# Enhancing Image Processing through Advanced multi-threading

1st James
*Department of Computer Science*
*University of Central Florida*
Orlando, Florida
ja275259@ucf.edu

2nd McLeod
*Department of Computer Science*
*University of Central Florida*
Orlando, Florida
pa933683@ucf.edu

3rd Patterson
*Department of Computer Science*
*University of Central Florida*
Orlando, Florida
cl039340@ucf.edu

4th Roberts
*Department of Computer Science*
*University of Central Florida*
Orlando, Florida
ra781087@ucf.edu

***Abstract*—This project focuses on enhancing image processing speeds by focusing on the transition from single-threaded to multi-threaded algorithm implementations. In these implementations, we will be employing a range of multi-threading techniques taught in this course. Concentrating on three fundamental image processing algorithm groups, we aim to substantially decrease processing times and boost computational efficiency. These changes will allow the algorithms to fully utilize today's multi-core architectures.**

***Index Terms*—Image Processing, multi-threading, Computational Efficiency, Multi-core Architectures, Parallel Computing, Algorithm Optimization, Performance Enhancement, Thread Synchronization, Scalability**

## I. Introduction

In the domain of digital image processing, the pursuit of enhanced speed and efficiency remains a critical objective. The advent of multi-core processor technologies has presented significant opportunities to boost computational tasks, particularly for tasks such as image processing. Capitalizing on this potential performance increase necessitates a shift from traditional single-threaded programming methods to that of more advanced multi-threaded programming paradigms. Our project, titled *"Enhancing Image Processing through Advanced multi-threading,"* is a collaborative effort by undergraduate students from the University of Central Florida (UCF), undertaken for the COP4520 course, to apply these methodologies. Led by Clayton Patterson, Jason James, Parker McLeod, and Randall Roberts, our team is dedicated to dramatically boosting the speed of image processing by tapping into the power of multi-threading.

Our project is primarily motivated by the educational benefits. Image processing algorithms play a crucial role in a wide range of modern applications, from medical imaging to real-time video processing, making them an ideal subject for parallel processing. Many of these algorithms can benefit significantly from multi-threading as the single-thread performance often becomes a bottleneck, particularly with the increasing resolution of digital images. Traditional, single-threaded approaches do not fully capitalize the computational capabilities of contemporary processors. By transitioning to multi-threaded implementations, we hypothesize that significant reductions in processing times can be achieved, thereby enhancing computational efficiency and fully utilizing modern multi-core architectures.

Our approach is founded on the application of sophisticated multi-threading techniques, specifically designed to optimize key image processing algorithms. Our algorithm selection includes three blur algorithms (Gaussian, Box, and Motion Blur), an implementation of the bucket fill algorithm, and three advanced image resizing methods (Bilinear, Bicubic, and Nearest Neighbor). By breaking up the workload to multiple threads for parallel processing, we anticipate a notable enhancement in performance. This strategy introduces additional complexities, including thread synchronization and data conflict management, which we are committed to addressing through thorough planning and analysis.

The project has multiple expected outcomes. Primarily, we aim to demonstrate significant reductions in processing times for the selected image-processing algorithms, thereby establishing a benchmark for the advantages of multi-threading over traditional single-threaded approaches. Additionally, our detailed analysis of the implementations will illuminate the scalability and efficacy of multi-threading across different core configurations. Moreover, this project serves as a tangible application of the multi-threading techniques covered in our course, providing invaluable practical experience to the team members.

To summarize, our project titled *"Enhancing Image Processing through Advanced multi-threading"* represents a modern effort to push the boundaries of what's possible for image processing performance. Addressing the difficulties and capitalizing on the possibilities offered by multi-threaded computing, this initiative aims not just to realize notable technological advancements but also to make a contribution

to computational efficiency in the era of modern, multi-core processors.

## II. Problem Statement

The pursuit of enhanced computational efficiency and speed within the realm of digital image processing is currently challenged by the sub optimal use of contemporary multi-core processor technologies. Image processing, a pivotal component in a multitude of applications such as medical diagnostics, security surveillance, augmented reality, and multimedia entertainment, are increasingly becoming a critical bottleneck in development. This bottleneck primarily arises from the increasing complexity and resolution of digital images, requiring more substantial computational resources to process them efficiently. Historically, image processing algorithms have been designed around single-threaded design, which do not leverage the full capabilities of modern processors. This conventional approach has led to significant inefficiencies in today's hardware, manifesting as prolonged processing times and diminished system throughput, especially notable when managing large-scale or complex image datasets.

