Rhythmic Pixel Regions: Visual sensing architecture for flexible spatiotemporal resolution towards high-precision visual computing at low power Extended Abstract

Venkatesh Kodukula, Alexander Shearer, Van Nguyen, Srinivas Lingutla, and Robert LiKamWa
Arizona State University, Tempe, AZ, USA
{*vkoduku1, acshear1, vtnguy19, slingutl, likamwa*}@asu.edu

1. Motivation

High-precision visual computing on traditional frame-based architectures creates energy-hungry memory traffic. Visual sensing at 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, face identification, and object classification. Unfortunately, capturing and processing high spatiotemporal visual *frames* generates energy-expensive memory traffic due to the overwhelming data rate of pixel throughput e.g., requiring nearly 12 Gbps for 4K uncompressed frames at 60 fps. On the other hand, low resolutions can reduce memory throughput, but also prohibit high-precision visual sensing.

However, our intuition is that not all parts of the scene need to be captured at a uniform resolution. Far and fast moving objects and areas of a frame may require high spatiotemporal resolutions, but close and/or static objects can be captured with lower resolutions without impairing accuracy. We propose that *rhythmic pixel regions* with independently controllable spatial and temporal resolutions can yield high-precision visual computing at energy-efficient data rates. Around this idea, we design a visual computing pipeline that discards unnecessary pixels before they reach DRAM memory, while preserving frame access abstractions for app developers.

Unlike visual computing based on a few Regions-of-Interest (ROIs), rhythmic pixel regions leverage encoded data representations that scalably allow for the capture of hundreds of regions, with a multitude of independently foveated spatiotemporal resolutions. Our architectural support allows developers to dynamically define the region selection, allowing them to drive the rhythmic pixel region configuration by visual features that are already being computed for augmented reality, face detection, and/or object recognition workloads.

2. Limitations of the State of the Art

Multi-ROI cameras do not scale with number of regions: Many image sensors are capable of selecting a ROI for readout [4]. However, to the best of our knowledge, these sensors support at most 16 regions, and those regions are typically captured at a uniform resolution and frame rate. In contrast, our architectural support for rhythmic pixel regions can uniquely support hundreds of regions with different spatio-temporal resolutions, allowing for rich configurability and adaptivity to match the variability of the spatial environment.

Event-driven cameras do not scale with resolution: Event-driven cameras focus their sampling on pixels that change their value [1]. However, the per-pixel circuitry has a motion detection module making it spatially expensive, thereby limiting the frame resolution and spatial precision, e.g., to 128 x 128 pixels. In contrast, our architecture works with off-the-shelf mobile cameras which can support higher resolutions and can flexibly deduce the regions based on developer-defined responses to visual information.

Video compression techniques are memory-inefficient: Commercial video codecs, such as H.264, reduce redundant information by leveraging estimated motion information from frame-to-frame [2]. 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.

3. Key design nuggets

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. We prioritize hardware and software support for *flexibility* and *scalability* to uniquely enable support for hundreds of regions with independent spatiotemporal resolutions.

3.1. Rhythmic pixel encoder architecture

The **rhythmic pixel encoder module** intercepts the incoming pixel stream from a conventional image sensor pipeline and uses developer-specified region labels to encode pixels into an encoded frame before writing them to DRAM. As shown in Fig. 5, the encoder selectively samples pixels from the camera stream based on whether the pixel is in any of the regions and matches the region's specified spatiotemporal resolution stride.

Raster-scan optimized sampling: A naive approach would fully parallelize the region label checking process, which would exponentially increase the resource footprint, limiting encoder's scalability. Our approach is to instead exploit the raster-scan patterns of the incoming pixel stream to reduce and reuse the work of the region search.

There are three key techniques: **(a) Search-space reduction:** 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. **(b) Spatial locality:** Within a row, if we find that if a pixel belongs to a region, we can apply the same comparison result for the next

region width number of pixels. **(c) Pre-sorting:** We also simplify the process of finding relevant regions for a row through sorting the regions in the order of y-indices. This can be done either at the application level or at the OS kernel level.

3.2. Rhythmic pixel decoder architecture

The **rhythmic pixel decoder module** fulfills pixel requests from the vision application. This request path is managed by a pixel memory management unit (PMMU), as shown in Fig. 6. The return path returns the pixel values to the vision application through a Resampler Unit.

