designed a low-power IoT camera that has power consumption proportional to frame rate, using a hierarchical column multiplexer and variable capacitance buffer. LiKamWa et al LiKamWa *et al.* (2013) explore clock scaling and aggressive standby techniques for sensor energy-efficiency. Murakami et al Murakami *et al.* (2022) proposes an approach to dynamically scale frequencies and voltages of image sensor's digital circuits within a frame to save power during vertical blanking period.

Similar to these imagers, we also lower the sensor's analog voltage to save power consumption. Different from these imagers, our undervolting mechanisms do not need explicit changes to the sensor design and can be directly applied on off-the-shelf cameras.

Systems for efficient visual computing: Many researchers proposed visual systems that can programmatically configure different sensors' parameters to achieve desirable system trade-offs. Stagioni Kodukula *et al.* (2021a) adapts the temperature of a 3D stacked sensor by varying the amount of near-sensor processing, to trade thermal noise with system energy. Stagioni migrates the vision task between near-sensor and far-sensor VPU in a duty-cycled fashion to regulate temperature.

Banner Hu *et al.* (2019) adapts sensor resolution on a frame-level granularity to trade task accuracy with system energy, by avoiding repeated memory allocation. Rhythmic Pixel Regions Kodukula *et al.* (2021b) adapts sensor resolution on a region-level granularity for fine-grained accuracy-energy trade-offs, through a encoder-decoder architecture. Our work is in the similar spirit as these works but we reconfigure the sensor's analog voltage to trade sensor energy with imaging fidelity.

3.7 Future Directions

Network friendliness: Many IoT systems deployed in the wild are constrained by network bandwidth and use ultra-low bandwidth protocols such as LoRA to communicate between edge device and server. Our voltage reconfiguration API requires only a few bytes of data per request, thereby not burdening the network interface. To make our system even more network friendly, we will explore better data transmission protocols and intelligent ways to configure voltage. For instance, the system could occasionally send voltage requests that hold for a bunch of frames instead of sending requests on a per-frame basis.

Voltage scheduling policies: While we mainly explore policies that generate coarse-grained voltage schedules enabling fidelity control on a per-frame basis, our system allows generating fine-grained voltage schedules enabling fidelity control on a per-line basis within a frame. Such fine-grained schedules have to be generated in tandem with rolling shutter based camera capture schedules to allow sampling different parts of the frame at different desirable fidelities, which could be interesting for applications such as semantic segmentation. This fine-grained voltage scheduling would also lead to more sensor power savings, and we will investigate policies that would enable such control as future work.

Our policies also operate the sensor at extreme voltage levels as there is no incentive to operate the sensor at an intermediate voltage level for vision task. However, as mentioned in the characterization section, intermediate voltage level might be useful in low-light situations whereby undervolting adds brightness to the signal without adding too much noise. We will investigate the usefulness of intermediate voltage control in applications that can tolerate some accuracy loss as future work.

Dynamic frequency scaling: While we exclusively focus on studying the effects of voltage scaling on the sensor's analog circuitry in this work, as a natural extension, we would like to study the power and performance implications of clock frequency scaling on the sensor's digital circuitry. Notably, we would want to explore how analog voltage scaling would work in tandem with digital clock scaling to provide desired fidelity and frame rate needed for vision applications.

Voltage scaling on other camera types: In this work, we extensively studied the implications of voltage scaling on 2D image sensors. We will extend this study to other sensor modalities such as depth cameras and event cameras. Specifically, we will study how voltage scaling would affect different sensing and readout patterns of off-the-shelf depth and event cameras in the context of emerging 3D vision and eye tracking applications, respectively.

Chapter 4

CONCLUSION

In conclusion, our research on frame-based computing limitations and energy-efficiency has led to the development of two innovative approaches. The first approach introduces an architecture that samples regions with different spatiotemporal rhythms and incorporates a runtime for region selection control. By efficiently encoding and decoding these regions before writing them to memory, we alleviate interface traffic and reduce the memory footprint, ultimately achieving improved system energy-efficiency.

Moreover, our second approach addresses the energy-efficiency limitations imposed by fixed voltage supply to power the analog circuitry of visual computing systems. Through the Squint framework, we enable aggressive undervolting of the sensor's analog voltage, resulting in significant efficiency gains. This is achieved through the integration of a programmable voltage controller hardware and a voltage configuration API software, allowing dynamic adaptation of sensor voltage from vision applications.

