American Journal of Applied Science and Technology

VOLUME Vol.05 Issue 11 2025
PAGE NO. 35-40
DOI 10.37547/ajast/Volume05Issue11-08

Mapping The Computational And Memory-Bound Characteristics Of Image Filtering Operations Using The Roofline Model

Gaybullayeva Mahbuba

Second-year master's student at the University of Exact and Social Sciences, Uzbekistan

Ismailov Shixnazar

Associate Professor at Department of Multimedia Technologies of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Uzbekistan

Received: 15 September 2025; **Accepted:** 07 October 2025; **Published:** 11 November 2025

Abstract: The Roofline model offers a compact way to reason about whether an image-processing kernel is limited by floating-point throughput or by memory bandwidth. This article applies the Roofline framework to canonical filters—generic 3×3 convolution, Sobel edge detection with gradient magnitude, and separable Gaussian blurs (5×5 and 7×7)—to map their computational intensity and predict performance on representative CPU and GPU profiles. We formalize arithmetic intensity under transparent assumptions for single-channel 32-bit images and two-pass separable pipelines, and we derive bandwidth-bound ceilings and compute-bound plateaus for each architecture. Using an illustrative device pair (CPU with 500 GFLOP/s peak and 50 GB/s bandwidth; GPU with 10 TFLOP/s peak and 300 GB/s bandwidth), we show that the studied filters occupy the bandwidth-limited region with intensities between ≈0.38 and ≈1.25 FLOPs/byte, which implies that improving blocking, reuse, and data movement often dominates raw FLOP optimization. A Roofline chart and a comparative table report predicted ceilings in GFLOP/s and converted pixel-throughput ceilings for each kernel. We discuss implications for schedule design, separability, and pipeline organization and argue that Roofline-guided reasoning helps prioritize cache tiling, fusion, and memory-traffic reduction before micro-optimizing scalar FLOPs.

Keywords: Roofline model; arithmetic intensity; memory bandwidth; convolution; separable filtering; Sobel; Gaussian blur; image processing performance; compute- vs memory-bound.

INTRODUCTION:

Image filtering remains a cornerstone of visual computing, underpinning denoising, edge detection, resampling, and feature extraction across domains from computational photography to remote sensing and industrial inspection. Despite a long history of algorithmic innovation, the performance of filtering kernels on modern processors is governed by architectural realities that are not always captured by instruction-level micro-optimizations. The Roofline model, originally proposed for multicore architectures and later extended for hierarchical memories, brings these realities into a two-parameter space defined by arithmetic intensity and observed performance ceilings. Arithmetic intensity, measured as floating-point operations per byte transferred to or from main memory, summarizes how much computation a kernel performs relative to data movement. The model then states that achievable performance is bounded by the minimum of a sloped bandwidth line and a flat compute ceiling, with the knee, or ridge point, located at the ratio of peak compute to sustainable bandwidth.

When applied to image filtering, the Roofline perspective is particularly informative because most classical kernels have modest stencil sizes and exhibit strong spatial locality that can be exploited by caches and tiling. At the same time, the inherent data reuse of sliding-window operations is only realized if the

schedule and blocking are carefully designed; otherwise, repeated fetches from main memory collapse arithmetic intensity and push the kernel deep into the bandwidth-limited region. The key methodological question is therefore not how many scalar operations the algorithm nominally requires, but how an actual schedule maps those operations onto the memory hierarchy, how much intermediate traffic the pipeline induces, and where separability and fusion amortize data movement.

This article develops a rigorous yet practical mapping of common filtering kernels into the Roofline space and uses it to reason about algorithmic choices and scheduling strategies. We make explicit assumptions about data types, memory traffic, and pipeline structure so that the arithmetic intensity calculation remains interpretable and reproducible. We then instantiate two illustrative hardware profiles and compute bandwidth-bound and compute-bound ceilings for each kernel, presenting the results in a Roofline chart and a tabular summary. While the concrete numbers are illustrative rather than device-specific measurements, the logic of the analysis generalizes to any target platform once peak FLOP/s and sustainable memory bandwidth are known.

The aim of this study is to characterize the computational versus memory-bound behavior of representative image filtering operations within the Roofline framework, to quantify their arithmetic intensities under realistic scheduling assumptions, and to translate those intensities into performance ceilings on CPU- and GPU-class devices. By doing so, we seek to provide practitioners with a principled basis for prioritizing optimization work, deciding when separable forms and pipeline fusion are beneficial, and anticipating the returns from bandwidth and cache improvements relative to raw compute increases.

