Algorithm-SoC Co-Design for Energy-Efficient Continuous Vision*

Yuhao Zhu[r] Anand Samajdar[g] Matthew Mattina[a] Paul Whatmough[a]

[r]University of Rochester [g]Georgia Institute of Technology [a]ARM Research

1. Introduction

Domain-specific architectures (DSAs) are widely recognized as a promising solution to the Dark Silicon challenge. Due to the high non-recurring engineering (NRE) cost of designing custom architectures and hardware chips, we must identify key application domains whose demands are large enough and whose social impacts are high enough to justify the efforts of designing DSAs. We focus on the domain of *continuous computer vision*, which encapsulates a class of emerging applications that extract visual insights from real-time camera streams to guide high-level decision making. Continuous vision applications are widely deployed in both consumer mobile devices and "always-on" embedded systems such as Unmanned Aerial Vehicles, Augmented Reality, and smart-city sensing.

Continuous vision applications impose an enormous compute requirement, which in turn consumes excessive energy. Consider a typical mobile power budget of 3W; continuous vision algorithms that operate on 720p (1280×720) resolution images at 30 frames per second (FPS) can only afford an energy budget of about 110 nJ/pixel. In contrast, today's CNN accelerators consume about 10,000 nJ/pixel [6], indicating an energy gap of two orders of magnitude.

Today's most prevalent efforts to close this energy gap, have largely focused on optimizing hardware (micro-)architecture for the vision kernels of interest, with little regard to the wider-system. Fundamentally, these approaches treat frames in real-time video streams as independent entities, and optimize the execution efficiency of each frame. In contrast, this work takes a step back and re-examines the inherent execution model of continuous vision at the system-level. In particular, we harness a key trait of continuous vision: visual information changes only *incrementally* across frames in a real-time video stream. Therefore, the vision results can be incrementally computed from one frame to another without having to re-execute the entire vision algorithm on each frame. We propose a new algorithm that encodes frame pixel changes as object motion, and leverages the motion data to simplify the vision computation for the majority of real-time frames.

Along with the new algorithm is the co-designed Systems-on-Chip (SoC) architecture. To maximize the efficiency of the incremental vision computation, the new SoC architecture exploits the algorithmic synergies between different vision SoC components. Specifically, we observe that the pixel motion information is naturally generated by the Image Signal Processor (ISP) for temporal de-noising, which is performed early in the vision pipeline. We propose lightweight SoC augmentations that enable reuses of the motion data between the ISP and the vision algorithm with little compute overhead.

Our work embodies two key themes of optimizing mobile continuous vision. First, re-examining the system-level execution model of continuous vision unlocks hidden dimensions for optimization. Instead of executing the same algorithm on all the frames, we identify incremental computation as an energy-efficient execution model. Second, expanding the architecture design scope from an individual accelerator (e.g., a CNN engine) to the entire SoC exposes lucrative optimization opportunities. Our work exploits the synergies between the two key IP blocks in a vision SoC–the ISP and the vision engine–and simplifies the SoC design.

2. Euphrates Design

We first introduce motion-based incrementation computation and describe how it applies to continuous vision. We then discuss the SoC support for the algorithm. Our algorithm-SoC co-designed system is called EUPHRATES.

2.1. Motion-based Incremental Computation

Frames in a real-time video stream are not independent. Instead, pixel changes are correlated in time, due to visual object motion. Our algorithm leverages the pixel *motion* information to incrementally execute vision algorithms so as to greatly reduce the total compute requirement. This motion-based incremental computation is formulated as follows:

$$f(x_t) = f(x_{t-1}) \oplus \delta(x_t, x_{t-1})$$

where x_t is the current frame, x_{t-1} is the previous frame, δ denotes the operation that calculates the increments between two frames (i.e., motion), and $\oplus$ denotes the operation that produces the vision results for the current frame by combining the previous frame's result with the pixel motion data. If both the δ and $\oplus$ operations are computationally cheaper than the original vision algorithm $f(\cdot)$, the vision results for the current frame can be calculated in a much more efficient way.

