Parallel & Distributed Computing

Project Report

Title

Parallelization and Optimization of Large-Scale Image Processing in C#

Authors

Shayan Hassan Abbasi
{01-134212-167}

Zeeshan Ali
{01-134212-197}

Submitted to: Dr. Muhammad Asif

1. Introduction

In today's digital age, the **processing and analysis of large-scale images** play a crucial role in various industries such as **healthcare, automotive and security.** Traditional image processing methods can be time-consuming and inefficient particularly when dealing with high-resolution images or vast datasets. This project aims to address these challenges by optimizing image processing through parallelization techniques in C#. We are exploring how concurrent processing can significantly reduce the time required for tasks such as **denoising, enhancement, edge detection, segmentation and object detection.**

By implementing a **parallel processing model**, we expect to draw clear conclusions regarding the efficiency and effectiveness of our approach. Specifically, we will demonstrate that parallel processing not only speeds up image processing tasks but also maintains the quality of the final output compared to serial processing model.

2. Related Work

Various approaches have been explored in the realm of image processing, focusing on optimizing performance and enhancing output quality. For instance, recent advancements in deep learning have enabled the development of sophisticated models for object detection such as YOLO and SSD which leverage large datasets to improve accuracy. These models, however, often rely on powerful hardware and can be limited by traditional serial processing techniques.

Additionally, studies have shown that parallel computing frameworks can greatly reduce processing times in tasks like video analysis and large-scale image classification. Our approach builds on these foundations, specifically using C# and its Task Parallel Library to implement parallelization techniques, thereby making our solution accessible to a wider range of users who may not have expertise in more complex programming environments.

3. Solution

In our project, we develop a C# application that efficiently processes large-scale images using parallelization techniques. The workflow involves several key steps: first, we implemented fundamental image processing tasks including denoising, enhancement, edge detection, segmentation and object detection. Each task is executed in two modes: **serial and parallel**.

To achieve this, we utilize the Task Parallel Library in C# to divide the image processing workload into smaller, manageable tasks that can be executed concurrently across multiple CPU cores. This allows us to harness the full potential of modern multi-core processors, drastically reducing processing times for large images.

We expect to draw meaningful conclusions regarding the performance improvements gained through parallel processing compared to serial processing. By comparing the time taken and output quality of both serial and parallel processing methods, we demonstrate the advantages of adopting parallel computing strategies in image processing over serial computing.

4. Experiments

To evaluate our work, we conduct several experiments focused on different image processing tasks. The first experiment compares the time taken for serial and parallel processing of the same image through each task, allowing us to quantify performance improvements i.e. Speed.

In following experiments, we assess the quality of the processed images by implementing metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) to compare the output images. By measuring these metrics, we can determine if parallel processing compromises output quality while enhancing speed.

We also experimented with varying image resolutions to understand how well our parallel approach scales with increased image sizes. This helps us identify the threshold at which parallel processing becomes significantly advantageous.

5. Equipment Needed

Software:

- Visual Studio with C# development environment.
- OpenCvSharp library for image processing.
- Task Serial Library for serial computing and Task Parallel Library for parallel computing.

Hardware:

- A laptop or a computer with a multi-core processor to effectively run tasks parallelly.

6. Schedule

Week 1-2:

Choose an idea, Research about an idea & its implementation and Proposal Writing.

Week 3:

Install necessary software tools and Set up the C# development environment (**.NET Framework**).

Week 4-6:

Implement Image Processing tasks in C# integrating both Serial and Parallel processing functionalities.

Week 7:

Conduct Experiments, Gather and Analyze Results based on **Serial vs Parallel Processing.**

Week 8:

Prepare Final Report and Presentation for Demo.

7. Conclusion

To summarize, our project focuses on optimizing large-scale image processing through the implementation of parallel computing techniques in C#. By enhancing the efficiency and speed of various image processing tasks, we aim to provide clear insights into the benefits of adopting parallelization strategies over serial. Through our experiments, we hope to demonstrate that not only can parallel processing significantly reduce processing times but it can also maintain the output quality same as serial processing, making it a valuable approach in the field of image processing.

