MASSIVELY PARALLEL IMPLEMENTATION OF GAUSSIAN SMOOTHING ON MANY-CORE ARCHITECTURES

Muzaffar Xolmirzo o‘g‘li Umarov
a lecturer in the Department of Exact Sciences at
Tashkent International University of Kimyo Technology.
E-mail: ***muzaffarbekumarov98@gmail.com***

Abstract: *As image processing techniques demand higher performance, traditional single-threaded convolution methods become increasingly inadequate. This study focuses on the parallel implementation of 2D Gaussian convolution to overcome these computational hurdles. By offloading the workload to parallel systems, we significantly reduce processing latency. The paper evaluates the efficiency of Gaussian filtering on heterogeneous platforms, specifically contrasting multi-core CPU architectures against GPU acceleration. Experimental results, conducted on the Google Colab platform, demonstrate the superior speedup achieved through hardware-level parallelization.*

Keywords: *Parallel Computing, Image Convolution, Gaussian Blur, Google Colaboratory.*

INTRODUCTION. The proliferation of high-performance computing is driven by the necessity to address increasingly complex computational problems and the exponential growth of large-scale datasets (Bozkurt et al., 2015). In the domain of image processing, there has been a paradigm shift from sequential execution to parallel programming frameworks. This transition is motivated by the fact that core operations, such as image filtering and convolution, are inherently resource-intensive, with their complexity scaling proportionally to image resolution (Reddy et al., 2017).

Convolution, a fundamental yet computationally demanding mathematical operator, serves as the backbone of modern image processing (Novák et al., 2012). Contemporary parallelization strategies typically leverage the distinct architectures of multi-core Central Processing Units (CPUs) and Graphics Processing Units (GPUs). While CPU speedup is intrinsically constrained by the number of physical cores, GPUs—powered by many-core architectures and platforms like CUDA—have emerged as the superior alternative. These systems can achieve acceleration factors ranging from hundreds to thousands of times greater than sequential implementations (Reddy et al., 2017).

This paper evaluates the performance of 2D Gaussian convolution across parallel architectures within the Google Colaboratory environment. We provide a comparative analysis of execution efficiency between multi-core CPUs and GPUs, specifically assessing the impact of cloud-based infrastructure on throughput. The remainder of this study is organized as follows: Section 1 contextualizes parallel systems and Gaussian filtering; Section 2 details

the experimental methodology, hardware architecture, and results; finally, we conclude with a summary of our findings and their implications."

CPU and GPU. A multi-core CPU serves as a unified computing unit integrating multiple independent execution cores. To leverage this hardware parallelism, application programming interfaces (APIs) such as OpenMP and Threading Building Blocks (TBB) have become industry standards for optimizing multi-core efficiency (Polesel et al., 2000). In this research, we employ the Pymp library, which facilitates OpenMP-like shared-memory parallelization within the Python environment, allowing for the concurrent distribution of algorithmic tasks across CPU cores.

Conversely, the Graphics Processing Unit (GPU) is engineered based on a Single Instruction, Multiple Data (SIMD) stream architecture. This design is exceptionally well-suited for applications where a uniform instruction set is executed simultaneously across vast data arrays. In the context of image convolution, pixels are treated as autonomous data elements, rendering the GPU's massively parallel architecture ideal for such operations (Polesel et al., 2000). To maximize GPU utilization, this study utilizes the CUDA (Compute Unified Device Architecture) platform, currently the most prevalent framework for high-performance GPGPU computing.

The fundamental architectural divergence between CPU and GPU lies in the allocation of processing resources (Reddy et al., 2017). While a CPU comprises fewer, more powerful cores optimized for complex control logic and extensive caching, a GPU is composed of hundreds or thousands of smaller, specialized cores designed for high-throughput data processing. This structural difference enables GPUs to achieve a significantly higher degree of parallelism compared to traditional multi-core CPUs.

NVIDIA, CUDA Architecture and Threads. The GPU architecture is fundamentally based on the SIMD (Single Instruction, Multiple Data) programming model, comprising hundreds of individual processing units known as Scalar Processors (SPs). These SPs are organized into clusters called Streaming Multiprocessors (SMs), with each SM typically integrating eight SPs (Lad et al., 2012). Within an SM, all constituent SPs execute a uniform instruction concurrently, a paradigm specifically referred to as Single Instruction, Multiple Thread (SIMT).

NVIDIA’s Compute Unified Device Architecture (CUDA) serves as a transformative parallel computing platform that leverages this hardware to achieve significant performance gains. CUDA provides an extension to the C language, enabling automated parallelization and alleviating the need for complex structural modifications to traditional sequential code. Beyond its threading capabilities, CUDA introduces memory scatter operations, providing enhanced flexibility for complex data manipulation (Reddy et al., 2017). This architecture has been widely adopted across diverse scientific domains, including bio-chemistry, fluid dynamics, and medical imaging, such as Computerized Tomography (CT) (Bozkurt et al., 2015).

