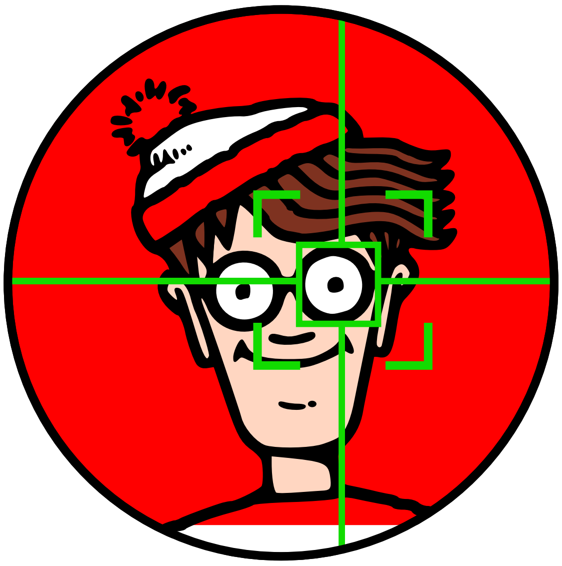
Project Preparation Report
Where's Wally?

Cian Booth

March 14, 2013



Contents

1 Introduction

Where's Wally is a series of puzzle books with a simple problem; you must find the eponymous Wally. He is wearing very distinct clothes; a pair of blue jeans, a red and white stripy jumper, with a woollen hat to match. Although he appears young, he carries a walking stick, and he wears glasses around his face. As such, viewed alone, he stands out as an entirely distinct. However, Wally appears alone only on the front cover of each book; within he is hidden behind a crowd of people and a plethora of objects. The human eye is a well honed tool; evolved over millions of years for the exact purpose of recognising objects visually. Even with such powerful tool, it can still take many minutes to find Wally.

Computers on the other hand, have not spent millions of years developing object recognition schemes. Indeed, the origins of computer vision are rooted merely 40 years ago. Computers excel in logical operations, and recognising an object is not an immediately logical operation. Many factors change what an object looks like; perspective, lighting, objects in the background and foreground, etc. To a person, it is generally still obvious that a given object is the same object when these things change. It is not as simple for a computer. This computer must put in a lot of processing time to be able to recognise an object. Problems like Where's Wally are not trivial for it to solve, and they take a lot of time.

Despite this, it is relatively easy to explain *how* a program might solve a Where's Wally puzzle. Furthermore, parallelising the process should also be similarly easy. These two factors make a parallel solution of Where's Wally very alluring prospects for HPC Outreach.

If the solution completes at approximately the same time that a human can solve the problem, then a meta-game could be produced. A given member of the public could race the program to find Wally. Meanwhile, the program would be visually displaying the techniques it was using to find Wally computationally, with emphasis on the HPC elements.

2 Previous Dissertation Reviews

When reading previous dissertations, an effort was made to read a range of papers to maximise their benefit. The first paper was strongly related to this project, *'High Performance Computer Vision'* by Benjamin Eagan[1]. This paper could potentially be a good way to learn about any hints or difficulties found within the topic of Computer Vision. The second paper was *'Parallelization and Optimization of a Tax and Benefits Model'* by Thomas McClintock[2]. This has no direct relation to Where's Wally, being a simulation of how taxes affect various groups of people. Thus, this paper could reveal general things that I should note in order to complete this project efficiently. These two papers were both on the list of dissertations which achieved a distinction. Therefore, a third paper was reviwed; *'Novel energy efficient compute architectures on the path to Exascale'* by Ioan Corneliu Hadade[3]. With this paper, differences between two standards of paper could be recognised. Acknowledging these differences would allow improvement of the final project report.

2.1 High Performance Computer Vision

Eagan's paper mostly deals with two topics; object tracking in real time videos, and disparity mapping. The first is useful for having more natural human-computer interactions. The second is for allowing computers to map 3D areas from 2D images. Although these are not the areas of computer vision that concern this project, it is still within the same branch of computer science.

