MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

PARALLEL PROGRAMMING (CSE -3261)
MINI PROJECT REPORT

Optimizing Edge Detection: A Comparative Study of Sobel Algorithm Performance Across CPU, GPU, and Hybrid Computing Environments

SUBMITTED TO

Department of Computer Science & Engineering

by

Anurag Nayak, 210905016, anurag.nayak@learner.manipal.edu
Kushala Sarada, 210905189, kushala.v@learner.manipal.edu
Adruti Onam, 210905190, adruti.onam@learner.manipal.edu

6th A

Name & Signature of Evaluator

(Jan 2024 – May 2024)

Table of Contents

Chapter 1. Introduction

1.1 Introduction to the Problem Statement

Edge detection serves as a critical step in interpreting visual data, allowing computers to perceive the world in a manner similar to human vision. The Sobel edge detection algorithm, by approximating gradients in image intensity, provides a foundational approach for identifying these transitions. However, the algorithm's computational intensity, especially for high-resolution images or real-time processing needs, challenges conventional CPU-based methods. The advent of General Purpose GPU (GPGPU) computing, particularly through CUDA (Compute Unified Device Architecture), offers a transformative potential for accelerating such image processing tasks. This research is aimed at a systematic comparison of the Sobel edge detection algorithm's performance across various computational platforms: the traditional CPU, the GPU utilizing CUDA, and a hybrid approach incorporating CUDA and MPI. This comparison seeks to underscore the efficacy of parallel computing in significantly enhancing image processing workflows.

For **small-scale image processing tasks**, CUDA alone is highly efficient. This efficiency comes from the ability to leverage GPU parallelism directly, without the overhead of managing communication between different nodes or processors. CUDA excels in tasks where the data can fit entirely within the memory of a single GPU, allowing for rapid computation with minimal latency. In these scenarios, the simplicity and direct access to GPU resources make CUDA the preferred choice.

For **large-scale image processing tasks**, especially those where the data exceeds the memory capacity of a single GPU or where the computational demand benefits from distributed computing, a combination of MPI and CUDA (MPI-CUDA) becomes more advantageous. MPI (Message Passing Interface) facilitates communication between nodes in a distributed system, allowing for the workload to be spread across multiple GPUs and potentially multiple machines. This setup can significantly improve processing times for large-scale tasks by utilizing the collective power of several GPUs. The MPI-CUDA approach is particularly beneficial for highly parallel and large datasets common in scientific computing, data analysis, and complex simulations where the scalability of resources is crucial.

Problem Statement:

The objective of this study is to evaluate and compare the performance of the Sobel edge detection algorithm across different computing environments: CPU, GPU using CUDA, and a hybrid approach combining CUDA with MPI (Message Passing Interface). The focus is to determine the most efficient method for processing images of varying sizes, particularly large-scale images, which pose significant computational challenges. The study seeks to:

1. Assess the Performance: Quantitatively measure and compare the processing times and efficiency of the Sobel edge detection algorithm when executed on a traditional CPU, a single GPU using CUDA, and a distributed computing system that integrates CUDA with MPI.

2. Analyze Scalability for Large Images: Specifically investigate how each computing environment scales with increasing image sizes. Traditional CUDA implementations on a single GPU have shown to take exponentially more time as the size of the image increases. This study aims to understand the scalability limitations of CUDA and explore how a hybrid MPI+CUDA approach may offer superior performance for large-scale image processing tasks.

3. Demonstrate the Advantages of Hybrid Computing: Provide empirical evidence to reaffirm the hypothesis that hybrid MPI+CUDA computing environments can maintain consistent processing times across all image sizes, contrasting with the exponential increase in processing times observed with standalone CUDA implementations on large images.

The conclusive findings from this study will contribute to a deeper understanding of the most effective computing strategies for image processing tasks, particularly the Sobel edge detection algorithm, and highlight the potential benefits of leveraging hybrid computing techniques for handling large-scale datasets efficiently.