While incremental computation in general is a technique used in program analysis and optimization [3, 4], our work applies it to the domain of continuous vision. We demon-

*The original article is "*Euphrates: Algorithm-SoC Co-Design for Low-Power Mobile Continuous Vision*" by Yuhao Zhu, Anand Samajdar, Matthew Mattina, and Paul Whatmough, published in 45th International Symposium on Computer Architecture (ISCA), June 2018, 547-560.

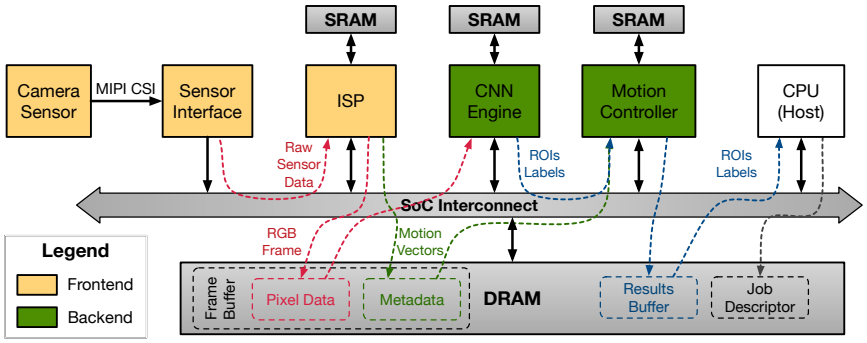


Fig. 1: Block diagram of the augmented continuous vision subsystem in a mobile SoC.

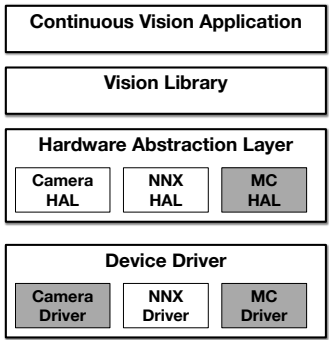


Fig. 2: Vision software stack with modifications shaded.

strate one set of δ and $\oplus$ operations that perform well in practice. In particular, we propose to encode $\delta(x_t, x_{t-1})$ as motion vectors [2], and to realize the $\oplus$ operation through motion extrapolation, which is six orders of magnitude lighter than executing a full CNN inference (i.e., $f(\cdot)$).

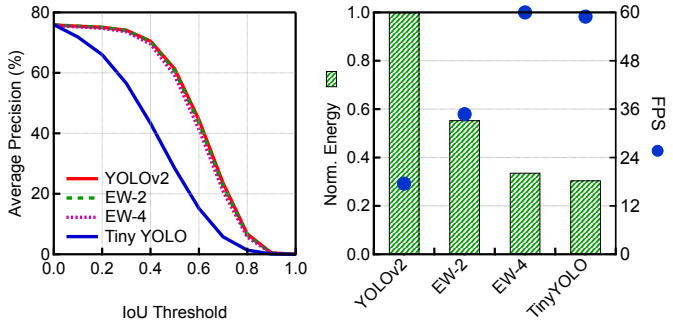
2.2. SoC Architecture Support

The continuous vision pipeline in the SoC must act autonomously to avoid constantly interrupting the CPU which needlessly burns CPU cycles and power. This design principle motivates us to provide autonomous architectural support for the new algorithm, rather than implementing it in software, to remove the CPU from the frame processing loop.

Fig. 1 illustrates the augmented mobile SoC architecture. The key to our SoC co-design is the observation that motion vectors, although expensive to compute, are already calculated by the ISP early in the vision pipeline, and thus are cheap to re-use at the system-level. We augment the ISP to directly expose the motion vectors to the vision engine such that the δ operation is performed "for free." This strategy exploits the algorithmic synergies between the ISP and the vision engine to avoid redundant computations while simplifying SoC design.