By pushing the energy-efficiency boundaries in visual computing, our research has significant implications for various domains such as drones, smartphones, AR, and VR systems. These advancements extend the battery life of these devices, enabling new forms of low-power visual computing. Additionally, our hardware and software interfaces pave the way for imaging-aware voltage scheduling techniques, benefiting IoT and AR systems.

BIBLIOGRAPHY

1996scarlet, “RetinaNet”, https://github.com/1996scarlet/faster-mobile-retinaface (1996).

Adams, A., E.-V. Talvala, S. H. Park, D. E. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico *et al.*, “The frankencamera: an experimental platform for computational photography”, in “ACM SIGGRAPH”, (2010).

Albert Theuwissen, “How to measure the average dark signal ?”, https://harvestimaging.com/blog/?p=795 (2011).

Albert Theuwissen, “How To Measure Temporal Noise in Dark ?”, https://harvestimaging.com/blog/?p=994 (2012).

Android, “Android Camera API documentation”, https://developer.android.com/guide/topics/media/camera (2019).

ARM, “Arm Cortex-M4 Datasheet”, https://www.arm.com/-/media/ArmDeveloperCommunity/PDF/ProcessorDatasheets/ArmCortex-M4ProcessorDatasheet.pdf (2022).

Baker, R. J., *CMOS: circuit design, layout, and simulation* (John Wiley & Sons, 2019).

Bick, C. S., I. Lee, T. Coote, A. E. Haponski, D. Blaauw and D. Ó. Foighil, “Millimeter-sized smart sensors reveal that a solar refuge protects tree snail partula hyalina from extirpation”, Nature’s Communications Biology (2021).

Buckler, M., P. Bedoukian, S. Jayasuriya and A. Sampson, “Eva^2: Exploiting temporal redundancy in live computer vision”, in “ACM/IEEE 45th Annual Int. Symp on Computer Architecture (ISCA)”, (2018).

Cao, Z., G. Hidalgo Martinez, T. Simon, S. Wei and Y. A. Sheikh, “Openpose: Realtime multi-person 2d pose estimation using part affinity fields”, IEEE Transactions on Pattern Analysis and Machine Intelligence (2019).

Chandrakasan, A. P., S. Sheng and R. W. Brodersen, “Low-power cmos digital design”, IEICE Transactions on Electronics (1992).

Chen, T. Y.-H., L. Ravindranath, S. Deng, P. Bahl and H. Balakrishnan, “Glimpse: Continuous, real-time object recognition on mobile devices”, in “Proc. of the 13th ACM Conf. on Embedded Networked Sensor Systems”, (2015).

Chen, Y.-H., T.-J. Yang, J. Emer and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices", IEEE Journal on Emerging and Selected Topics in Circuits and Systems (2019).

Cheng, D., D. K. Prasad and M. S. Brown, "Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution", JOSA A (2014).

Choi, J., S. Park, J. Cho and E. Yoon, "An energy/illumination-adaptive CMOS image sensor with reconfigurable modes of operations", IEEE Journal of Solid-State Circuits (2015a).

Choi, J., J. Shin, D. Kang and D.-S. Park, "Always-on cmos image sensor for mobile and wearable devices", IEEE Journal of Solid-State Circuits (JSSC) (2015b).

Dehmelt, F., "Adaptive (dynamic) voltage (frequency) scaling—motivation and implementation", Texas Instruments – Application Report pp. 1–10 (2014).

Delbracio, M., D. Kelly, M. S. Brown and P. Milanfar, "Mobile computational photography: A tour", Annual Review of Vision Science URL https://doi.org/10.1146/annurev-vision-093019-115521 (2021).

Ebner, M., "Combining white-patch retinex and the gray world assumption to achieve color constancy for multiple illuminants", in "Joint Pattern Recognition Symposium", pp. 60–67 (Springer, 2003).

Ebner, M., "The gray world assumption", Color Constancy; John Wiley & Sons: Chichester, UK pp. 106–108 (2007).

EETimes, "Tensilica's New Vision/AI DSP Guns for SLAM", https://www.eetimes.com/tensilicas-new-vision-ai-dsp-guns-for-slam (2020).

Elgebaly, M. and M. Sachdev, "Variation-aware adaptive voltage scaling system", IEEE Tran. on Very Large Scale Integration (VLSI) Systems (2007).

