DOI: 10.35598/mcfpga.2026.034

Optimization of a Real-Time Edge-Detection Pipeline on an ARM Cortex-M55 Microcontroller

Kyryll Filippenkov
ORCID 0009-0008-6720-5728
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
filippenkovkyryll@gmail.com

Iryna Zeleneva
ORCID 0000-0002-4042-4540
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
irina.zeleneva@gmail.com

Tetiana Holub
ORCID 0000-0001-6024-008X
Computer Systems and Networks Department
National University "Zaporizhzhia Polytechnic"
Zaporizhzhia, Ukraine
golub.tv6@gmail.com

Abstract*—The migration of computer-vision workloads from the cloud to battery-powered edge devices (Edge Computing) demands that low-power microcontrollers extract the maximum performance from their hardware. Modern Armv8.1-M cores such as the Cortex-M55 add the Helium (M-profile Vector Extension, MVE) SIMD unit, but the auto-vectorizers of current compilers exploit only the simplest loops and miss patterns that are critical for image processing – gather/scatter loads, tail predication and fused multiply-accumulate. This paper presents a hardware-oriented optimization of a classic three-stage edge-detection pipeline – monochrome conversion, Gaussian blur, and the Sobel operator with thresholding – on an Infineon PSOC Edge E84 (Cortex-M55). Six complementary methods are applied: fixed-point integer arithmetic, separable convolution kernels, center/edge loop versioning, manual Helium MVE intrinsic vectorization, placement of hot data and code in tightly-coupled memory (TCM), and an L1 gradient metric with a dynamic threshold. Measured on a 128×128 image with the PMU, the full pipeline is accelerated 10.1× at -O2 and 7.6× at -O3 relative to the scalar baseline, while preserving visual quality. The pipeline is demonstrated in real time on an OV7675 camera at QVGA and VGA resolutions.

Keywords*—ARM Cortex-M55, Helium MVE, SIMD, edge detection, Sobel operator, Gaussian blur, separable kernel, fixed-point arithmetic, tightly-coupled memory, embedded vision, real-time, PSOC Edge E84

I. Introduction

The current stage of information and communication technology is marked by a pronounced migration of computation away from the cloud and back toward end-user devices and the edge of the network – the edge-computing trend. Applied to artificial intelligence and computer vision it is known as Edge AI and Edge Vision. According to Gartner, by 2025 around 75% of enterprise-generated data will be processed outside traditional centralized data centers and the cloud, compared with less than 10% in 2019 [1]. Computer vision is at the forefront of this shift because image sensors are among the most data-intensive sources: a single QVGA sensor produces 76 800 pixels × 30 fps ≈ 2.3 megapixels per second – a stream that is impractical to offload continuously but entirely tractable to process locally given adequate optimization.

Microcontroller-class hardware has expanded rapidly to meet edge-AI demand. The Helium M-profile Vector Extension (MVE) of Armv8.1-M was announced in 2019 [2]; the first core to implement it, the Cortex-M55, together with the Ethos-U55 micro-NPU, followed in 2020 [3]; and the Infineon PSOC Edge E84 (Cortex-M55 + Ethos-U55) reached the market in 2024 [4]. Exploiting this new hardware is not automatic, however: the auto-vectorizers of current compilers (gcc 14, clang 18) emit Helium instructions only for the simplest loops and routinely miss patterns that are decisive for computer-vision throughput – gather/scatter loads, tail predication and specialized multiply-accumulate (MAC) instructions. Significant speed-ups therefore still require manual programming with intrinsics combined with algorithmic optimization.

The aim of this work is to design, implement and quantitatively evaluate a vectorized edge-detection pipeline – monochrome conversion, Gaussian blur, and the Sobel operator with thresholding – on a Cortex-M55 microcontroller using Helium MVE, and to demonstrate it in real time on a camera module. Because the pipeline is purely sequential, it is a convenient benchmark: each stage can be optimized in isolation and its contribution measured separately.