Further, we propose a new IP block called the motion controller, which implements the $\oplus$ operation (i.e., extrapolation) in the algorithm. It is important that we do not directly bake the extrapolation operation into a particular CNN accelerator. The rationale is that CNN accelerators are still evolving rapidly with new models and architectures constantly emerging. Tightly coupling our algorithm with any particular CNN accelerator is inflexible in the long term. Our partitioning accommodates future changes in inference algorithm, hardware IP (accelerator, GPU, etc.), or even IP vendor. The motion controller also coordinates the backend under the new algorithm without significant CPU intervention.

EUPHRATES makes no changes to application developers' programming interface and the CV libraries as shown in Fig. 2. The new motion controller (MC) needs to be supported at both the Hardware Abstraction Layer (HAL) and driver layer. In addition, the camera driver is minimally enhanced to configure the base address of motion vectors.



(a) Average precision comparison. (b) Energy and FPS comparison.

Fig. 3: Average precision, normalized energy consumption, and FPS comparisons between various object detection schemes.

3. Euphrates Results

We base our evaluation on a mix of real-system measurements and RTL implementation. We model after the Nvidia's TX2 mobile SoC, from which we obtain power measurements. We implement in RTL the SoC augmentations that are unavailable on TX2 and perform synthesis, placement, and layout.

We evaluate EUPHRATES on two main continuous vision use-cases: visual tracking and object detection. Due to the space limit, here we show only the results of object detection. Fig. 3a shows the average precision (AP) between the state-of-the-art object detector CNN YOLOv2 and EUPHRATES under two different extrapolation window sizes (2 and 4). EW-2 and EW-4 both achieve an AP close to the baseline YOLOv2, represented by the close proximity of their corresponding curves. Specifically, under an IoU of 0.5, a commonly acceptable detection threshold, EW-2 loses only 0.58% accuracy compared to the baseline YOLOv2.

Fig. 3b shows the energy consumptions of different mechanisms normalized to the baseline YOLOv2. We overlay the FPS results on the right y-axis. The baseline YOLOv2 consumes the highest energy and achieves only about 17 FPS, far from being real-time. EW-2 reduces the total energy consumption by 45% and improves the frame rate from 17 to 35; EW-4 reduces the energy by 66% and achieves real-time frame rate at 60 FPS. Overall, EUPHRATES achieves significant energy savings and speedup with little accuracy loss.

4. Long-Term Impact

Our paper has both algorithmic and architectural impact.

Motion-based Incremental Computation Our work demonstrates the effectiveness of motion-based incremental computation as a fundamental computing paradigm for continuous vision applications. However, we see a much broader scope in the general domain of *visual computing*, which encapsulates a wide variety of applications that operate on visual information such as images and videos. Table 1 lists different visual computing applications in which incremental computation is either inherently part of the algorithms or can be applied as an effective optimization.

These applications operate on a sequence of (real-time) images as inputs, in which pixel changes in consecutive frames represent motion. For instance, temporal denoising removes the noise of the current frame using pixel motion data along with the noise-free counterparts in the previous frame. Similarly, frame upsampling artificially increases the frame rate by interpolating new frames between successive real frames based on object motion. Recently developments of Timewarp and Spacewarp in Virtual Reality (VR) use motion information to generate frames that sustain real-time requirements while lowering the compute demand for VR devices.

We thus see a promising direction of research to provide principled architecture support for motion-based incremental computation. Critically, in applying incremental computation, these applications differ in two fundamental aspects: how the motion data is obtained (i.e., the motion estimation operator, δ) and how the motion data is used to generate new results (i.e., the motion synthesis operator, $\bigoplus$). The key architectural challenge is thus how to generally support different motion estimation and synthesis operators.