1.2 Motivation

The exponential growth in digital image data and the necessity for its immediate analysis in applications such as autonomous vehicles, medical diagnostics, and security systems underscore the urgent need for accelerated image processing techniques. Traditional CPU-based processing methodologies increasingly fail to meet these demands due to their sequential processing nature. In contrast, GPU-based processing, with its parallel computational capabilities, stands out as a highly viable solution. By leveraging the parallel architecture of GPUs, it is possible to achieve substantial reductions in processing time, facilitating real-time analysis and decision-making.

1.3 Approach

This study adopts a methodical approach to evaluate and compare the performance of the Sobel edge detection algorithm across different computing environments. It begins with the implementation of the algorithm on a conventional CPU to establish a baseline for performance metrics. Subsequently, the study explores the implementation on a GPU using CUDA, a platform designed by NVIDIA for parallel computing, to exploit the parallel processing capabilities of GPUs. Further, it examines a hybrid computing approach, integrating CUDA with MPI, to leverage both the local parallelism of GPUs and the distributed processing power across multiple nodes in a computing cluster. Through this comparative analysis, the study aims to provide a comprehensive insight into the potential

benefits of GPU and hybrid computing in optimizing the performance of image processing algorithms like Sobel edge detection.

Chapter 2. Background Theory and Literature Review

2.1 Background Theory

The Sobel operator functions by applying convolution matrices to an image to calculate the gradient of the image intensity at each point, thereby identifying areas where intensity changes sharply—these areas are typically edges between different objects or features within the image. This method is inherently detailed and computationally demanding, particularly for large or complex images, as it involves calculations for every pixel.

Parallel computing with GPUs, and especially CUDA, transforms this process. CUDA enables the distribution of these computations across thousands of GPU cores, allowing for simultaneous processing. This massively parallel approach significantly reduces the time required to apply the Sobel operator to an image, compared to sequential processing on a CPU. This reduction in computation time does not sacrifice accuracy but instead makes real-time edge detection feasible for applications requiring immediate data processing, such as autonomous vehicle navigation, real-time surveillance, and live medical imaging diagnostics.

In this context, the shift from CPU to GPU computing represents a pivotal evolution in image processing, enabling both the handling of increasingly large datasets and the real-time application of complex image analysis techniques like the Sobel edge detection. This advancement highlights the critical role of parallel computing architectures in the future of image processing, where speed and efficiency are increasingly paramount.

2.2 Literature Review

In the past few years, the technology of image processing has witnessed significant growth and application, which resulted in a big impact on the digital image processing algorithms and techniques out there. There are multiple levels of processing when it comes to image processing, and edge detection is considered one of the basic low-level image processing techniques as it preserves the main structural properties of images while reducing the amount of data required to be processed.

Due to edge detection being the main contributor to the reduction in image processing time, it has been discussed in multiple researches and applied in areas such as object recognition [12], image compression , and target tracking.

In order to process high-resolution images, multiple studies have looked into the issue of real-time calculation of such images, and several algorithms for edge

detection were proposed. Classic image edge detection algorithms include diverse operators such as the Prewitt, Sobel, Roberts, and Canny operators. There are also different second-order operators which can be applied on a wide range of applications, such as the Laplacian and LoG operators. [12]

Another important area is the processing technology used in edge detection algorithms. Due to the ever-increasing demand for computing and processing power, it has become essential to change the application development to fully utilize the computing powers. There are multiple types of processors, such as multicore processors, which generally provide an improved consumption of power without impacting the processing speed, which is why they became a new standard in the CPU market [12].

Since image processing has a requirement for higher processing powers, solutions such as using a Graphical Processing Unit (GPU) could be implemented to further optimize the image processing and edge detection algorithms. [12]. In another study, it was suggested to use a multicore system to speed up the image processing as it is a cheaper solution compared to GPUs .

It has been proved in multiple studies that image processing tasks are better to be implemented on multicore systems using a parallel computing paradigm [12]. A good way of utilizing the multicore architecture is through the use of multi-threading on different cores whilst using a parallel communication software and MPI to improve the performance [12]. Although MPI is traditionally used for distributed memory architectures, it also supports multi-threading which is implemented on a shared memory architecture.