Feng, Y., P. Whatmough and Y. Zhu, "Asv: accelerated stereo vision system", in "Proc. of the 52nd Annual IEEE/ACM Int. Symp. on Microarchitecture", (2019).

Fielding, R. T., *Architectural styles and the design of network-based software architectures* (University of California, Irvine, 2000).

Fossum, E. R. and D. B. Hondongwa, "A review of the pinned photodiode for ccd and cmos image sensors", IEEE Journal of the electron devices society (2014).

Gallego, Guillermo and Delbruck, Tobi and Orchard, Garrick and Bartolozzi, Chiara and Taba, Brian and Censi, Andrea and Leutenegger, Stefan and Davison, Andrew and Conradt, Jörg and Daniilidis, Kostas and others, "Event-based vision: A survey", arXiv preprint arXiv:1904.08405 (2019).

Ghose, S., A. G. Yaglikçi, R. Gupta, D. Lee, K. Kudrolli, W. X. Liu, H. Hassan, K. K. Chang, N. Chatterjee, A. Agrawal, M. Connor and O. Mutlu, "What your dram power models are not telling you: Lessons from a detailed experimental study", Proc. of the ACM on Measurement and Analysis of Computing Systems (2018).

Gondimalla, A., N. Chesnut, M. Thottethodi and T. Vijaykumar, "Sparten: A sparse tensor accelerator for convolutional neural networks", in "Proc. of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture", (2019).

Google, "Take off to your next destination with Google Maps", https://www.blog.google/products/maps/take-your-next-destination-google-maps (2019).

Gupta, M., A. Agrawal, A. Veeraraghavan and S. G. Narasimhan, "Flexible voxels for motion-aware videography", in "European Conference on Computer Vision", (2010).

Hameed, R., W. Qadeer, M. Wachs, O. Azizi, A. Solomatnikov, B. C. Lee, S. Richardson, C. Kozyrakis and M. Horowitz, "Understanding sources of inefficiency in general-purpose chips", in "Proc. of the 37th annual intl. symp on Computer architecture", (2010).

Hennessey, J. L. and D. A. Patterson, *Computer Architecture: A Quantitative Approach* (Elsevier, 2018).

Ho, R., K. W. Mai and M. A. Horowitz, "The future of wires", Proc. of the IEEE (2001).

Höllerer, T. and S. Feiner, "Mobile augmented reality", Telegeoinformatics: Location-based computing and services **21** (2004).

Horowitz, M. A., "Self-clocked structures for low power systems", ARPA semi-annual report (1993).

Hu, J., A. Shearer, S. Rajagopalan and R. LiKamWa, "Banner: An image sensor reconfiguration framework for seamless resolution-based tradeoffs", in "Proc. of the 17th Annual Int Conf on Mobile Systems, Applications, and Services", (2019).

Hu, J., J. Yang, V. Delhivala and R. LiKamWa, "Characterizing the reconfiguration latency of image sensor resolution on android devices", in "Proc. of the 19th International Workshop on Mobile Computing Systems & Applications", (2018).

Iaian Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia* (2004).

Image Sensors World, "Era of 2.2V Supply Coming?", https://image-sensors-world.blogspot.com/2021/09/era-of-22v-supply-coming.html (2021a).

Image Sensors World, "Samsung Announces 200MP 0.64um Pixel Mobile Sensor", https://image-sensors-world.blogspot.com/2021/09/samsung-announces-200mp-064um-pixel.html (2021b).

Image Sensors World, "Samsung Announces 50MP 1um Pixel Sensor with All-Direction PDAF", https://image-sensors-world.blogspot.com/2021/09/samsung-announces-50mp-1um-pixel-sensor.html (2021c).

Iqbal, O., S. Siddiqui, J. Martin, S. Katoch, A. Spanias, D. Bliss, S. Jayasuriya and S. Center, "Design and fpga implementation of an adaptive video subsampling algorithm for energy-efficient single object tracking", in "IEEE Int Conf on Image Processing", (2020).

Iyer, V., A. Najafi, J. James, S. Fuller and S. Gollakota, "Wireless steerable vision for live insects and insect-scale robots", Science robotics (2020).

Kim, W., M. S. Gupta, G.-Y. Wei and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators", in "IEEE 14th Int Symp. on High Performance Computer Architecture", (2008).

