Image Processing using CUDA

October 2023

By

Dawid Kisielewski

Student number 202106560

Word count: 1775

Contents

1. Project background and purpose

1.1. Introduction

Image processing is the process of computing an image to enhance it or modify the image in a way to ease the process of extracting data from it. This process can often involve removing graininess and noise from an image, segmenting an image, detection of edges, and many other different processes (Niblack, W. 1986). Graphical Processing Units (GPU) have been mainly used as 3D graphics accelerators for rendering pipelines such as OpenGL or DirectX but after gaining cores people realised that moving simple parallel tasks off the CPU and onto the GPU would be more efficient. As shown by Castaño-Díez's study in 2008, by using the GPU the time for image processing loads to be completed decreases (Castaño-Díez, D. 2008).

For the purposes of this project, the only 3 types of algorithms I will be using will be edge detection, segmentation and noise reduction. Edge detection is the process of extracting the edges in an image. This results in an image containing only the edges found in the image. Segmentation separates an image into separate parts allowing easier analysis of the image. This is often used when doing medical imagery. When taking images of organs, they usually result in an image in which it's hard to tell areas apart. Image segmentation alleviates this issue by colouring separate areas in different colours making each section distinct. Noise reduction refers to the process of removing unwanted artefacts from an image. Taking images will always result in there being noise. It could be the artefacts left after ray tracing, radio interference in a radio telescope or even radiation hitting the image sensor of a digital camera. All of these would need to be treated with a noise reduction algorithm.

CUDA is a closed-source API that allows programmers to use the GPU for computing. CUDA was developed by Nvidia, which means that only Nvidia GPU have the ability to run CUDA code. Many different languages can be used to program CUDA, some notable examples are C++, Python and Fortran there are many other languages too.

1.2. Objectives

- Create 3 image processing algorithms. Each algorithm needs to have a CPU version and a GPU version. One algorithm will be an edge detection algorithm, another one be a segmentation algorithm and the last one will be a noise reduction algorithm. each algorithm explores separate aspects of image processing giving me more representative data for differences between the CPU and the GPU.
- Analyse the performance differences between the algorithms running on the CPU compared to the GPU. I will run the algorithms on various image resolutions to see if there is a different impact on both the GPU and CPU depending on the resolution of the image. I will collect the time it takes to run the algorithm as well as the end results.
- All my findings will be compiled into a document. This document will contain both the GPU and CPU versions of the algorithms as well as an explanation of what the code does. The performance of each algorithm will also be documented. Both the time it takes depends on the resolution of the image and the results of the algorithms.
- Any findings will be written in the conclusion whether a certain resolution is better to run on the CPU or GPU, or the relationship between time and resolution on the CPU and GPU.

1.3. Scope

To do image processing the program needs to have the ability to read from an image file. For this project, I will be using a library that already has this functionality as my project’s only interest is image processing and not loading and saving images.

1.4. Deliverables

The data collected from all my algorithms will be written into a document. This document will also contain explanations of how the algorithms work as well as comparisons between all the algorithms' performance.

1.5. Constraints

Many different APIs allow computation on the GPU such as OpenCL. For this project, I will only be using CUDA as I already have access to computers with CUDA-compatible GPUs. CUDA has an extensive library of example programs that will help in learning and optimising my code.

2. Project rationale and operation

2.1. Project benefits

The PDD is a document explaining and proposing my project. It contains my plan and my methodology in which I will complete my project.

By the end of the project, I will have a document containing all my findings on the performance differences between the algorithms on the CPU compared to the GPU. Each algorithm will have a section with the code itself as well as some outputs of the algorithm and the time it takes to process images of different resolution.

The skills I develop by the end of the project will also be invaluable to my future as a computer programmer. I will improve my skills using the programming language I choose. I will also gain the ability to program and optimize compute algorithms on the GPU.

2.2. Project operation

For this project, I'll be implementing the Agile workflow to help adapt to any issues I come across. I'll be running sprints with specific targets to meet. At the end of each sprint, I'll reflect on the work I have done and then evaluate my Gantt chat and deadlines as well as if there is any additional decomposition required.