From a software perspective, the CUDA API manages execution through a hierarchical thread structure: individual threads are grouped into blocks, which collectively constitute a grid. These blocks are dynamically assigned to available SMs for serial execution, ensuring efficient resource utilization across the GPU's hardware layers (Lad et al., 2012).

Related Work. The optimization of image convolution remains a pivotal research area in computational image processing. Prior studies have extensively investigated the performance gains achieved by transitioning from sequential to parallel execution. For instance, Novák et al. (2012) explored the efficiency of Gaussian blur filters on multi-core CPU architectures, demonstrating significant improvements over single-threaded implementations. Similarly, Chauhan (2018) examined the acceleration of Gaussian filtering through the CUDA platform, highlighting the potential of GPU-based processing.

While earlier research primarily focused on maximizing performance within a single architecture—either by optimizing multi-core CPU usage or comparing GPU execution against sequential code—recent literature has shifted toward cross-platform comparative analysis. Samet et al. (2015) evaluated the speedup of real-time applications using C++ and OpenMP across CPU and GPU environments. Furthering this comparison, Reddy et al. (2017) analyzed the performance of edge detection algorithms using C and CUDA on the NVIDIA GeForce 970 GTX.

Building upon these foundations, our study provides a comparative performance evaluation between two distinct parallel systems: a multi-core CPU and a high-performance GPU. Unlike previous works that predominantly utilized C/C++, we leverage Python's parallel ecosystem—specifically the Pymp library for CPU multi-threading and CUDA for GPU acceleration. The experimental benchmarks are conducted on enterprise-grade hardware, utilizing an Intel Xeon CPU and an NVIDIA Tesla P100 GPU within a cloud-based infrastructure.

EXPERIMENTAL SETUP. In our experiment, we are using Google Colaboratory or "colab" to run our code. Google Colaboratory is a free online cloud-based Jupyter notebook environment that allows us to train PyCUDA and which gives you easy, pythonic access NVIDIA's CUDA parallel computation API. To be able to run our code on Google Colabs, we used the Python programming language. Python pymp library is used for the multiprocessor code. This package brings OpenMP-like functionality to Python. It takes the good qualities of OpenMP such as minimal code changes and high efficiency and combines them with the Python Zen of code clarity and ease-of-use. Python PyCUDA library is used to be able to call CUDA function in our python code and PyCUDA gives you easy, pythonic access to NVIDIA's CUDA parallel computation API.

Experiment Results. The experimental evaluation was conducted using three benchmark images of varying resolutions: 256x256 (Small), 1920x1200 (HD), 3840x2160 (4K). To ensure statistical significance, the Gaussian filter was executed across multiple kernel sizes, and the average processing time was recorded over 10 independent runs. The image processing pipeline followed a structured sequence:

1. Channel Decomposition: Extraction of the primary Red, Green, and Blue (RGB) color channels.

2. Parallel Convolution: Independent convolution of each channel with the Gaussian kernel.

3. Channel Integration: Recomposition of the processed channels to generate the final blurred output.

For the GPU implementation, we utilized the PyCUDA library, enabling the execution of optimized CUDA C kernels within the Python environment. The workflow involved a rigorous data orchestration process, including host-to-device memory transfer, kernel execution on the GPU, and the subsequent retrieval of processed data (device-to-host).

The performance was benchmarked by comparing the execution latency of the dual-threaded CPU implementation against the many-core GPU acceleration, as detailed in Table 1 and Table 2. The resulting processing times and speedup factors are visualized in Figures 2 and 3, respectively. Qualitative results for the 256x256 resolution with a 7x7 kernel are illustrated in Figures 4 (Original) and 5 (Processed).

A comparative analysis reveals that our GPU-based speedup significantly exceeds previous benchmarks. Specifically, our implementation achieved an acceleration factor more than three times higher than the results reported by Bozkurt et al. (2015), validating the efficacy of our optimization approach on contemporary hardware.

Table 1: CPU time.

Image resolution pixels	**Kernel size**	**Processing time(s)**
256 x 256	7 x 7	8.2
	13 x 13	34.9
	15 x 15	47.8
	17 x 17	62.3
1920 x 1200	7 x7	429.3
	13 x 13	1358.2
	15 x 15	1882.9
	17 x 17	2294.15
3840 × 2160	7 x 7	1296.4
	13 x 13	5322.2
	15 x 15	7347.02
	17 x 17	9225.1

Table 2: GPU time.