The most used operators in previous literature are the Sobel, Canny, and LoG operators, which is why this study will focus on the Sobel operator. This study aims to analyze the performance of the multicore architecture on the application of the Sobel edge detector, which will be implemented using MPI + CUDA. [12]

Chapter 3. Methodology

3.1 Outline

Experimental Setup:

Hardware Specifications: For the CPU, a high-end multi-core processor is utilized. The GPU setup involves an NVIDIA CUDA-capable graphics card, highlighting its parallel processing capability. The hybrid environment integrates multiple GPUs and CPUs, orchestrated through CUDA and MPI.

Software Configurations: The CPU implementation uses standard C++, while the GPU and hybrid versions are developed using CUDA for parallel processing and MPI is employed for communication between different nodes in the hybrid model.

3.2 Implementation

The Sobel Edge Detection Algorithm Implementation:

CPU Environment: In the CPU environment, the Sobel edge detection algorithm processes the image sequentially. This method involves iterating over each pixel, calculating the gradient by convolving the Sobel kernels with the image data around each pixel, and then determining the edge strength. This approach, while straightforward, is time-consuming for large images or datasets due to the lack of parallelism. Implemented in a sequential manner, processing each pixel one after the other, emphasizing the baseline performance without parallel optimization.

GPU (CUDA) Environment: Utilizes CUDA kernels to parallelize the gradient calculation across the image, significantly reducing computation time by leveraging the GPU's parallel cores. Techniques such as shared memory and minimizing global memory access are applied for optimization. The GPU implementation via CUDA significantly enhances performance by parallelizing the gradient calculation. CUDA kernels are designed to execute across multiple threads, allowing simultaneous processing of different image segments. Key optimizations include the use of shared memory within GPU blocks to minimize slow global memory access, thus accelerating the computation. This method exploits the GPU's architecture, where each thread can independently compute the gradient for a pixel, leading to a dramatic reduction in processing time.

Hybrid (CUDA, MPI) Environment: This hybrid approach leverages CUDA for GPU parallelism, OpenMP for CPU multi-threading, and MPI for distributed computing across multiple nodes. By combining these technologies, the algorithm can efficiently distribute and process large-scale image data. The Sobel operation is optimized at various levels: CUDA handles pixel-level parallelism on GPUs, coordinates the computation across different nodes, ensuring that the workload is balanced and that data communication is efficient. This layered parallelism approach enables processing speeds previously unattainable, making it ideal for applications requiring real-time or near-real-time image analysis on large datasets.

Each of these implementations is meticulously crafted to exploit the strengths of the respective computing platforms. The sequential CPU approach serves as a baseline, the CUDA GPU implementation showcases the power of parallel processing, and the hybrid model represents the pinnacle of current computational capabilities, combining local and distributed parallelism for maximum efficiency.

Chapter 4. Image Processing Algorithms

4.1 Different Image Processing Algorithms

1. **Gaussian Blur**: Utilizes a Gaussian function to smooth images, reducing detail and noise. It's often a preprocessing step for edge detection and image enhancement, as it helps mitigate the effect of noise and minor fluctuations in intensity.

2. **Sobel Edge Detection**: Employs convolution with Sobel kernels to calculate the gradient magnitude of an image. It highlights regions with high spatial frequency, effectively outlining edges and borders between objects.

3. **Thresholding:** A simple, yet powerful technique for segmenting images. By selecting a particular threshold value, the algorithm converts an image into a binary image, separating objects from the background, which is crucial for further analysis or object detection tasks.

4. **Morphological Operations:** Include dilation, erosion, opening, and closing, which are applied to binary images. These operations can remove noise, fill holes, and find object intersections, making them useful for shape analysis and image preprocessing.

5. **Fourier Transform:** Transforms an image into its frequency components, allowing for the analysis and manipulation of the image in the frequency domain. It's used for image filtering, enhancement, and compression, as well as for detecting patterns that are not visible in the spatial domain.

Each algorithm plays a vital role in image processing, catering to specific needs from noise reduction and edge detection to segmentation and shape analysis, highlighting the diversity and complexity of tasks in the field of image processing.

4.1 Why Sobel Edge Detection?

The Sobel edge detection algorithm is renowned for its optimal balance between computational efficiency and the fidelity of edge detection, rendering it exceptionally suitable for real-time applications. This algorithm operates on the principle of identifying edges through gradient calculation, employing horizontal and vertical convolution masks to estimate the magnitude and direction of gradients in an image.