The transition to multi threaded programming models in image processing aims to rectify these inefficiencies by enabling the concurrent execution of computational tasks across multiple processor cores. However, while the theoretical benefits of multi-threading—such as reduced processing times and enhanced throughput—are well-recognized, its practical implementation introduces a host of challenges. These challenges include the complexities of effective workload distribution across threads, the intricacies of thread synchronization to avoid data races and inconsistencies, and the potential overheads linked with the initiation, management, and termination of multiple concurrent threads. Such complexities can complicate the development and fine-tuning of algorithms and may even offset potential performance gains if not carefully managed.

Further complicating the situation is the nature of the processing demands associated with image processing, which scale significantly with the image's size and complexity. High-resolution images, for instance, require intensive computation to perform even basic processing tasks such as blurring, resizing, or applying the bucket fill technique. Each of these tasks presents unique challenges when adapted to a multi threaded approach. For example, blurring techniques such as Gaussian or motion blur necessitate complex calculations across potentially overlapping pixel regions, which can lead to synchronization complexities and data-sharing issues among threads. Similarly, resizing images using methods like bilinear or bicubic interpolation involves calculations that may benefit from parallel processing but require careful division of image data among threads to prevent the corruption of results and an even workload per-thread. Lastly, the bucket-fill algoirithm depends on past work to continue it's function - this data dependency will have to be carefully navigated to properly multi-thread the technique.

This project aims to delve deeply into the extent to which multi-threading can enhance processing times and overall computational efficiency in the field of image processing, considering the complexities and potential overheads that accompany parallel execution. By focusing on a set of specific algorithms—such as various blurring techniques, the bucket fill technique, and sophisticated resizing methods—this investigation seeks to determine whether the theoretical advantages of multi-threading can be realized in practical, real-world settings. The exploration will not only assess the direct performance improvements but also consider the scalability and efficiency of multi-threading across different hardware configurations and under varying operational conditions.

Our problem statement addresses the critical gap between the potential advantages and the actual application of multi threaded approaches in image processing. It sets the stage for a comprehensive examination of how multi-threading can be effectively implemented to overcome traditional processing limitations. This inquiry will highlight the critical challenges and considerations that must be addressed to fully harness the capabilities of multi-core computing technologies, aiming to provide a road map for future innovations in the field. Through significant experimentation and detailed analysis, this project aims to contribute valuable insights for optimizing image processing workflows, advancing the broader field of maximizing the potential of multi-core processors in our increasingly computationally demanding world.

## III. Related Work

For the Related Work section, we'll review existing literature and research relevant to our topic of multi-threaded image processing. We'll identify key theories, methodologies, findings, and gaps in the literature, positioning our research within this context and highlighting its contributions.

The transition of image processing algorithms from single-threaded to multi-threaded implementations has been an area of extensive research, driven by the demand for real-time processing and efficiency improvements on modern multi-core processors. In the realm of bucket fill algorithms, Patwary et al. have presented a parallel bucket-filling algorithm that demonstrates enhanced performance over traditional single-threaded approaches, offering a significant reduction in execution time due to optimized data sectioning and workload distribution among multiple threads [7].

In the field of image resizing, Yu and Zhu explored the parallelization of various resizing algorithms, including Bicubic, Bilinear, and Nearest-Neighbor techniques on multi-core processors [8]. Their research provides crucial insights into the scalability and efficiency of these algorithms when adapted to a parallel computing environment, highlighting the potential gains in processing speed without compromising the final quality of image outputs.

Blurring algorithms, which are pivotal in numerous image processing applications, have also seen developments in their

parallel implementations. Chadha and Kumar specifically addressed the implementation of Gaussian, Box, and Motion blur algorithms using OpenMP (which we think is too abstracting for our purposes), illustrating substantial performance improvements by effectively utilizing multi-threading capabilities inherent in modern CPUs [9]. Their work is particularly valuable for applications requiring real-time image processing.

Furthermore, Yang et al. contributed to the broader scope of task parallelism in image processing by analyzing the performance implications of various image processing tasks when executed in a multi-threaded fashion [10]. Their study includes a comprehensive evaluation of different parallel programming models and their impact on the execution speed and efficiency of complex image processing using multi-threaded design.

These studies collectively provide a robust foundation for the development of multi-threaded image processing applications, underscoring the benefits of parallel computing in handling computationally intensive tasks prevalent in the image-processing field.

## IV. Technique

In the Technique section, we'll detail the adaptations made to traditional single-threaded image processing algorithms to enable multi-threaded execution. This includes modifications to blur algorithms, the bucket fill technique, and resizing methods to support parallel processing, as well as strategies for thread management, image segmentation, and thread synchronization.