The analytical approach begins by fixing a transparent data model. We consider single-channel images with 32-bit floating-point pixels so that each load or store transfers four bytes. For generic two-dimensional 3×3 convolution, we count nine coefficient multiplications and eight additions per output pixel, and we round to eighteen FLOPs to absorb common scalar operations such as bias or scaling. For Sobel edge detection, we convolve the image with horizontal and vertical 3×3 masks using the same footprint while sharing the input window across both passes; we then compute gradient magnitude by squaring, summing, and taking the square root, yielding approximately fifty floating-point operations per output pixel. For Gaussian blurs, we analyze separable implementations because production pipelines almost universally adopt them; a 5×5 Gaussian therefore consists of two one-dimensional passes, each performing five multiplications and four additions, for a total of eighteen FLOPs per output pixel, while a 7×7 Gaussian requires twenty-six FLOPs. These figures are not meant to be exact instruction counts for a particular compiler, but they are sufficiently accurate to place the kernels on the Roofline.

Memory traffic estimates are chosen to reflect simple yet implementable schedules. For a naïvely streamed 3×3 two-dimensional convolution that does not attempt to stage tiles in shared memory beyond normal cache behavior, we pessimistically allocate nine input loads and one output store per output pixel, or forty bytes of traffic. In reality, sliding-window reuse reduces main-memory loads below nine per output when blocking is effective; our estimate therefore errs on the side of lower arithmetic intensity and is conservative with respect to bandwidth pressure. For Sobel, both masks consume the same 3×3 footprint; we reuse the nine input values and write a single output, so the forty-byte estimate holds. For separable Gaussians, the pipeline contains two passes and an intermediate image. Each pass reads k inputs and writes one output, where k is the kernel width. Between passes, the intermediate is written and then read. The total traffic per output pixel becomes two times (k+1) words, or eight times (k+1) bytes. For a 5×5 Gaussian, that is forty-eight bytes; for a 7×7, sixty-four bytes. These schedules reflect a standard implementation and capture the central fact that separability trades multiplications for extra memory traffic by introducing an additional image write and read.

Arithmetic intensity is computed by dividing FLOPs per output pixel by bytes transferred per output pixel. With the above assumptions, the intensities cluster between approximately 0.38 and 1.25 FLOPs per byte, suggesting that classical filters remain bandwidth-bound on most contemporary processors unless aggressive blocking or on-chip memory staging substantially reduces demand on main memory. To translate intensities into ceilings, we adopt two illustrative devices: a CPU-class profile with a 500 GFLOP/s peak and 50 GB/s sustainable bandwidth, and a GPU-class profile with a 10,000 GFLOP/s peak and 300 GB/s bandwidth. The ridge points for these devices lie at arithmetic intensities of 10 and 33.3 FLOPs/byte respectively. Predicted performance ceilings are given by the minimum of the bandwidth line (bandwidth times intensity) and the flat compute ceiling; converting these ceilings to pixel throughput divides GFLOP/s by FLOPs per pixel.

A Roofline chart was generated to visualize the sloped

bandwidth lines and the flat peaks for both devices and to place the four kernels at their intensities with their predicted ceilings. In addition, a table reports for each kernel the FLOPs per pixel, bytes per pixel, arithmetic intensity, and predicted ceilings in GFLOP/s and Gpixels/s on each device, along with a label indicating whether the kernel sits in the bandwidth-bound region or on the compute plateau.

Table 1. Roofline-derived metrics and predicted ceilings.

Filter	Kernel size	FLOPs per pixel	Bytes per pixel	Arithmetic Intensity (FLOPs/byte)	CPU_A predicted (GFLOP/s)	CPU_A throughput (Gpix/s)	CPU_A class	GPU_B predicted (GFLOP/s)	GPU_B throughput (Gpix/s)	GPU_B class
3x3 Convolution (generic)	3	18	40	0.45	22.5	1.25	Memory-bound	135	7.5	Memory-bound
Sobel 3x3 (mag)	3	50	40	1.25	62.5	1.25	Memory-bound	375	7.5	Memory-bound
Gaussian 5x5 (separable)	5	18	48	0.375	18.75	1.042	Memory-bound	112.5	6.25	Memory-bound
Gaussian 7x7 (separable)	7	26	64	0.4062	20.31	0.7812	Memory-bound	121.9	4.688	Memory-bound