The high quality of this paper was apparent throughout. The efficient use of English language made sure that the paper was continually understable. The figures included were engaging, allowing prompt understanding of the message being conveyed. The nature of the project that Eagan had chosen also strongly highlighted the importance of Risk Assessment. When choosing an algorithm to use, Eagan was very thorough in explaining why it was the optimum choice.

However, the paper had some issues. The introduction section felt quite long, and took quite a while to get around to basic message of the section. The explanation of the project goals was obfuscated by the lengthy introduction. Many figures were of processed images, but there were no figures of the original, making it hard to see the true effectiveness of the code. Some of the images showing errors in the processing lacked captions explaining what the cause of the problem was. Finally, some of the lines in the graphs included were very hard to distinguish from each other, making the plots somewhat indecipherable.

The paper also contained elements that had not been considered.

- The paper it written in something very close to default LaTeX.
- Within the introduction, were standard definitions of speed-up and efficiency.
- Includes a list of figures and tables
- Each section is concluded with a brief summary of what came before
- Eagan chose to use the HSV colour scheme over the RGB scheme.
- Benchmarking in computer vision is not standardised yet

2.2 Parallelization and Optimization of a Tax and Benefits Model

In this paper, McClintock solves the problem of parallelising a tax and benefits model for varying groups of people. This was done using dynamic programming of the optimal choice problem. This is very far from my own domain of work; McClintock is starting with serial code, and is optimising explicitly mathematical algorithms.

This paper was very well written, the start is engaging and the acknowledgements seem genuine. This improves the likelihood of the details of the paper being read. The problem was established well in the introduction, and equally the method used was explained well. Terms in the mathematical equations were explained, making the maths much more understandable. McClintock was honest about the crashes that happened in his program, and the results are clear. The graphs, in particular, are very easy to read, and well created.

There were some flaws with this paper though. Some of the tables presented were not captioned, and so could be confused. The choice of new pages was not always optimal either, making it somewhat difficult to read.

Things to consider;

- McClintock uses the inclusive 'we', instead of the passive voice when referring to work done.
- Results that are not immediately useful are located in the appendix.

- The performance gains of changing the number of threads running is given.
- Required to use `THREADPRIVATE` directive, which neither McClintock or his supervisor were familiar with.

2.3 Novel Energy Efficient Compute Architectures on the Path to Exascale

This paper focuses on the implementation of highly energy efficient super computers, with an eye to expanding towards the exascale era. Although Hadade originally intended to work on a heterogenous system consisting of an ARM chip and a GPU accelerator, timely procurement proved difficult. Instead, the paper focuses on the benchmarking of a Barcelona based supercomputer, Tibidabo, composed of Nvidia Tegra chips.

The introduction brings the paper in very well, it interleaves examples and requirements of an exascale system very well. There is a very clear explanation of why energy efficiency is so important for the development of exascale computers. The specifics of the Tibidabo system are explained thoroughly. Hadade goes on to extrapolate results to show that the supercomputer would scale onto the Green List. The benchmarks used seemed thorough, especially with respect to the use of demanding software like LQCD solvers.

However, there are many basic issues with the paper. To begin, the paper has not been spellchecked very well; there are several somewhat obvious typos included. Hadade also overuses the word 'Primordial' at the start, which is unusual enough to inhibit easy reading. The main bulk of the work is a dry list of hardware architechtures, and is somewhat hard to read through. The amount of results compared to the amount of topics discussed makes the results seem somewhat sparse. The conclusion is especially short, about a paragraph long.

Considerations:

- Hadade had to change project goals significantly
- References and citations are not in chronological order

2.4 Conclusions Drawn

Upon reading these projects, the components of a successful project became clearer. Of foremost importance is risk analysis. Eagan found himself somewhat overworked with his project, and Hadade had to completely change project goals. This makes it apparent that being prepared for risks is extremely important when a project goes awry. Unfamiliar techniques might have to be used. As using machine learning methods is being considered, this consideration gains relevance.

The layout of the project also needs some consideration. Explaining the layout of the thesis was not an element that was considered. Neither was including a list of figures and tables. If these are standard elements of a project, then they should also be included in this one. The difference between Eagan and McClintocks graphs are also very important. The clarity of McClintocks graphs make understanding the results extremely simple, much improving the ease of reading. Finally, control over LaTeXand the placement of new pages and diagrams should be exercised strongly.