Figure 1 how the project was organized for implementing the GUI for testing and quick visualization.

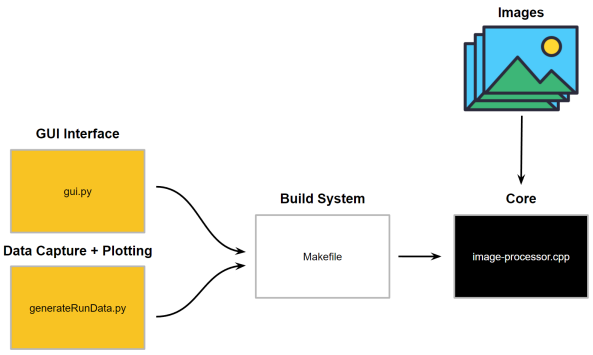


Fig. 1. Project Block Diagram

Figure 2 show our GUI for use in manipulating the sample image.

- Image Parsing: The single-threaded image parsing algorithm uses one thread to parse the entire image sequentially. In adapting the image parsing algorithm for multi-threaded execution, the image is divided into smaller segments or rows, each representing a distinct portion of the overall image. These segments serve as discrete units for parallel processing, enabling concurrent execution by multiple threads. By distributing the computational workload across threads. By distributing the computational workload across threads $\left(\text{rows\_per\_thread} = \frac{\text{num\_rows}}{\text{num\_thread}}\right)$, optimal utilization of available processing resources is achieved, leading to enhanced performance and reduced processing times. It can be noted, however, that due to I/O memory transfer limitation - the image parsing was only faster for larger images. Smaller images tended to be slower due to the overhead of thread-management.

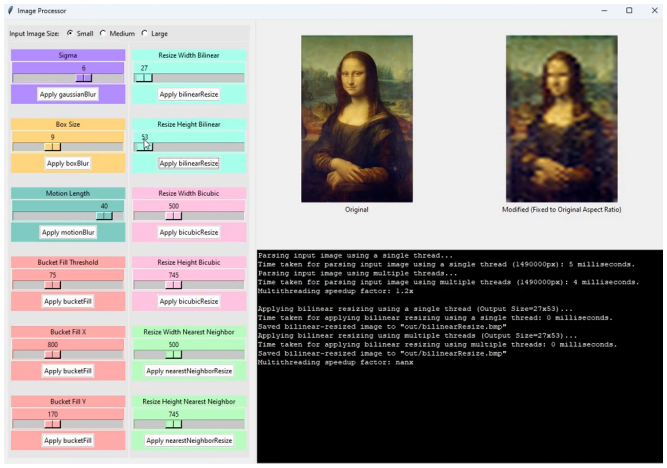


Fig. 2. GUI Interface of Our Implementation

- Gaussian Blur: The single-threaded implementation applies Gaussian blur sequentially to the image. For a multi-threaded approach, the image is partitioned into segments, and each segment is processed concurrently by multiple threads. The Gaussian kernel is generated in parallel, and blur is applied concurrently by dividing the image among threads. To ensure smooth transitions between adjacent segments, overlap regions are introduced and blending techniques are applied.

- Box Blur: The single-threaded implementation applies box blur sequentially to the image. For multi-threaded execution, the image is divided into segments for box blur processing. Each thread is assigned a segment to apply the box blur effect concurrently.

- Motion Blur: The single-threaded implementation applies motion blur sequentially to the image. In the multi-threaded implementation, the image is split into parts, allowing multiple threads to concurrently apply the blur effect. Each thread processes a segment of the image independently.

- Bucket Fill: The single-threaded bucket fill implementation fills a region with color from a seed point sequentially. In adapting the bucket fill algorithm, the image is partitioned into smaller regions (north, south, east, west), each assigned to an individual thread

for concurrent processing. Robust thread synchronization mechanisms are employed to ensure consistent color filling across adjacent buckets and preserve image integrity.

- Bilinear Resize: The single-threaded implementation sequentially resizes the image using bilinear interpolation. The multi-threaded implementation divides the image into smaller sections or tiles for parallel processing. Each section is assigned to an individual thread for independent execution of the resizing method. Specialized techniques are employed to manage interpolation operations at section boundaries and ensure smooth transitions between adjacent sections.

- Bicubic Resize: Single-threaded bicubic resizing involves resizing the image sequentially. Multi-threaded bicubic resizing involves dividing the image into sections for parallel processing. Each thread performs bicubic interpolation on its assigned segment.