Kitamura, K., T. Watabe, T. Sawamoto, T. Kosugi, T. Akahori, T. Iida, K. Isobe, T. Watanabe, H. Shimamoto, H. Ohtake *et al.*, "A 33-megapixel 120-frames-per-second 2.5-watt cmos image sensor with column-parallel two-stage cyclic analog-to-digital converters", IEEE Transactions on Electron Devices (2012).

Kodukula, V., S. Katrawala, B. Jones, C.-J. Wu and R. LiKamWa, "Dynamic temperature management of near-sensor processing for energy-efficient high-fidelity imaging", Sensors (2021a).

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

LiKamWa, R., B. Priyantha, M. Philipose, L. Zhong and P. Bahl, "Energy characterization and optimization of image sensing toward continuous mobile vision", in "Proc. of the 11th annual international conference on Mobile systems, applications, and services", (2013).

Lin, C.-I., C.-H. Lai and Y.-C. King, “A four transistor cmos active pixel sensor with high dynamic range operation”, in “Proceedings of Asia-Pacific Conference on Advanced System Integrated Circuits”, (IEEE, 2004).

Lu, C., M. Hirsch and B. Scholkopf, “Flexible spatio-temporal networks for video prediction”, in “Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition”, (2017).

Magen, N., A. Kolodny, U. Weiser and N. Shamir, “Interconnect-power dissipation in a microprocessor”, in “Proc. of the international workshop on System level interconnect prediction”, (2004).

Malladi, K. T., F. A. Nothaft, K. Periyathambi, B. C. Lee, C. Kozyrakis and M. Horowitz, “Towards energy-proportional datacenter memory with mobile dram”, in “39th Annual IEEE Int. Symp. on Computer Architecture (ISCA)”, (2012).

Max Planck Institute for Informatics, University of Bonn, “PoseTrack Dataset and Benchmark”, https://posetrack.net/ (2018).

Meindl, J. D., J. A. Davis, P. Zarkesh-Ha, C. S. Patel, K. P. Martin and P. A. Kohl, “Interconnect opportunities for gigascale integration”, IBM journal of research and development (2002).

Micheal Promonet, “v4l2rtspserver”, https://github.com/mpromonet/v4l2rtspserver (2022).

Microchip, “MCP4725, 12-Bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6”, https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/22039d.pdf (2022).

Micron technologies, “Micron system power calculators”, https://www.micron.com/support/tools-and-utilities/power-calc (2019).

Microsoft, “Azure Kinect DK”, https://azure.microsoft.com/en-us/products/kinect-dk (2020).

MIPI Alliance, “MIPI Camera Serial Interface 2 (MIPI CSI-2)”, https://www.mipi.org/specifications/csi-2 (2020).

Mur-Artal, Raúl, Montiel, J. M. M. and Tardós, Juan D., “ORB-SLAM: a versatile and accurate monocular SLAM system”, IEEE Trans. on Robotics (2015).

Murakami, H., E. Bohannon, J. Childs, G. Gui, E. Moule, K. Hanzawa, T. Koda, C. Takano, T. Shimizu, Y. Takizawa *et al.*, “A 4.9 mpixel programmable-resolution multi-purpose cmos image sensor for computer vision”, in “IEEE Int. Solid-State Circuits Conference (ISSCC)”, (2022).

Naderiparizi, S., P. Zhang, M. Philipose, B. Priyantha, J. Liu and D. Ganesan, "Glimpse: A programmable early-discard camera architecture for continuous mobile vision", in "Proc. of the 15th Annual Int. Conf. on Mobile Systems, Applications, and Services", (2017).

Nakai, M., S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano and M. Shimura, "Dynamic voltage and frequency management for a low-power embedded microprocessor", IEEE journal of solid-state Circuits (2005).

NICTA, "ChokePoint Dataset", http://arma.sourceforge.net/chokepoint/ (2015).

OmniVision, "OV5647 datasheet", https://www.uctronics.com/download/Image_Sensor/OV5647_DS.pdf (2022a).

OmniVision, "PYTHON 1.3/0.5/0.3 MegaPixels Global Shutter CMOS Image Sensors", https://www.onsemi.com/pdf/datasheet/noip1sn1300a-d.pdf (2022b).