Pixel Memory Management Unit: The PMMU works in the same spirit as a traditional memory management unit (MMU) but has the responsibility of translating decoded ("original") pixel addresses to encoded pixel addresses stored in DRAM. It has three key elements. The *out-of-frame handler* forwards the transaction if it is requesting a pixel; otherwise, it will bypass, allowing for standard memory access. The *transaction analyzer* analyzes the encoding sequence of the transaction and generates different sub-requests based on where the encoded pixels are present, as we explain below. These sub-requests are fed to a *translator* which performs the conversion from orignal to encoded space.

Resampling Unit: The interpolation engine uses a FIFO to buffer data packets received from a pixel-based DRAM transaction. To prepare a pixel value to send back to the original request, the engine either dequeues the pixel data from the FIFO, re-samples the previous pixel (in the case of stride), or samples a black pixel, based on the combined bitmask information.

3.3. 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. This allows fully transparent use of existing vision software libraries and hardware accelerators, with no modification necessary.

We develop runtime support to allow the developers to flexibly specify region labels. A runtime service receives function calls to send the region label list to the encoder via memory-mapped registers. Region label lists can be set on a per-frame basis or persist across frames.

Policy-based usage of rhythmic pixel regions Developers can build various policies that autonomously guide the region selection. A feature-based policy can use proxies such as feature scale and displacement to estimate the spatial and temporal resolutions of regions.

The overall process of policy generation and modifying the application around that policy could be cumbersome for a general application developer. To reduce the developer burden, we propose two tiers of developers: *Policy makers* can design feature-based, accelerometer-based, or other forms of adaptive policies that dynamically select region labels. *Policy users* can select policies from a pool to serve their application needs while reaping the benefits of rhythmic pixel regions.

4. Implementation and key results

FPGA based implementation: We integrate our hardware and sofware extensions on top of Xilinx's reVISION stack platform [3] which implements an end-to-end video pipeline. We evaluate three workloads: (a) Visual SLAM (b) Face detection (c) Human pose estimation.

Evaluation Results: We summarize important results:

- Rhythmic pixel regions use is *flexible*, supporting large numbers of regions, e.g., 1000 regions for V-SLAM with independent resolutions
- Vision apps are still *reliable*, with algorithms operating with minimal and controllable loss in accuracy
- Using rhythmic pixel regions is *memory friendly*. With early pixel discard, we find a significant reduction (roughly 50%) in memory throughput for our evaluated workloads.
- Our hardware extensions are *scalable*, supporting increasing numbers of regions without needing significantly more transistors
- Our extensions are *performant*, as the end-to-end system pipeline runs in real-time, servicing 2 pixels per clock.
- Our hardware extensions are *power-efficient*, consuming a few 10's of mW on an **FPGA** target

5. Why ASPLOS

With a focus on architectural support through our encoder/decoder hardware extensions and software runtime support, rhythmic pixel regions emphasizes the synergy of two areas ("AS" and "OS") to fit with ASPLOS interests.

6. Citation for Most Influential Paper Award

This paper introduces *rhythmic pixel regions*, a paradigm that enables unprecedented visual precision on mobile systems by eliminating energy-expensive memory throughput where spatiotemporal resolution is not needed by visual tasks, while preserving detail where necessary. The presented architectural solutions enable developers to specify the capture and processing of hundreds of visual regions with independent spatiotemporal resolution that match the needs of visual algorithms and apps. As such, this work is a fundamental enabler for high-precision, energy-efficient and performant visual computing for augmented reality, object recognition, face detection, and other futuristic use cases.

7. Revisions

Compared to a previous submission, we have redesigned architectural elements to promote lightweight and scalable encoding/decoding. We have added more comparative baselines (multi-ROI cameras and video compression) to the evaluation.

References

[1] 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.

[2] Iaian Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. 2004.

[3] Xilinx. reVISION Getting Started Guide 2018.3 (UG1265). https://github.com/Xilinx/reVISION-Getting-Started-Guide.

[4] ximea. Multiple ROI cameras. https://www.ximea.com/support/wiki/allprod/Multiple_ROI.