Image resolution (pixels)	**Kernel size**	**Processing time (s)**
256 x 256	7 x 7	0.00200

	13 x 13	0.00298
	15 x 15	0.00312
1920 x 1200	17 x 17	0.00363
	7 x 7	0.02726
	13 x 13	0.03582
	15 x 15	0.04410
3840 × 2160	17 x 17	0.054377
	7 x 7	0.06950
	13 x 13	0.11282
	15 x 15	0.14214
	17 x 17	0.17886

CONCLUSIONS. This study demonstrates that the Google Colaboratory platform provides a robust and accessible infrastructure for high-performance computing, particularly leveraging its cloud-based GPU resources. Our experimental results confirm the architectural superiority of the CUDA-enabled GPU over multi-core CPU configurations for data-intensive tasks. Quantitative analysis, as illustrated in Figures 2 and 3, reveals that the GPU execution latency is negligible compared to the CPU's processing time. While increasing CPU thread counts may offer marginal performance gains, it remains fundamentally incapable of bridging the performance gap inherent in many-core versus few-core architectures.

The findings reinforce the conclusion that the GPU's SIMT (Single Instruction, Multiple Thread) model is the optimal candidate for spatial domain operations like 2D convolution, where uniform, complex mathematical functions are applied across massive pixel arrays. Consequently, Gaussian blur filtering for noise reduction achieves significantly higher throughput and efficiency when offloaded to parallel GPU systems.

Future research will focus on benchmarking this algorithm across a broader spectrum of heterogeneous multi-core CPU and GPU architectures. Furthermore, we intend to conduct an in-depth investigation into memory orchestration, specifically analyzing and optimizing the host-to-device data transfer latency. Reducing this overhead is critical for maximizing end-to-end performance and achieving even higher speedup factors in real-time image processing applications.

REFERENCES:

1) David B. Kirk, Wen-mei W. Hwu – Programming Massively Parallel Processors: A Hands-on Approach, Morgan Kaufmann, San Francisco, 2016, 1–512 b.

2) Jason Sanders, Edward Kandrot – CUDA by Example: An Introduction to General-Purpose GPU Programming, Addison-Wesley, Boston, 2010, 1–300 b.

3) Richard Szeliski – Computer Vision: Algorithms and Applications, Springer, London, 2022, 1–979 b.

4) Rafael C. Gonzalez, Richard E. Woods – Digital Image Processing, Pearson, New York, 2018, 1–1024 b.

5) NVIDIA – Parallel Programming in CUDA C++, NVIDIA, Santa Clara, 2023, 1–800 b.

6) Umarov, Muzaffar, et al. "Yashil Iqtisodiyot Sharoitida Kichik Biznes Va Xususiy Tadbirkorlikni Rivojlantirishda Davlatning Roli." *Green Economy and Development* 3.4 (2025): 665223.

7) Muradovich, Rustamov Rakhmatali, et al. "Methodology of development of technical and economic model of agricultural machinery dealer point." (2021): 2649-2655.

8) Toshboltaev, M. T., and Z. A. Seytimbetova. "Multichannel System State Graph of Combine Harvesters Branded Technical Service in Uzbekistan." *Agricultural Machinery and Technologies* 14.1 (2020): 46-49.

9) Жураев, Жамшед Рахматиллоевич, Сайдулла Собиржонович Исоков, and Шавкат Куранбаевич Яхшибоев. "Фалсафа доктори (PhD) илмий даражасини олиш учун ихтисослик буйича малакавий имтихон топширувчи талабгорлар РУЙХАТИ." *Социология* 22: 01.

10) Sherzodbek, N. (2025). Til o'rganishda motivatsiyaning roli: ichki va tashqi omillar. *yangi o ‘zbekiston, yangi tadqiqotlar jurnali*, *2*(5), 470-474.

11) Nargiza, O., & Diyora, A. (2025). Building a theoretically informed uzbek-english speech recognition system: practical implementation and evaluation. *Great britain-scientific review of the problems and prospects of modern science and education*, *1*(8), 36-41.

12) Tozhibayev, Bahromjon, and Saidulla Isokov. "Youth Social Deviation in the Process of Reforms in Uzbekistan." *IX International Scientific and Practical Conference “Current Problems of Social and Labour Relations"(ISPC-CPSLR 2021)*. Atlantis Press, 2022.

13) Usmanovich, Ibragimov Isroil, and Milimo Mundia. "Issues of improving the organizational and economic mechanism of business management." (2021): 10694-10702.

14) Габдулхаков, Фарит Авхунович, and Расима Фаридовна Габдулхакова. "Текст как средство подготовки к межкультурному диалогу при обучении русскому языку." *Текст культуры и культура текста*. 2017.

15) Umarov, Muzaffar, et al. "Yashil Iqtisodiyot Sharoitida Kichik Biznes Va Xususiy Tadbirkorlikni Rivojlantirishda Davlatning Roli." *Green Economy and Development* 3.4 (2025): 665223.