OpenCV, "OpenCV KeyPoint Class Reference", https://docs.opencv.org/3.4/d2/d29/classcv_1_1KeyPoint.html (2015).

Osawa, M., S. Hiraide, S. Suzuki, H. Kato, K. Tamiya, Y. Harada, K. Raczkowski, J.-L. Bacq, P. Van Wesemael, M. Liu *et al.*, "An adaptive frame image sensor with fine-grained power management for ultra-low power internet of things application", in "IEEE 44th European Solid State Circuits Conference (ESSCIRC)", (2018).

Pandiyan, D. and C.-J. Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms", in "2014 IEEE Int. Syump. on Workload Characterization (IISWC)", (2014).

Rabaey, J. M., A. Chandrakasan and B. Nikolic, "Digital integrated circuits, vol. 996", (1996).

Raghunathan, V., M. B. Srivastava and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication", in "ACM Proc. of the 40th annual Design Automation Conference", (2003).

Reddy, D., A. Veeraraghavan and R. Chellappa, "P2c2: Programmable pixel compressive camera for high speed imaging", in "CVPR", (2011).

Richard Szeliski, *Computer Vision: Algorithms and Applications 1st ed.* (2010).

Sony, "IMX219 datasheet", https://github.com/rellimmot/Sony-IMX219-Raspberry-Pi-V2-CMOS/blob/master/RASPBERRYPICAMERAV2DATASHEETIMX219PQH5_7.0.0_Datasheet_XXX.PDF (2022a).

Sony, "IMX219PQH5-C", https://www.arducam.com/downloads/modules/RaspberryPi_camera/IMX219DS.PDF (2022b).

Stemmer Imaging, "Teledyne DALSA Piranha4 - Dual-Line-CMOS line camera", https://www.stemmer-imaging.com/en/products/series/teledyne-dalsa-piranha4 (2019).

Sturm, J., N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems", in "Proc. of the International Conference on Intelligent Robot Systems (IROS)", (2012).

Synopsys, "Augmenting Your Reality with Deep Learning", https://www.synopsys.com/designware-ip/technical-bulletin/augmenting-your-reality-dwtb_q318.html (2020).

Takayanagi, I., M. Shirakawa, K. Mitani, M. Sugawara, S. Iversen, J. Moholt, J. Nakamura and E. R. Fossum, "A 1.25-inch 60-frames/s 8.3-m-pixel digital-output cmos image sensor", IEEE Journal of Solid-State Circuits (2005).

Texas Instruments, "OPA55x High-Voltage, High-Current Operational Amplifiers", https://www.ti.com/lit/gpn/OPA551 (2022).

Tom's guide, "Magic Leap 2 release date, price and all the new features", https://www.tomsguide.com/news/magic-leap-2-everything-weve-heard-so-far (2022).

Van De Weijer, J., T. Gevers and A. Gijsenij, "Edge-based color constancy", IEEE Transactions on image processing (2007).

Vasisht, D., Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan and S. Stratman, "{FarmBeats}: An {IoT} platform for {Data-Driven} agriculture", in "14th USENIX Symp. on Networked Systems Design and Implementation (NSDI 17)", (2021).

Vogelsang, T., "Understanding the energy consumption of dynamic random access memories", in "43rd Annual IEEE/ACM Int Symp. on Microarchitecture", (2010).

Weiser, M., B. Welch, A. Demers and S. Shenker, "Scheduling for reduced cpu energy", in "Mobile Computing", pp. 449–471 (Springer, 1994).

Wikipedia, "Foveated Rendering", https://en.wikipedia.org/wiki/Foveated_rendering (2020).

Wikipedia, "Representational state transfer", https://en.wikipedia.org/wiki/Representational_state_transfer (2022).

Xilinx, "reVISION Getting Started Guide 2018.3 (UG1265)", https://github.com/Xilinx/reVISION-Getting-Started-Guide (2018).

Xilinx, “Vivado Design Suite”, https://www.xilinx.com/products/design-tools/vivado.html (2019a).

Xilinx, “Xilinx Power Estimator”, https://www.xilinx.com/products/technology/power/xpe.html (2019b).

Xilinx, “H.264/H.265 Video Codec Unit v1.2”, https://www.xilinx.com/support/documentation/ip_documentation/vcu/v1_2/pg252-vcu.pdf (2020a).