II. Target Platform and Baseline Implementation

A. PSOC Edge E84 and Helium MVE

The PSOC Edge E84 is a heterogeneous SoC with three Arm cores: a Cortex-M55 application core with Helium MVE, and two Cortex-M33 cores (secure and non-secure) that handle boot, TrustZone and peripheral initialization [13], [14]. Helium provides 128-bit vector registers – eight uint16_t, sixteen uint8_t or four uint32_t/float lanes – eight architectural registers Q0–Q7 shared with the FPU, beat-based execution, lane predication for loop tails, gather/scatter memory access, and accumulating MAC instructions (vmlaq) [2], [7]. Crucially for real-time determinism, the core provides tightly-coupled memory (TCM): 256 KB ITCM for code and 256 KB DTCM for data, both with zero-wait-state access [9], alongside 5 MB of shared SOCMEM. Table I compares it with two alternative embedded-vision platforms.

Table I. Embedded-Vision Hardware Platforms

PARAMETER	JETSON NANO	STM32H7	PSOC EDGE E84
ARCHITECTURE	CORTEX-A57 + GPU	CORTEX-M7	CORTEX-M55 + 2×M33
VECTOR EXTENSION	NEON + CUDA	NONE (FPU ONLY)	HELIUM MVE 128-BIT
CLOCK, MHZ	1430	480	400
TDP, W	5–10	0.5–1	0.1–0.5
REAL-TIME	LIMITED (OS)	FULL	FULL (TCM, 0-WAIT)
PRICE, USD	150–200	20–30	60–80

B. *Baseline Pipeline and Its Bottlenecks*

The reference pipeline has three sequential stages. Monochrome conversion maps RGB to luminance with the weighted sum Y = 0.30R + 0.59G + 0.11B. Gaussian blur convolves the image with a Gaussian kernel to suppress the high-frequency sensor noise that a gradient operator would otherwise mistake for edges. The Sobel operator [5] approximates the intensity gradient with two 3×3 kernels Gx, Gy, and the gradient magnitude is thresholded to a binary edge map. A plain-C baseline was written for readability and used both to generate reference outputs and as the performance reference. Profiling exposed four bottlenecks: floating-point arithmetic (scalar FPU MAC throughput is one operation per cycle, against four float or eight uint16 per cycle in SIMD); element-by-element scalar processing; per-iteration boundary-check branches that break pipelining; and the square root in the magnitude computation, which stalls the pipeline for tens of cycles. At -O2 the baseline costs 508 cycles/pixel, of which Gaussian blur and Sobel together account for more than 95%.

III. OPTIMIZATION METHODS

Six complementary methods were applied to the pipeline.

1) Fixed-point integer arithmetic. Floating-point operations are replaced by integer arithmetic with fixed-point scaling. The monochrome coefficients (0.30, 0.59, 0.11) are scaled by 256 to the integers 77, 151, 28, and the result is shifted right by 8 bits; the Gaussian kernel [0.25, 0.50, 0.25] (actual values vary - these are provided to illustrate the approach used) becomes [64, 128, 64] with the same shift.

2) Separable kernels. Each rank-1 2-D convolution is decomposed into two 1-D passes through an intermediate buffer (Fig. 1). For the Gaussian kernel this reduces multiplications per pixel from 9 to 6 at K = 3 (33%) and from 25 to 10 at K = 5 (60%); the Sobel kernels factor as $Gx = [1,2,1]^T \cdot [-1,0,1]$. The gain is paid for in space – an extra H×W buffer – a classic time–space trade-off that must be weighed against the limited TCM budget.

3) Center/edge loop versioning. Boundary checks are hoisted out of the hot loop by splitting processing into a branch-free central region (up to 95% of pixels) and separate edge cases, removing the per-iteration if-branches that throttle instruction-level parallelism.