2.3. Options

CUDA can interface with a variety of different programming languages. C++, Python, C#, Fortran, and others all can be used when doing CUDA programming. Deciding which language to use is important as it will affect the performance of the algorithms. After picking a language I'll need to pick a library that fits the project requirements. Many different libraries fit, such as OpenCV for C++ and Python, ImageProcessor for C#, and many others. Finally, many different image processing algorithms do different things. I will need to decide which ones to implement. My target is to implement at least 3 different algorithms. I have decided on the families of algorithms I will implement. Edge detection, segmentation and noise reduction are these 3 families.

2.4. Risk analysis

Hazard	Risk	Mitigation	Likelihood	Severity	Impact	Residual Impact
Illness	Harder to focus and Concentrate on the project meaning more time wasted extending the project's time to complete	Plan time to work on tasks a bit longer to compensate for any wasted time. Take medicine and recover quickly	High - 4	Low - 2	8	8
Data Loss	Project files being lost due to failure of hardware and or data corruption	The use of git and GitHub for cloud backups. Committing work after every task.	Low - 2	Very High - 5	10	4

Loss of Internet Service	Losing the internet will slow down my progress in investigating literature and researching any problems I encounter in my project	Going to work on campus as the internet on campus is free to use for students. Report the issue to my landlord to get the problem fixed as quickly as possible	Low - 2	Medium - 3	6	4
GPU Breakage	Loss of the GPU means that the CUDA program will not be able to pe run on the computer	Going to work at the University high performance lab would allow me to continue my work	Low - 2	Hight - 4	8	4
Underestimated time of delivery	The time span I gave myself to implement all the tasks and objectives was too small resulting in me not completing the project	Have reflections at the end of every sprint to evaluate my progress and make changes as necessary	Low - 2	Very high - 6	12	6

2.5. Resources required

As I will be using the CUDA API for this project, I will require a computer with a CUDA-compatible GPU to implement and develop my image processing algorithm. This also means any demonstrations I preform will require me to have a Nvidia GPU.