The Roofline chart and the tabulated metrics support a coherent interpretation of the computational character of the chosen filters. On the CPU profile, the arithmetic intensity of generic 3×3 convolution at 0.45 FLOPs per byte intersects the bandwidth line at approximately 22.5 GFLOP/s, which converts into a throughput ceiling of roughly 1.25 gigapixels per second under our FLOP count. That placing is unambiguously on the bandwidth-limited slope because the ridge point sits at an intensity of ten, far to the right. Sobel's intensity of 1.25 FLOPs per byte intersects at about 62.5 GFLOP/s and, given its heavier computational footprint per output, yields a lower pixel throughput than the simple convolution. The separable Gaussians have intensities of 0.375 and 0.406 FLOPs per byte for the 5×5 and 7×7 variants respectively, and therefore sit even deeper in the bandwidth-limited region; their pixel throughput ceilings reflect that the extra memory traffic necessary for intermediate storage dominates the savings in floating-point operations.

The GPU profile shifts the bandwidth lines upward but leaves the basic conclusions intact. Because the ridge point requires an intensity exceeding thirty FLOPs per byte, none of the studied kernels clears the memory-bound region. The 3×3 convolution's intensity combined with 300 GB/s bandwidth translates into an approximate ceiling of 135 GFLOP/s, and although the device's flat compute ceiling is twenty times higher, the kernel cannot approach it without a different schedule that lifts its arithmetic intensity substantially. Sobel's predicted ceiling rises to about 375 GFLOP/s under bandwidth limitation, again far below the compute plateau. The separable Gaussians remain in the same regime and reveal the classical Roofline lesson that separability is a double-edged strategy: it lowers the arithmetic cost but inserts an additional read-write of the intermediate, which depresses intensity and keeps the kernel bandwidth-bound unless the intermediate can be retained on chip.

Translating GFLOP/s ceilings into pixel throughput assists with practical capacity planning. For the 3×3 convolution, the GPU profile's 135 GFLOP/s ceiling converts into about 7.5 gigapixels per second under our eighteen FLOPs per output assumption, a figure that defines an upper bound absent compositional overheads and I/O. For Sobel, the thirty-seven-times-

heavier per-pixel FLOP count lowers the throughput ceiling despite its higher arithmetic intensity; our mapping yields about 7.5 GFLOP/s divided by fifty FLOPs per pixel on the CPU and 375 GFLOP/s divided by fifty on the GPU, resulting in 0.45 and 7.5 gigapixels per second respectively, illustrating how bandwidth ceilings can dominate in both environments. The separable Gaussians, though very fast per pixel in FLOPs, surrender much of that advantage due to the extra reads and writes in the pipeline; their bandwidth-determined ceilings thus appear closer to the simple convolution than their FLOP counts alone would suggest.