4) Manual Helium MVE vectorization. The inner loops are rewritten with arm_mve.h intrinsics [10]: gather loads (vldrbq_gather_offset_u16) deinterleave the RGB888 channels, vmlaq_n_u16 performs the convolution MAC on eight lanes at once, vabsq_s16 and vaddvq_s16 form and reduce the gradient, and vctp16q with predicated loads/stores handles loop tails for arbitrary image widths.

5) Hot data and code in TCM. The reused buffers (gauss_buffer, sobel_buffer) and the kernel coefficients are placed in DTCM and the inner-loop functions in ITCM through linker-section attributes, guaranteeing zero-wait access; the functions are marked NO_INLINE for accurate PMU profiling.

6) Simplified gradient metric with a dynamic threshold. The Euclidean magnitude $\sqrt{(gx^2 + gy^2)}$ is replaced by the L1 approximation |gx|+|gy|, removing two multiplications, an addition and a square root from the hot loop. The fixed threshold is replaced by the per-frame mean magnitude [6], accumulated in parallel with the convolution via vaddvq_s16.

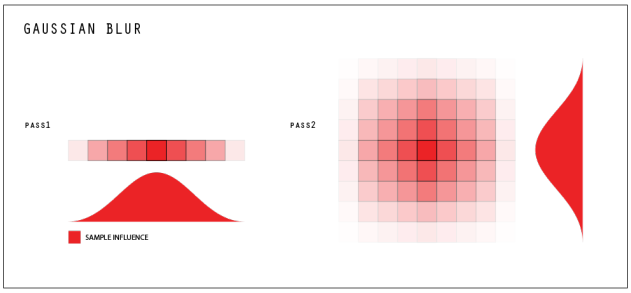

Fig. 1. Decomposition of the 2-D Gaussian kernel into a horizontal (pass 1) and a vertical (pass 2) 1-D pass through an intermediate buffer.

IV. EXPERIMENTAL RESULTS

A. *Measurement Methodology*

Performance was measured with the Cortex-M55 Performance Monitoring Unit (PMU) [8], reading the cycle counter and the retired-instruction and retired-MVE-instruction events, from which cycles/pixel, IPC and MAC/cycle are derived; the full source code is openly available [12]. All results are reported in pairs for the two most practically relevant Arm GNU Toolchain (gcc 14.2) levels, -O2 and -O3. At -O2 auto-vectorization is essentially inactive, so the scalar baseline is close to pure scalar performance and manual vectorization yields the largest relative gain; at -O3 the auto-vectorizer already converts some baseline loops into Helium sequences, raising the baseline 2–3× and lowering the relative speed-up, although the manually vectorized version still attains the lowest absolute cycles/pixel.

TABLE II. SPEED-UP OVER THE SCALAR BASELINE (128×128)

PIPELINE STAGE	SPEED-UP (-O2)	SPEED-UP (-O3)
MONOCHROME CONVERSION	1.57×	1.56×
GAUSSIAN BLUR	18.26×	16.39×
SOBEL OPERATOR	10.53×	8.85×

FULL PIPELINE	10.14×	7.56×

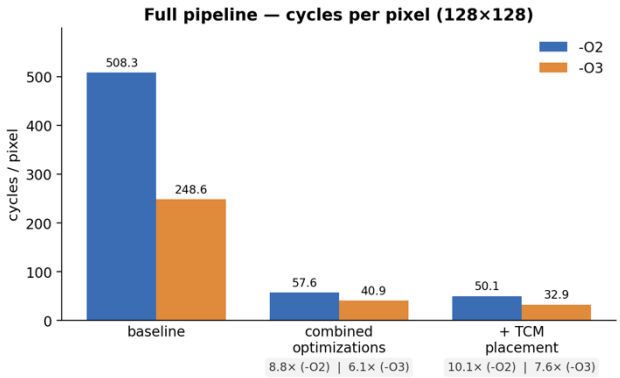

Fig. 2. Full-pipeline cost in cycles per pixel: from the baseline, through the combined algorithmic and vector optimizations, to TCM placement (128×128).