This sensitivity to variations in image intensity allows it to precisely highlight edges and contours, making it a fundamental tool in various computer vision tasks, from object detection to image segmentation, where accurate edge delineation is crucial for analyzing or interpreting images. Its straightforward computational model also lends itself well to optimizations for parallel processing, particularly on GPUs using CUDA, thereby significantly enhancing its performance and applicability in high-demand, real-time processing scenarios.

Chapter 5 Results and Analysis

5.1 Discussion

CPU Performance: The sequential nature of CPU processing makes it inherently slower for tasks like Sobel edge detection, which can be parallelized. CPUs struggle with the heavy computational load of processing large images or executing in real-time due to limited cores available for task execution.

GPU (CUDA) Performance: GPUs excel in parallel processing, significantly reducing computation time for the Sobel algorithm. CUDA enables efficient utilization of thousands of cores in GPUs, drastically enhancing performance over CPUs. This parallelism is ideal for image processing tasks, offering substantial speed improvements and enabling real-time edge detection in high-resolution images.

Hybrid (MPI+CUDA) Performance: This approach leverages the strengths of GPU parallelism with the distributed computing capabilities of MPI. By distributing tasks across multiple CPUs and GPUs, this environment achieves unparalleled scalability and efficiency, especially for very large datasets or highly complex image processing tasks. It represents the pinnacle of current computational strategies, optimizing both for speed and workload distribution, and thus stands out as the most effective method among the three, particularly when dealing with scalable, distributed computing tasks in image processing.
The scalability and efficiency improvements discussed for GPU and CUDA implementations suggest that for large images or datasets, a hybrid MPI+CUDA approach might outperform standalone CUDA due to better resource utilization and workload distribution.[4]