Xilinx, “Zynq DPU v3.2”, https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_2/pg338-dpu.pdf (2020b).

ximea, “Multiple ROI cameras”, https://www.ximea.com/support/wiki/allprod/Multiple_ROI (2019).

Zhang, Z., H. Wang, S. Han and W. J. Dally, “Sparch: Efficient architecture for sparse matrix multiplication”, in “IEEE Int. Symp. on High Performance Computer Architecture (HPCA)”, (2020).

Zhu, Y., A. Samajdar, M. Mattina and P. Whatmough, “Euphrates: Algorithm-soc co-design for low-power mobile continuous vision”, ISCA (2018).

Ziyadinov, V. and M. Tereshonok, “Noise immunity and robustness study of image recognition using a convolutional neural network”, Sensors (2022).

APPENDIX A

FRAME STACKS AND ENERGY MODEL

A.1 Rhythmic Pixel Regions in Action

In Figs. A.1-A.6, we show the progression of frames for different workloads in action.

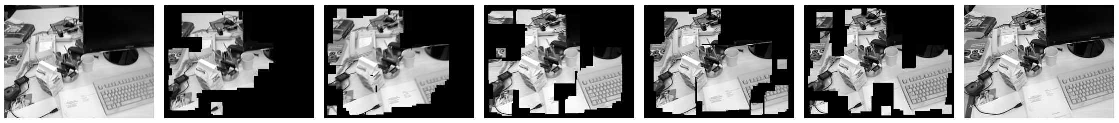

(a) Frame 1 (100%) (b) Frame 2 (37%) (c) Frame 3 (31%) (d) Frame 4 (34%) (e) Frame 5 (27%) (f) Frame 6 (35%) (g) Frame 7 (100%)

Figure 27: Task: Visual SLAM; Benchmark: TUM freiburg1-xyz

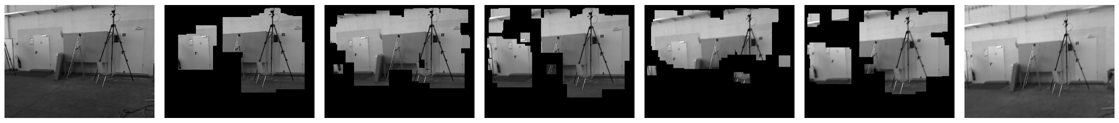

(a) Frame 1 (100%) (b) Frame 2 (42%) (c) Frame 3 (36%) (d) Frame 4 (39%) (e) Frame 5 (33%) (f) Frame 6 (40%) (g) Frame 7 (100%)

Figure 28: Task: Visual SLAM; Benchmark: TUM freiburg1-floor

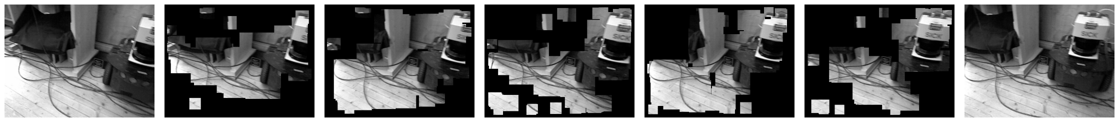

(a) Frame 1 (100%) (b) Frame 2 (44%) (c) Frame 3 (41%) (d) Frame 4 (37%) (e) Frame 5 (32%) (f) Frame 6 (35%) (g) Frame 7 (100%)

Figure 29: Task: Visual SLAM; Benchmark: TUM freiburg2-360-kidnap-secret

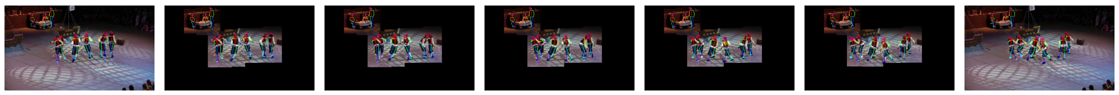
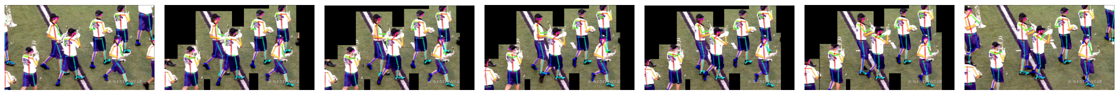
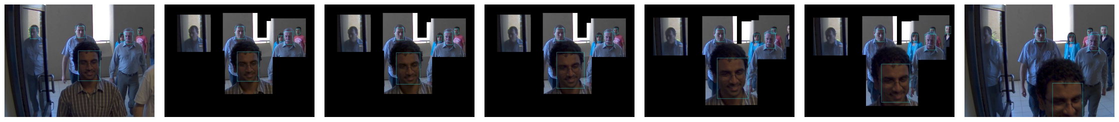