3 Work Plan

According to the MSc. Programme Handbook[4], the start of the dissertation period is on the 3rd of June 2013. The dissertation must be submitted by the 23rd of August 2013. This means that there are 88 days to start and complete the project.

This project has two main strands of development: the linear code and the parallel code. This division is somewhat coarse, however, so each region will be divided into three new sections.

The workplan is broken down below. Note that each section has built in time for weekends, plus some stretch for running overtime. Furthermore, the text inside the square brackets (e.g. [L1]) indicates the path on the Gantt chart. This can be found in the appendix.

Linear (~30 days)

Designing [L1] (4 days) This will consist of deciding the format of the linear code. This includes choosing a programming language and library to use. Basic function concepts will also be developed here. It is hoped that 4 days would be enough to design this solution.

Testing [L2] (6 days) This project will be test driven. As such, creating tests comes before any implementation stages. This time will be spent devising and implementing tests to ensure that the program works correctly. The problem is conceptually simple and it is hoped that the tests will reflect this. Six days have been assigned to creating the tests.

Implementation [L3] (14 days) Working from the basis of the tests, code will be written that satisfies the tests. Once this has happened, a working solution for the Where's Wally puzzle will have been created. Fourteen working days have been given to implementing the solution.

Optimisation [L4] (any remaining time) If the linear and parallel programs have both been produced, then spare time will be used optimising both solvers. This is important, as it is the main way to benchmark any speedup that the parallel code exhibits. Optimising the linear program will likely involve reducing memory access, and so time will be focused on this.

Parallel (~40 days)

Designing [P1] (4 days) Although a program utilising parallel methods is more complicated than a linear one, this project intends to expand on the linear solution. This means that functions from the linear program will be analysed for potential parallelism. Four days have been put aside for this stage.

Testing [P2] (6 days) Here, new tests will have to be written to ensure that the parallelised functions designed above work correctly. The old tests should hold true however, so only a small amount of testing needs to be done. Creating specific tests for the parallel version should 6 days.

Implementation [P3] (21 days) The implementation of the parallel program consists of satisfying the tests laid out during the testing period. It is intended that this will mostly be through the modification of existing code. As parallel programming is more difficult than linear program, implementing a parallel solution has been given 3 weeks.

Optimisation [P4] (any remaining time) Again, any time remaining at the end of the project will be devoted to optimising the solver. The goal is to have the parallel run as fast, if not faster, than a human solving the same puzzle.

Dissertation Report (ongoing)

Writing Report [D1] (ongoing) The writing of the report can be begin as soon as any amount of design has been done. The report will be updated as work continues, ensuring that the report is never a large workload. This has the added benefit of allowing more time for editing and correction. During this time, the project presentation will also be written.

Present Dissertation [D2] (2 days) Over two days, the dissertation presentations will be held. Time between the submission date and the presentation of this report will be spent practicing the presentation.

4 Risk Analysis

Listed here are some of the regular problems, followed by risks specifically related to this project.

4.1 Common Risks

Slow Pace (Probability: Possible, Impact: Severe)
This happens when the target time for a project is underestimated, or when the project hits an unexpected snag. Firstly, slack time has been built into each section of the work plan, as seen in the work plan section. If the slow pace becomes critical, a simple version of the linear code will be used to produce a simple parallel version.

Fast Pace (Unlikely, Negligible)
In the unlikely situation work proceeds faster than expected, there are several areas for continued work. Firstly, optimisation of pre-existing programs, both linear and parallel, would be extremely useful. This will give definitive results about how much speedup the program can obtain when recognising Wally in parallel. Secondly, a user interface that shows members of the public how the program works could be developed. This will start to create a project suitable for HPC Outreach.

In Absentia (Possible, Moderate)
Often in projects, a supervisor or student will be unable to attend meetings. A netbook is intended to be used regularly, enabling work to be done remotely. This extends to meetings, which can be done online using emails or other virtual techniques.