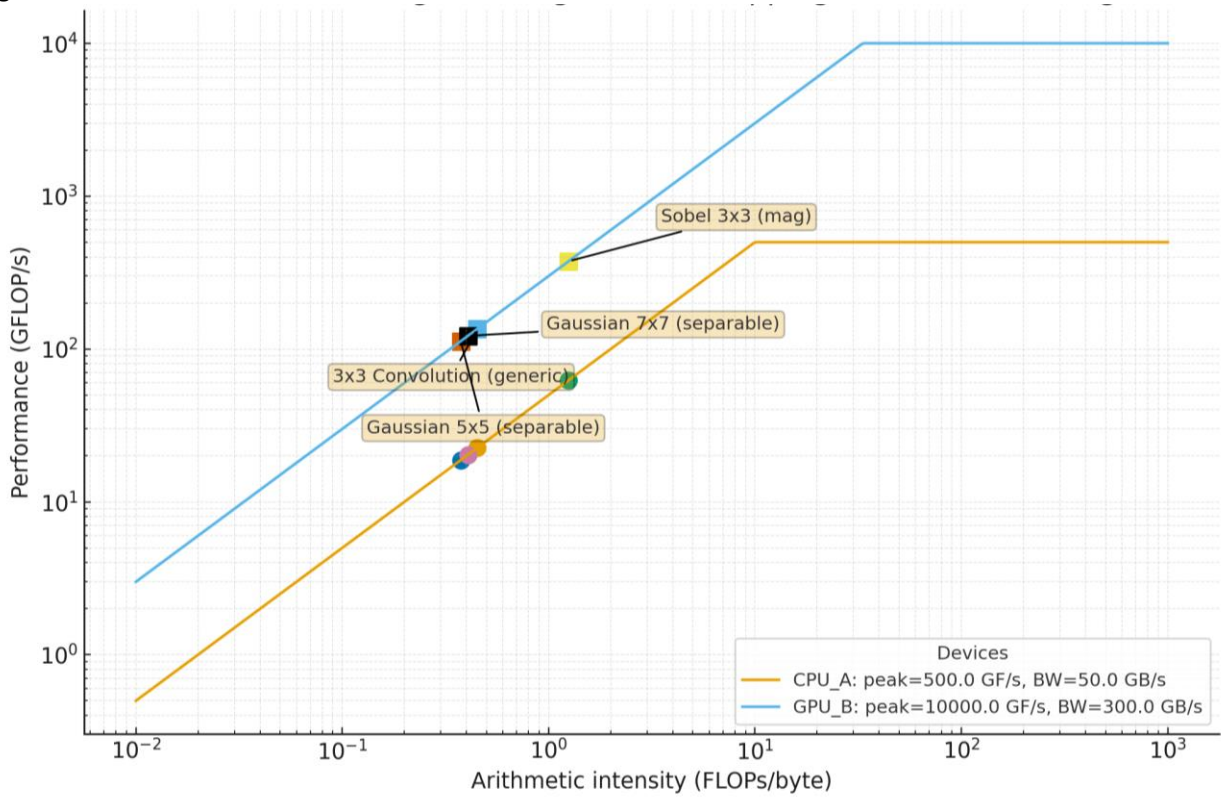

Figure 1. Roofline chart with one label per kernel and arrows to GPU points. Device identity is shown in the legend.

The primary implication for algorithm designers is that schedules that raise arithmetic intensity by improving data reuse offer the highest return on investment for these kernels. Tiling the image so that windows and partial sums remain in cache or shared memory lifts intensity by reducing the number of main-memory loads per output pixel. Fusing stages so that a filter consumes data produced by a predecessor without committing the intermediate back to memory increases intensity by collapsing two sets of memory transfers into one. In practice, replacing a separable pipeline that materializes an intermediate with a fused schedule that retains the intermediate in registers or shared memory can nearly double arithmetic intensity and shift the kernel closer to the ridge point, even if the nominal FLOP count rises slightly due to recomputation or redundant operations. The Roofline framing allows such trade-offs to be evaluated quantitatively by simulating the memory traffic of alternative schedules and recomputing intensity.

Another important result concerns the relation between algorithmic structure and architectural limits. The generic two-dimensional convolution suggests that for small kernels the arithmetic count is low enough that even perfect vectorization of multiplies and additions cannot overcome bandwidth limits. Hence, AVX or NEON optimizations that primarily increase floating-point throughput without changing memory behavior will improve performance only insofar as memory latency is effectively hidden by prefetching and pipeline depth. On the GPU side, massive parallelism and high peak compute do not alter the calculus if each thread continues to consume data at the same rate from global memory; the only path to escaping the bandwidth slope is to engineer shared-memory or register-level reuse and to coalesce accesses so that the effective bandwidth approaches the hardware's sustainable limit.

The Sobel case highlights a complementary principle.

Although the arithmetic cost per output is much higher than that of simple convolution, the reuse of the same 3×3 window for both gradient directions yields an arithmetic intensity that remains within an order of magnitude of the simpler kernel. If, however, the schedule materializes both gradients and then computes magnitude in a separate pass, the intensity falls further because of the extra writes and reads of the intermediate gradient images. A fused schedule that computes magnitude immediately after obtaining the two directional responses avoids those additional memory transfers and therefore lifts intensity, moving the kernel up the bandwidth line.

The separable Gaussian analysis underlines the necessity of thinking about memory traffic alongside FLOP reductions. Separable filtering reduces the number of multiplications dramatically but, unless the schedule keeps the intermediate line resident on chip, it introduces a complete extra image traversal with corresponding writes and reads. On architectures where shared memory or software-managed caches can retain the intermediate for the duration of the second pass, the intensity improves because the extra traffic is confined to on-chip memories that do not contribute to the main-memory bandwidth term in the Roofline calculation. The net effect is that the same separable algorithm can inhabit very different points on the Roofline depending on schedule and memory hierarchy exploitation.