(a) Frame 1 (100%) (b) Frame 2 (31%) (c) Frame 3 (24%) (d) Frame 4 (29%) (e) Frame 5 (23%) (f) Frame 6 (33%) (g) Frame 7 (100%)

Figure 30: Task: Human pose estimation; Benchmark: PoseTrack 2017 024575-mpii



(a) Frame 1 (100%) (b) Frame 2 (39%) (c) Frame 3 (23%) (d) Frame 4 (36%) (e) Frame 5 (21%) (f) Frame 6 (41%) (g) Frame 7 (100%)

Figure 31: Task: Human pose estimation; Benchmark: PoseTrack 2017 015301-mpii



(a) Frame 1 (100%) (b) Frame 2 (28%) (c) Frame 3 (26%) (d) Frame 4 (22%) (e) Frame 5 (37%) (f) Frame 6 (31%) (g) Frame 7 (100%)

Figure 32: Task: Face detection; Benchmark: Chokepoint dataset P2E-S5

Table 8: Energy-per-Pixel of Various Components in the Vision Pipeline. As Is Widely Acknowledged, Communication Cost Is At Least Three Orders of Magnitude More Than Compute Cost.

Component	**Energy (pJ/pixel)**
Sensing	595 LiKamWa *et al.* (2013); Choi *et al.* (2015a)
Communication (SoC - DRAM)	2800 Xilinx (2019b); Pandiyan and Wu (2014); Ho *et al.* (2001); Raghunathan *et al.* (2003); Magen *et al.* (2004); Meindl *et al.* (2002)
Storage	677 Micron technologies (2019); Ghose *et al.* (2018); Malladi *et al.* (2012); Vogelsang (2010)
Computation (per MAC)	4.6 Hameed *et al.* (2010)

A.2 Energy Model

The power measurements directly taken off from the Xilinx Zynq FPGA board, which consumes power on the order of tens of watts, are not representative of mobile systems, which consume only a few watts of power for typical vision tasks. Therefore, we construct a coarse energy model based on reported numbers in component datasheets and computer architecture literature to estimate system power. Precise energy savings will depend on system characteristics that are highly specific to the device, operating system, and application usage pattern. We construct the first-order linear model to approximate power consumption so as to contextualize the benefits of reducing pixel memory throughput in a mobile system.

In particular, we break down the system energy into four components: sensing, computation, storage, and communication, as shown in Table 8. Sensing requires an energy of roughly 600 pJ/pixel LiKamWa *et al.* (2013); Choi *et al.* (2015a), mostly drawn from three components: pixel array, read-out circuits, and analog signal chain. DRAM storage on standard mobile-class memory chips (8 Gb, 32-bit LPDDR4) draws

677 pJ/pixel for writing and reading a pixel value Micron technologies (2019). This roughly divides into 300 pJ/pixel for reading and 400 pJ/pixel for writing Ghose *et al.* (2018); Malladi *et al.* (2012); Vogelsang (2010). Communication over DDR interfaces incur 4 nJ/pixel, mostly due to operational amplifiers on transmitters and receivers. We measure the interface power dissipation Xilinx (2019b) on 4-lane CSI interfaces and LPDDR4 interfaces by inputting several datarates. From this information, we construct a linear-regression model to estimate the energy per pixel to be 1 nJ/pixel over CSI and 3 nJ/pixel Pandiyan and Wu (2014); Ho *et al.* (2001); Raghunathan *et al.* (2003); Magen *et al.* (2004); Meindl *et al.* (2002) over DDR. We use 5 pJ per MAC operation Hameed *et al.* (2010) to estimate compute energy.

We use FPGA measurements of pixel memory read/write throughput and computation patterns to apply the runtime behavior of the system to our energy model. For communication and storage, we take memory traffic/footprint values from Fig 7 and apply them to the model. We obtain sensing energy by applying frame resolution from Table 3 on our model. For compute energy, we take the number of MAC operations for our CNN workloads and apply them to the model.