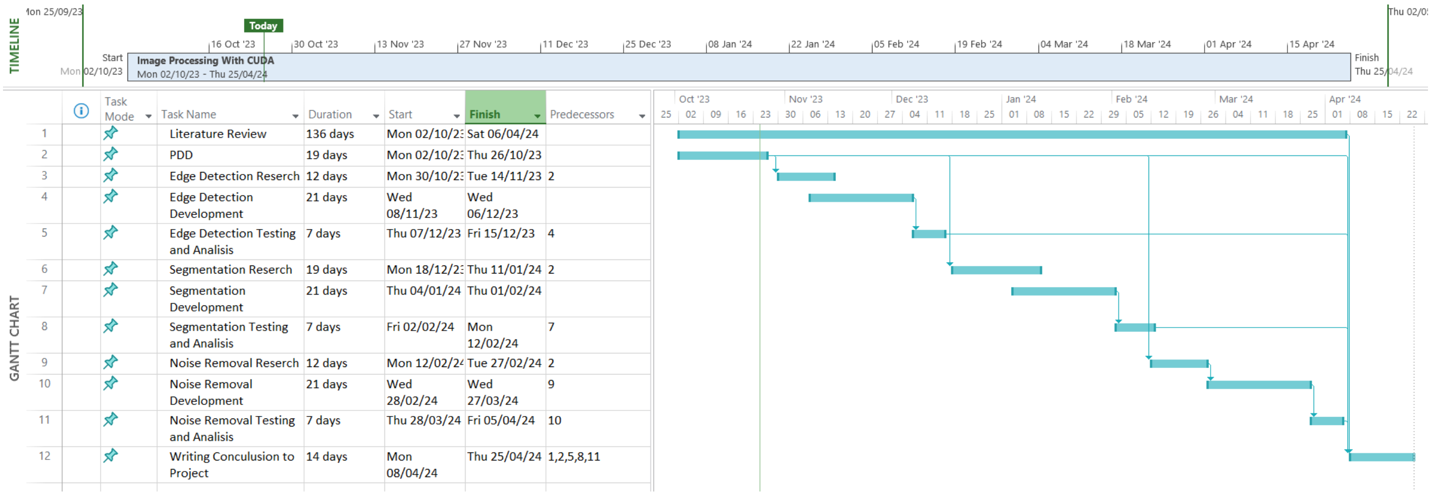
3. Project methodology and outcomes

3.1. Initial project plan

3.1.1. Tasks and milestones

- Literature Review
- Write introduction to the project document
- Edge Detection Algorithm
    - Research the Specific algorithm and specification
    - Implement Algorithm and test functionality
        - CPU
        - GPU
    - Analise performance and write findings
- Segmentation Algorithm
    - Research the Specific algorithm and specification
    - Implement Algorithm and test functionality
        - CPU
        - GPU
    - Analise performance and write findings
- Noise Suppression Algorithm
    - Research the Specific algorithm and specification
    - Implement Algorithm and test functionality
        - CPU
        - GPU
    - Analise performance and write findings
- Conclusion
    - Compare GPU performance between algorithms
    - Compare CPU performance between algorithms
    - Discuss differences in performance between image resolutions

3.1.2. Schedule Gantt chart



	ⓘ	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		📌	Literature Review	136 days	Mon 02/10/23	Sat 06/04/24	
2		📌	PDD	19 days	Mon 02/10/23	Thu 26/10/23	
3		📌	Edge Detection Reserch	12 days	Mon 30/10/23	Tue 14/11/23	2
4		📌	Edge Detection Development	21 days	Wed 08/11/23	Wed 06/12/23	
5		📌	Edge Detection Testing and Analisis	7 days	Thu 07/12/23	Fri 15/12/23	4
6		📌	Segmentation Reserch	19 days	Mon 18/12/23	Thu 11/01/24	2
7		📌	Segmentation Development	21 days	Thu 04/01/24	Thu 01/02/24	
8		📌	Segmentation Testing and Analisis	7 days	Fri 02/02/24	Mon 12/02/24	7
9		📌	Noise Removal Reserch	12 days	Mon 12/02/24	Tue 27/02/24	2
10		📌	Noise Removal Development	21 days	Wed 28/02/24	Wed 27/03/24	9
11		📌	Noise Removal Testing and Analisis	7 days	Thu 28/03/24	Fri 05/04/24	10
12		📌	Writing Conculusion to Project	14 days	Mon 08/04/24	Thu 25/04/24	1,2,5,8,11

3.2. Project control

As the project is run using AGILE, at the end of each sprint I'll make a small reflection on what I should focus on for the next sprint. As well as this I'll have the ability to make quick changes during each sprint.

3.3. Project evaluation

Using a dedicated image processing library or program I can check the functionality of the algorithms I implemented. For the data collected, I will cross reference it to the results of studies similar to mine.

4. References

Niblack, W. 1986, *An introduction to digital image processing,* Prentice-Hall International, Englewood Cliffs, N.J.

Solomon C. 2010, *Fundamentals of digital image processing: a practical approach with examples in Matlab,* Wiley-Blackwell.

Castaño-Díez, D., Moser, D., Schoenegger, A., Pruggnaller, S. & Frangakis, A.S. 2008, "Performance evaluation of image processing algorithms on the GPU", *Journal of structural biology,* vol. 164, no. 1, pp. 153-160.

Y. J. Zhang 1996, "A survey on evaluation methods for image segmentation", Pattern Recognition, vol. 29, issue 8, pp. 1335-1346.

Ziou, D. and Tabbone, S., 1998. Edge detection techniques-an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, *8*, pp.537-559.

Verma, R. and Ali, J., 2013. A comparative study of various types of image noise and efficient noise removal techniques. *International Journal of advanced research in computer science and software engineering*, *3*(10).

Abi-Chahla F. 2008, *Nvidia's CUDA: The End of the CPU*, https://www.tomshardware.com/reviews/nvidia-cuda-gpu,1954.html [Accessed 24/10/2023]