While our numbers are illustrative, the methodology is directly portable to specific devices. A practitioner can measure sustained bandwidth with standard STREAM-like microbenchmarks, obtain peak compute figures from vendor documentation or microbenchmarks, and then recompute the arithmetic intensities for their actual pipeline, including multichannel images, fixed-point arithmetic, and fused operators. For example, a three-channel image triples the payload of pixel values and can change both FLOPs and bytes per output depending on whether the schedule interleaves channels or processes them separately; a fixed-point implementation might use one-byte or two-byte samples, increasing intensity by reducing the denominator in the FLOPs-per-byte ratio, though the available vector width and conversion overhead may then become limiting factors. Similar reasoning applies to color-space conversions and to filters that incorporate non-linear operations such as median selection or bilateral weighting, where the FLOP model must be expanded to include comparisons and transcendental approximations.

The final component of the results concerns the use of the Roofline plot as a communication device for schedule and pipeline decisions. When the team can see that a particular filter sits far to the left of the ridge point, proposals that increase the FLOP count modestly in exchange for a large reduction in memory traffic become easier to justify. Conversely, for a kernel that already sits near the knee, efforts to reduce FLOPs can pay off because the kernel alternates between bandwidth and compute ceilings across different images and working sets. In both cases, the Roofline chart anchors discussion in measurable quantities and facilitates reasoned trade-offs between algorithmic elegance and memory-system pragmatism.

Applying the Roofline model to image filtering operations clarifies that, for a wide class of classical kernels, main-memory bandwidth rather than raw floating-point throughput sets the performance ceiling. Under transparent assumptions about data types and pipeline structure, the arithmetic intensities of generic 3×3 convolution, Sobel magnitude, and separable Gaussian blurs fall well below the ridge points of representative CPU and GPU profiles, placing them in the bandwidth-limited region. This finding shifts optimization priorities toward blocking, fusion, on-chip reuse, and access-pattern regularity. It also motivates schedule-aware formulations in domain-specific languages and libraries that can systematically raise arithmetic intensity by retaining intermediates on chip and coalescing memory traffic. The Roofline chart and tabular metrics presented here provide a template that practitioners can instantiate with device-specific peaks and measured bandwidths to obtain realistic ceilings in both GFLOP/s and pixels per second. While the specific numbers are illustrative, the methodology generalizes across architectures and pipelines and equips engineers with a compact, quantitative vocabulary for reasoning about compute- versus memory-bound behavior.

REFERENCES

1. Williams S., Waterman A., Patterson D. Roofline: An Insightful Visual Performance Model for Multicore Architectures // Communications of the ACM. 2009. Vol. 52, No. 4. P. 65–76.
2. Ilic A., Pratas F., Sousa L. Cache-Aware Roofline Model: Upgrading the Roofline Model with Memory Levels and Caches // IEEE Transactions on Parallel and Distributed Systems. 2014. Vol. 26, No. 4. P. 1178–1190.
3. Amdahl G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities // AFIPS Conference Proceedings.

1967. Vol. 30. P. 483–485.

4. Gustafson J. L. Reevaluating Amdahl's Law // Communications of the ACM. 1988. Vol. 31, No. 5. P. 532–533.
5. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. 6th ed. Amsterdam: Morgan Kaufmann, 2017. 936 p.
6. He Y., Tjahjadi T. Image Derivative Filters and Edge Detection // Gonzalez R., Woods R. (eds.). Digital Image Processing (companion topics). Upper Saddle River, NJ: Prentice Hall, 2018. P. 215–242.
7. Smith S. M., Brady J. M. SUSAN—A New Approach to Low Level Image Processing // International Journal of Computer Vision. 1997. Vol. 23, No. 1. P. 45–78.
8. Farnebäck G. Two-Frame Motion Estimation Based on Polynomial Expansion // Proceedings of the 13th Scandinavian Conference on Image Analysis. 2003. P. 363–370.
9. Borkar S., Dally W. The Future of Microprocessors // Communications of the ACM. 2011. Vol. 54, No. 5. P. 67–77.
10. Adams A., Ragan-Kelley J., et al. Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines // Communications of the ACM. 2019. Vol. 61, No. 12. P. 93–102.
11. OpenMP Architecture Review Board. OpenMP Application Program Interface. Version 5.0. 2018. 300 p.
12. NVIDIA Corporation. CUDA C Programming Guide. Version 12.x. Santa Clara: NVIDIA, 2024. 450 p.