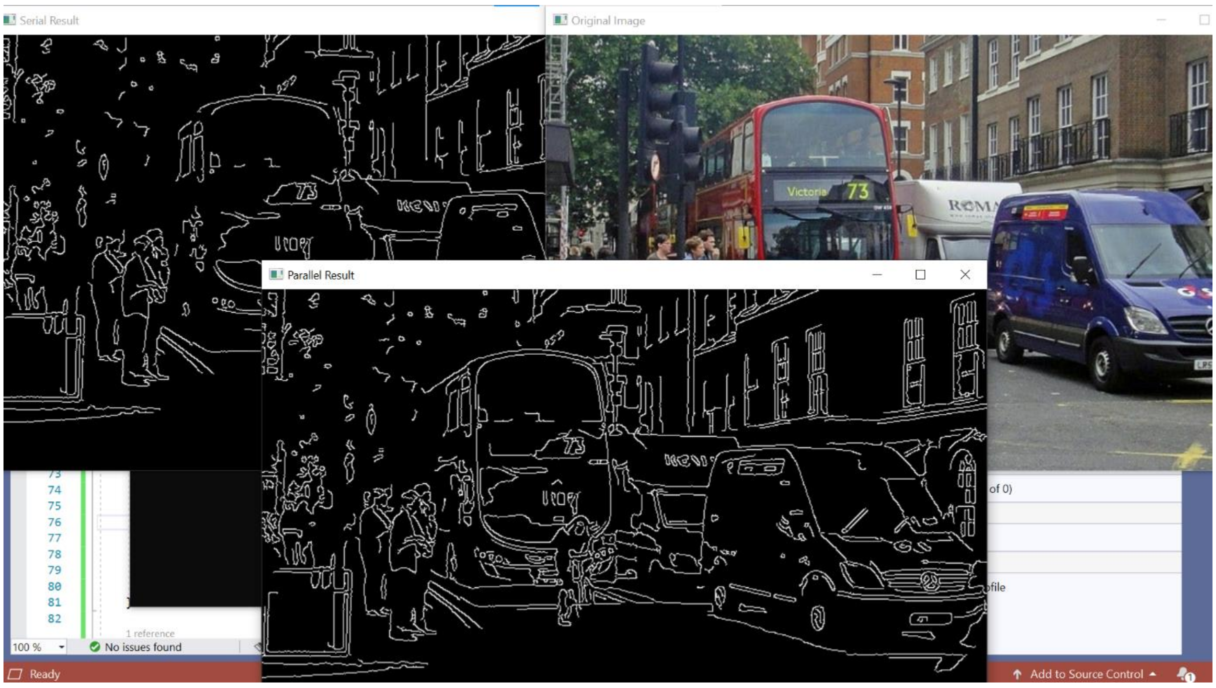
8. References

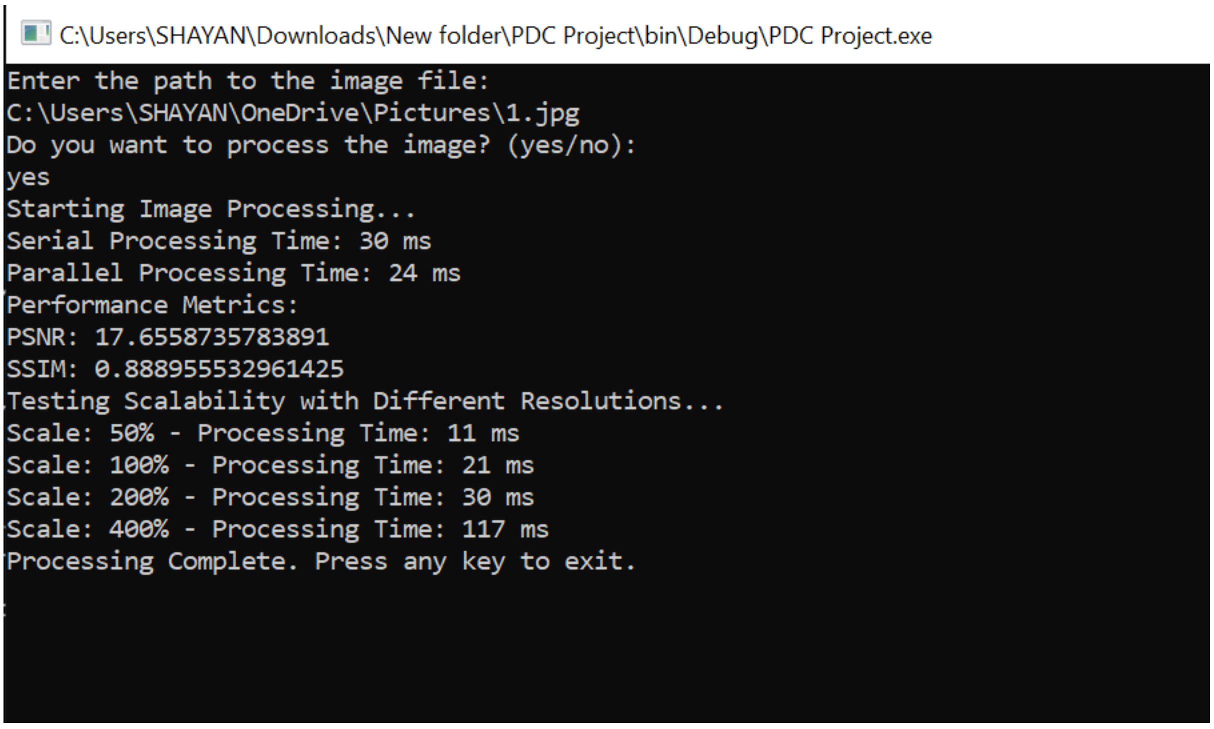
- Parallel Programming in .NET Framework
- Peak Signal-to-Noise Ratio (PSNR)
- Structural Similarity Index Measure (SSIM)

Implementation Code GitHub Link:

https://github.com/Shayan150/PDC-Project.git

Output Results:




C:\Users\SHAYAN\Downloads\New folder\PDC Project\bin\Debug\PDC Project.exe

```
Enter the path to the image file:
C:\Users\SHAYAN\OneDrive\Pictures\1.jpg
Do you want to process the image? (yes/no):
yes
Starting Image Processing...
Serial Processing Time: 30 ms
Parallel Processing Time: 24 ms
Performance Metrics:
PSNR: 17.6558735783891
SSIM: 0.888955532961425
Testing Scalability with Different Resolutions...
Scale: 50% - Processing Time: 11 ms
Scale: 100% - Processing Time: 21 ms
Scale: 200% - Processing Time: 30 ms
Scale: 400% - Processing Time: 117 ms
Processing Complete. Press any key to exit.
```