A potential solution is to consolidate different motion estimation and synthesis operators into dedicated IP blocks that can then be reused across different algorithms. For instance, the motion controller IP proposed in our paper implements one particular form of synthesis operator (extrapolation), and could be used as the basis for a generic "motion processor."

Expanding the Research Scope From Accelerators to SoCs By re-examining and optimizing the execution model of continuous vision, we promote a whole-of-system approach that exploits the synergies of various SoC components rather than focusing only on one particular accelerator. Specifically, EUPHRATES exploits the data reuse between the ISP and vision engine to improve compute-efficiency and simplify SoC design. As we enter the era of specialization where mobile SoCs integrate tens of domain-specific IP blocks [1, 5], we must expand the research horizon from individual accelerator optimizations to a holistic co-design of the entire SoC.

Today's mobile SoCs encompass five main application domains that are traditionally considered separately: imaging, vision, graphics, video processing, and acoustics. Although these components exist in mobile devices such as mobile phones, they are mostly isolated from one another throughout the system stack and rarely interact in an application. As a result, today's mobile SoCs are largely a collection of segregated sub-systems (accelerators), each specialized for a domain.

Table 1: Applications where motion-based incremental computation applies. The EUPHRATES work demonstrates the idea on one application domain, but the general idea is broadly applicable.

Domain	Application	Motion Operator (δ)	Synthesis Operator ($\bigoplus$)
Vision	Object Tracking/Detection	Motion Vector	Extrapolation
Imaging	Temporal Denoising	Motion Vector	Motion-Compensated Temporal Filtering
	High Dynamic Range	(Sub)pixel-level Motion Vector	Image Alignment
	Rolling Shutter Correction	Scanline-level Motion Vector	Motion Compensation
Video Processing	Video Encoding	Motion Vector	Motion Compensation
	Video Stabilization	Egomotion Vector	Motion Compensation
Display	Frame Upsampling	Motion Vector	Interpolation
	Spacewarp (in VR)	Motion Vector	Extrapolation
Graphics	Timewarp (in VR)	Egomotion Vector	Reprojection

Emerging applications such as Augmented Reality and Virtual Reality require the five domains to operate collectively and synergistically. For instance, an Augmented Reality application must perform imaging to convert camera sensor data to continuous frames, trigger vision algorithms to extract semantics information from the frames, and draw virtual graphical contents to augment the real scene, all the while producing an immersive acoustic experience and encoding the generated video content for future playback. The fusion of the five computing domains presents unprecedented opportunities to build mobile SoCs that co-optimize the five domains simultaneously.

This work demonstrates one particular form of co-design between imaging and vision. We hope that it is just the promising first step toward an exciting era of SoC architecture research.

Citation in 2029 In their landmark paper on algorithm-SoC co-design for mobile continuous vision, the authors pioneered the work on SoC support for motion-based incremental computation. Today, hardware support for motion-based incremental computation has become the norm in mainstream mobile SoCs. The publication also spurred numerous proposals on co-optimize SoC components for application domains beyond the paper's original focus on continuous vision.

References

[1] "NVIDIA Reveals Xavier SOC Details." https://www.forbes.com/sites/moorinsights/2018/08/24/nvidia-reveals-xavier-soc-details/amp/

[2] M. Jakubowski and G. Pastuszak, "Block-based Motion Estimation Algorithms–A Survey," *Opto-Electronics Review*, 2013.

[3] D. Michie, ""memo" functions and machine learning," *Nature*, vol. 218, no. 5136, p. 19, 1968.

[4] W. Pugh and T. Teitelbaum, "Incremental computation via function caching," in *Proc. of POPL*, 1989.

[5] Y. S. Shao *et al.*, "The aladdin approach to accelerator design and modeling," *IEEE Micro*, vol. 35, no. 3, pp. 58–70, 2015.

[6] A. Suleiman *et al.*, "Towards Closing the Energy Gap Between HOG and CNN Features for Embedded Vision," in *Proc. of ISCAS*, 2017.