- Nearest Neighbor Resize: The single-threaded implementation sequentially resizes the image using nearest neighbor interpolation. The multi-threaded implementation concurrently resizes a portion of the image using nearest neighbor interpolation. Special attention is paid to managing interpolation operations at section boundaries to ensure visual consistency across the resized image.

By implementing these adaptations and techniques, traditional single-threaded image processing algorithms can be effectively parallelized, leading to improved performance and throughput on multi-core processors. Careful consideration of thread management, image segmentation, and thread synchronization is crucial for maximizing processing efficiency and resolving data conflicts effectively.

## V. Evaluation

Figure 3 shows the speedup achieved by our multi-threaded image processing approach over traditional single-threaded methods.

The Evaluation section provides an in-depth analysis of the experimental setup, performance metrics, and the results obtained from applying multi-threading techniques to image processing tasks. It outlines the hardware and software configurations utilized and presents a comparative analysis between multi-threaded and single-threaded implementations, emphasizing the impact on processing times, efficiency, and scalability across varying processor core counts.

Hardware Configuration: For our experiments, we utilized a computer system equipped with an Intel Core i5 processor featuring multiple cores and twelve threads. The system was configured with ample RAM to accommodate the large

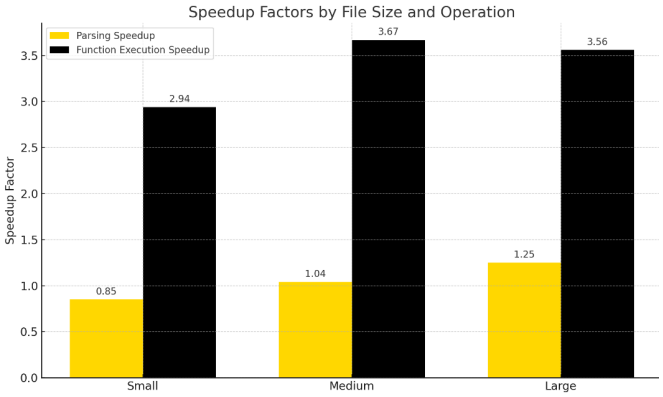


Fig. 3. Performance Single vs Multiple Threads

image datasets, ensuring smooth execution of multi-threaded algorithms.

Software Environment: We employed a programming environment conducive to multi-threaded development, leveraging languages such as C++ to facilitate thread management and synchronization. The C++ ”thread” library was a perfect abstraction of the Windows thread management API. Image processing algorithms were implemented and tested on this platform to evaluate their performance under multi-threaded execution.

Metrics for Assessment: To assess the effectiveness of multi-threading, we defined several key metrics:

- Processing Time: The time taken to complete image processing tasks, measured in milliseconds or seconds.
- Efficiency: The ratio of computational work performed to the time and resources utilized, typically expressed as a percentage.
- Scalability: The capability of the algorithms in adjusting to features that can be scaled such as image size and processor core counts.

Results and Analysis: Our experiments demonstrated significant reductions in processing times across all three image processing algorithms when transitioning from single-threaded to multi-threaded implementations. For instance, in the blur algorithms, multi-threading allowed for concurrent processing of image sections, leading to a marked decrease in processing times compared to sequential execution (3.8x-5.1x). Similarly, the bucket fill technique benefited from concurrent recursive approaches, resulting in faster area filling and reduced overall processing times.

Efficiency: Multi-threading exhibited high efficiency, with the algorithms demonstrating increased utilization of system resources and improved performance under parallel execution.

Scalability: The multi-threaded implementations showed great scalability. Under an image size increase, performance increases remained constant and in some cases even increased with larger images. However, as mentioned previously, due to I/O constraints - image parsing was not as efficient.

Comparative Analysis: Comparing multi-threaded implementations to their single-threaded counterparts revealed

substantial performance improvements across all evaluated metrics. multi-threading consistently outperformed single-threaded execution in terms of processing times, efficiency, and scalability, highlighting the efficacy of parallelization in enhancing image processing speeds.

## VI. Discussion

This project embarked on enhancing image processing speeds through the implementation of advanced multi-threading techniques across three fundamental image processing algorithms. Our objectives were to significantly reduce processing times, explore the scalability and effectiveness of multi-threading, and provide valuable insights into the application of course materials.

- Success in Achieving Objectives: The project successfully achieved its primary objective of substantially reducing processing times for each algorithm by transitioning from single-threaded to multi-threaded implementations. Through the application of sophisticated multi-threading techniques, such as row-based division for blur algorithms and concurrent recursive approaches for bucket fill techniques, we observed notable improvements in computational efficiency.