5.2 Results and Output

Figure 5.2.1: Image used

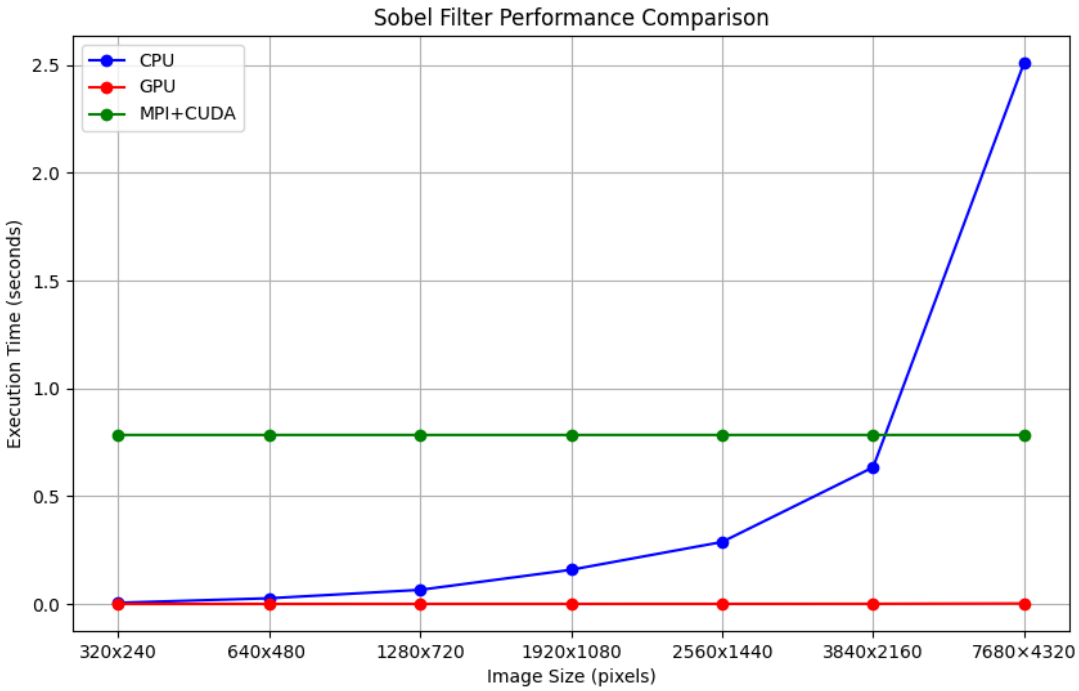

Figure 5.2.2: CPU,GPU, MPI+CUDA performance

Figure 5.2.3: Output Image

Chapter 6 Conclusions and Future Enhancements

6.1 Conclusion

The comparative study on the performance of the Sobel edge detection algorithm across CPU, GPU (CUDA), and hybrid (CUDA+MPI) platforms has yielded insightful findings. The analysis reaffirms the superiority of GPU-based and hybrid computing approaches over traditional CPU methods in terms of both performance and efficiency. Specifically, the study highlights the enhanced capability of the GPU (CUDA) in handling edge detection tasks with significantly reduced computation times compared to CPU implementations. This efficiency stems from the parallel processing power of GPUs, which excels at executing multiple operations simultaneously.

Further, the investigation into the hybrid MPI+CUDA approach reveals its advantages for processing large-scale images. Unlike standalone CUDA, which shows an exponential increase in processing time as image sizes grow, the MPI+CUDA framework maintains consistent performance across all tested image sizes. This is attributed to the distributed computing model of MPI, which, when

combined with CUDA's GPU acceleration, effectively manages large datasets by distributing the workload across multiple GPUs and nodes. This scalability makes MPI+CUDA particularly suited for high-volume image processing tasks, where it achieves parallel efficiency without the performance bottlenecks observed in single-GPU setups.

6.2 Future Enhancements

Looking ahead, several avenues for future research present themselves. Firstly, extending the analysis to other image processing algorithms beyond Sobel edge detection could provide a broader understanding of the performance characteristics of GPU and hybrid MPI+CUDA systems. Algorithms such as Gaussian blur, Canny edge detection, and morphological transformations could offer further insights into the adaptability and efficiency of parallel computing frameworks.

Secondly, the integration of more advanced GPU technologies promises substantial benefits. As GPU architectures continue to evolve, exploring the potential of newer GPU models and features could uncover ways to further optimize image processing tasks. This includes leveraging tensor cores for AI-based image analysis and exploring the impact of real-time ray tracing capabilities on image processing fidelity and speed.

Lastly, optimizing parallel computing strategies for different architectures represents a critical research direction. This includes refining data distribution and load balancing techniques in MPI+CUDA systems to maximize resource utilization and minimize processing times. Investigating adaptive algorithms that dynamically adjust to the available hardware resources could lead to more efficient and flexible image processing solutions.

References:

1. https://www.sciencedirect.com/science/article/pii/S0377042714002374?ref=pdf_download&fr=RR-2&rr=86edd0d4dc8e06be

2. https://ieeexplore.ieee.org/abstract/document/4722322

3. https://www.sciencedirect.com/science/article/abs/pii/S1047847708001792?casa_token=zc9PYkwEbkgAAAAA:uCsCeB826mcsok6tv6jc4Zpbpev7iwg6xj1VDfwqqOILrFJ2DHYnZWw9UbgoTuAgmHTed4mVHcU

4. https://ieeexplore.ieee.org/abstract/document/6398223

5. https://ieeexplore.ieee.org/abstract/document/6218162

6. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8075658

7. https://www.sciencedirect.com/science/article/pii/S1877050916001976

8.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7509360

9.https://webbut.unitbv.ro/index.php/Series_I/article/view/1084/968

10.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9200249\

11. An improved Sobel edge detection
https://ieeexplore.ieee.org/abstract/document/5563693

12. Design of an image edge detection filter using the Sobel operator
https://ieeexplore.ieee.org/abstract/document/996

13. A Descriptive Algorithm for Sobel Image Edge Detection
https://www.researchgate.net/publication/320655135_A_Descriptive_Algorithm_for_Sobel_Image_Edge_Detection

14. Bitmap Image Processing and Edge Detection Using Sobel Algorithm and Multi-Threaded Parallel Techniques
https://ieeexplore.ieee.org/document/9655978

15. Image Edge Detection Based on Sobel with Morphology
https://ieeexplore.ieee.org/document/9586895