The full pipeline is accelerated 10.14× at -O2 and 7.56× at -O3, dropping from 508.3 to 50.1 cycles/pixel at -O2 (Fig. 2). The largest single contribution is Gaussian blur at 18.26×, which confirms that the algorithmic optimization (separability) outweighs vectorization alone: for that stage the move from a 2-D to two 1-D passes contributed a further 1.72× on top of vectorization, and TCM placement another 1.07×.

For the Sobel stage the gains are spread more evenly across methods: vectorization alone accounts for 5.02×, separability adds a further 1.55×, fixed-point arithmetic 1.20× and the simplified metric 1.07×. In the fully optimized pipeline at -O2 the Sobel operator remains the dominant stage at 52.1% of the runtime, against 24.7% for monochrome conversion and 23.4% for Gaussian blur, because it performs two separable passes (one each for gx and gy) followed by the final binarization.

B. Accuracy

Accuracy was assessed by per-pixel comparison against the float baseline over five test images (Table III). Monochrome conversion and Gaussian blur are practically loss-free – maximum error $\leqslant 2$ and 100% of pixels within ± 4 intensity levels ($\approx 1.6\%$), visually indistinguishable from the reference. The large maximum error of the Sobel stage is a thresholding artifact: a pixel sitting just below the threshold in the reference (mapped to 0) may sit just above it in the optimized version (mapped to 255), giving a nominal difference of 255 even though the edge map is visually identical (Fig. 3). The dynamic threshold gives slightly better agreement (46.3% vs 42.3% exact) because it adapts to the content of each frame.

TABLE III. ACCURACY VS THE FLOAT REFERENCE (MEAN OVER FIVE IMAGES, DYNAMIC THRESHOLD)

STAGE	MAX ERR	MEAN ERR	% EXACT	% ±4
MONOCHROME	0.8	0.05	95.2	100.0
GAUSSIAN BLUR	1.4	0.33	67.1	100.0
SOBEL (DYNAMIC)	77.0	11.83	46.3	51.3

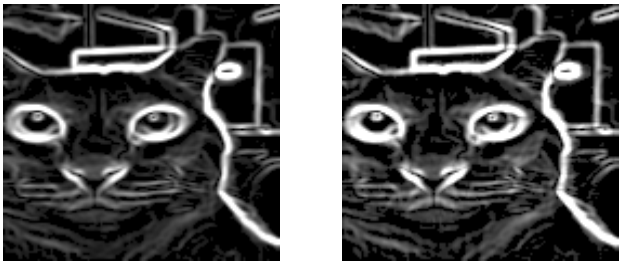

Fig. 3. Sobel edge map of a 128×128 test image: the scalar baseline (left) and the optimized Helium implementation (right) are visually simular, despite the large nominal max-error caused by threshold flips.

V. REAL-TIME CAMERA DEMONSTRATION

The pipeline was integrated with an OmniVision OV7675 CMOS sensor over its parallel DVP interface [11]. The sensor delivers RGB565 frames, so vectorized conversions are inserted at both ends of the pipeline (RGB565 to monochrome on input, monochrome back to RGB565 on output). Two configurations were built. At QVGA (320×240) the reused convolution buffers stay in DTCM while the stage output buffers move to SOCMEM. At VGA (640×480) a single 300 KB frame already exceeds DTCM, so the intermediate buffers are consolidated into one shared SOCMEM buffer and the camera/communication buffers are re-used as ping-pong intermediates; the Gaussian kernel is also enlarged to K = 5, which separability keeps inexpensive (10 multiplications instead of 25). Processed frames are streamed to a host over UART at 1 Mbit/s and shown by a Python/OpenCV visualizer.

Fig. 4 shows live output and the role of the blur stage: without Gaussian filtering the edge map is littered with spurious points from sensor noise, whereas with filtering only genuine object contours remain – confirming the value of the smoothing stage on real frames.