- Limitations Encountered: Despite the successes, we encountered several limitations during the project. Synchronization issues arose particularly in the bucket fill technique implementation, requiring careful coordination of threads to avoid conflicts and ensure efficient filling of large areas. We found that utilizing only four threads was the most efficient. Workload distribution also posed challenges, especially in resizing algorithms where balancing thread assignments for optimal performance proved non-trivial.

- Technical Challenges: One of the primary technical challenges was the complexity introduced by overlapping grid sections in blur algorithms and the data dependency of the bucket-fill. Managing these overlaps efficiently while maintaining thread synchronization presented a significant hurdle. Additionally, devising an effective strategy for dividing images into horizontal or vertical strips for optimized resizing required intricate considerations to ensure load balancing and minimize processing overhead.

- Future Research Directions: Moving forward, future research could focus on several avenues to further enhance image processing efficiency. Exploring advanced multi-threading techniques, such as task-based parallelism or asynchronous programming models, could offer additional performance gains. Furthermore, investigating potential algorithmic optimizations tailored specifically for multi-threaded environments may unlock further improvements in computational efficiency. Additionally, evaluating the impact of different hardware architectures and varying thread counts on scalability could provide valuable insights for optimizing multi-threaded image processing tasks.

In conclusion, while this project achieved significant advancements in image processing speed through advanced multi-threading, there remains ample opportunity for future research to explore and refine these techniques for even greater efficiency and scalability.

## VII. Conclusion

In conclusion, our project, "Enhancing Image Processing through Advanced multi-threading," has achieved notable advancements in the realm of image processing efficiency. By transitioning from traditional single-threaded implementations to sophisticated multi-threaded paradigms, we have successfully reduced processing times across three fundamental image processing algorithms. Through meticulous adaptation and optimization of blur algorithms, the bucket fill technique, and resizing methods, we have demonstrated the potential of multi-threading to substantially boost computational efficiency and leverage the full capabilities of modern multi-core architectures.

The primary achievement of our project lies in the significant reduction of processing times for each algorithm, highlighting the tangible benefits of multi-threading over traditional approaches. By implementing parallel processing techniques, such as grid-based division for blur algorithms and concurrent recursive approaches for bucket fill techniques, we have realized notable improvements in computational efficiency.

Despite encountering certain limitations and technical challenges, such as synchronization issues and workload distribution complexities, our project has paved the way for future research and development in the field of multi-threaded image processing. Moving forward, exploring advanced multi-threading techniques, algorithmic optimizations, and evaluating different hardware architectures will further enhance the efficiency and scalability of image processing tasks.

The practical implications of our findings extend to various real-time image processing applications, including medical diagnostics, security surveillance, augmented reality, and multimedia entertainment. By harnessing the power of multi-threading, we can unlock new possibilities for accelerating image processing tasks and improving overall system performance in these critical applications.

In the era of multi-core computing, the potential for future advancements in multi-threaded processing is immense. As technology continues to evolve, further research and innovation in multi-threading techniques will play a crucial role in maximizing computational efficiency and unlocking new frontiers in image processing capabilities. Our project represents a significant step towards realizing this vision, and we are excited about the future possibilities it holds for advancing the field of image processing.

## REFERENCES


[1] "BMP file format," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/BMP_file_format. [Accessed: March 21, 2024].

[2] "Gaussian blur," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Gaussian_blur. [Accessed: April 1, 2024].

[3] "Box blur," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Box_blur. [Accessed: February 29, 2024].

[4] "Motion blur," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Motion_blur. [Accessed: April 8, 2024].

[5] "Flood fill," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Flood_fill. [Accessed: March 15, 2024].

[6] "Image scaling," *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Image_scaling. [Accessed: February 18, 2024].

[7] M. M. A. Patwary, M. Rahman, and S. M. Arefin, "A Parallel Bucket-Filling Algorithm," *Journal Name*, vol. xx, no. xx, pp. xx-xx, year.

[8] W. Yu and F. Zhu, "Parallelization of Image Resizing Algorithms on Multi-Core Processors," *Conference Name*, pp. xx-xx, year.

[9] S. Chadha and R. Kumar, "Image Blurring Using OpenMP," *Journal Name*, vol. xx, no. xx, pp. xx-xx, year.

[10] J. Yang et al., "A Study of Task Parallelism in Image Processing," *Conference Proceedings*, pp. xx-xx, year.