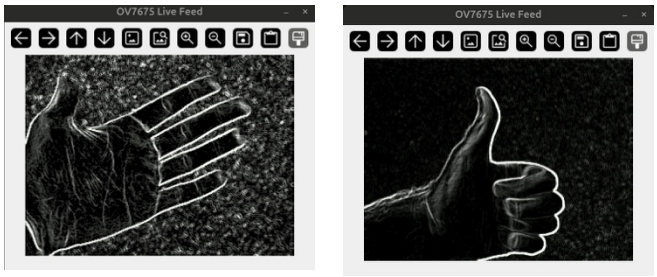

Fig. 4. Live OV7675 edge detection: without Gaussian blur (left) sensor noise produces spurious edges; with blur (right) only true contours remain.

VI. CONCLUSION

A vectorized three-stage edge-detection pipeline – monochrome conversion, Gaussian blur and the Sobel operator – was developed for the ARM Cortex-M55 with Helium MVE on an Infineon PSOC Edge E84. Six complementary optimizations accelerate the full pipeline 10.14× at -O2 and 7.56× at -O3 over the scalar baseline while keeping the result visually equivalent to the float reference, and the pipeline runs in real time on an OV7675 camera at QVGA and VGA resolutions.

The methods have clear limits. Fixed-point arithmetic sacrifices the full dynamic range of float; separability applies only to rank-1 kernels; Helium intrinsics are not portable across NEON, SSE/AVX or RISC-V V; TCM placement is hardware-specific and capacity-bounded (at VGA the 600 KB Sobel buffer must spill to SOCMEM); and the L1 metric biases the gradient magnitude by up to 41%, which is invisible after binarization but would affect algorithms that use the magnitude directly. Future work includes extending the Sobel stage to a full Canny detector (non-maximum suppression and double thresholding), offloading classification to the on-chip Ethos-U55 NPU, streaming over USB CDC to reach 30 fps at VGA, and tile-based processing to reduce the SOCMEM footprint.

REFERENCES

[1] "What edge computing means for infrastructure and operations leaders," Gartner, 2018. [Online]. Available: https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders

[2] "Arm Helium technology – M-profile Vector Extension (MVE) for Armv8.1-M," Arm Ltd., press release, Feb. 2019. [Online]. Available: https://www.arm.com/technologies/helium

[3] "Arm launches the Cortex-M55 and its micro-NPU companion, the Ethos-U55," WikiChip Fuse, Feb. 2020. [Online]. Available: https://fuse.wikichip.org/news/3306/

[4] "Infineon announces next-generation PSOC Edge portfolio (E81/E83/E84)," Infineon Technologies, press release, Apr. 2024. [Online]. Available: https://www.infineon.com/market-news/2024/infcss202404-086

[5] I. Sobel and G. Feldman, "A 3×3 isotropic gradient operator for image processing," Stanford AI Project, 1968.

[6] "Sobel edge detection algorithm with adaptive threshold," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 2, 2023.

[7] Introduction to Helium – Helium Programmer's Guide. Arm Ltd., 2020. [Online]. Available: https://developer.arm.com/documentation/102102/0103

[8] Arm Cortex-M55 Processor Software Optimization Guide. Arm Ltd., 2021.

[9] Selecting and Configuring Memories for Power and Performance in PSOC Edge MCU, Infineon Application Note AN239774, 2024.

[10] Arm Helium Technology Intrinsics Reference. Arm Ltd. [Online]. Available: https://developer.arm.com/architectures/instruction-sets/intrinsics/

[11] OV7675 CMOS VGA Camera Datasheet. OmniVision Technologies, 2016.

[12] K. Filippenkov, "Gaussian blur + Sobel edge detection on ARM Cortex-M55," project repository, GitHub, 2026. [Online]. Available: https://github.com/malnormalulo/gaussian-sobel-edge-detection

[13] PSOC Edge E84 AI Kit User Guide. Infineon Technologies, 2024.

[14] E84 PSOC Edge MCU Datasheet. Infineon Technologies, 2024.