Broken Equipment (Unlikley, Severe)
If the normal workstation, a netbook, happens to break, then work can be continued on a desktop. This should be possible with the use of a remote versioning system, like github.

Sickness (Possible, Moderate/Severe)
Illness cannot be prevented, but in general, some amount of work should be able to be done. In the case of a major illness, the solution for slow pace will have to be implemented.

Ennui (Certain, Moderate)
Writers block will be solved by writing the dissertation report, or by implementing other sections of code. In the case of serious disillusionment, basic willpower will have to be exercised.

4.2 Risks Specific to a Where's Wally Solver

Unable to Find Wally (Possible, Catastrophic)
In the situation that my program cannot find Wally, the problem will have to be simplified. In testing various libraries, code has already been produced that can recognise a specific icon in an array of icons. Thus, in the simplest of cases, some code can work, and can then be parallelised.

Not Allowed To Use Wally (Unlikely, Catastrophic)
It is conceivable that Walkers Books, publishers of Where's Wally would not allow me to use their Wally figure. Although this is legally dubious, there are ways around copyright laws. For example, it is unlikely they have copyrighted the concept of Where's Wally. This means a Where's Wally type puzzle could be generated. This could be done by randomly placing a large stock of images on a canvas.

Not Enough Wally Samples (Probable, Moderate)
This risk especially applies to any machine learning that might be done. The solution is to generate new puzzles, as with the above risk.

Not Fast Enough (Probable, Negligible)
If the program is not fast enough to compare to the speed that humans can solve a puzzle, then optimisations should be attempted. If this can't be done in a sufficient time frame, then the completion of a parallel solution will have to be the final stage of the project.

Not Suitable For HPC Outreach (Probable, Negligible)
This is very likely, and in this case, my project will have to be the starting point for another HPC Outreach project.

A Work Plan

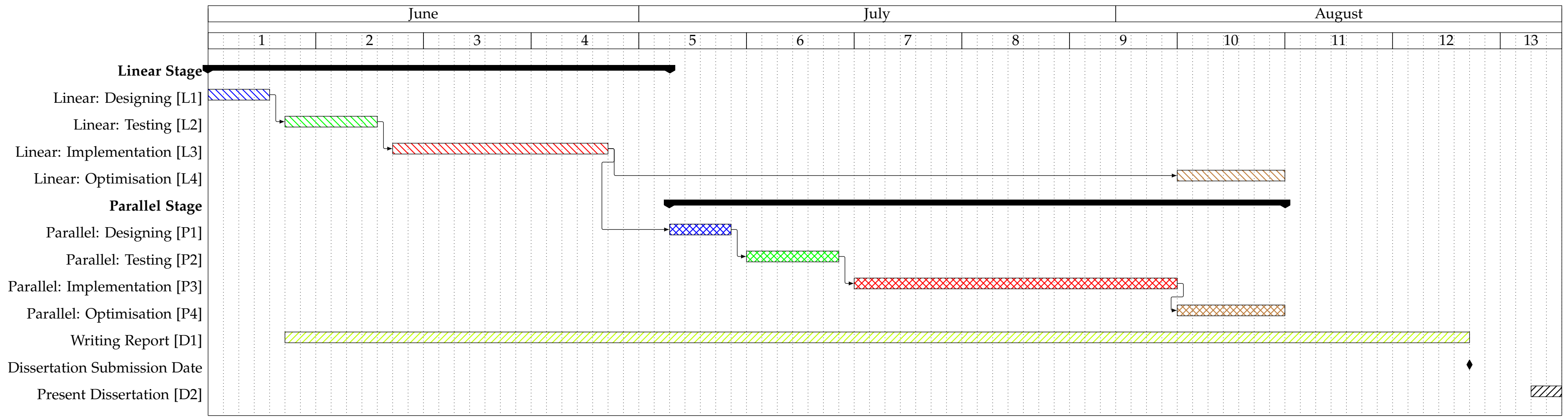


Figure 1: A Gantt Chart of how time will be divided between areas of work

References

[1] B. Eagan, “High performance computer vision,” Master’s thesis, School of Physics And Astronomy, 2012. http://www.epcc.ed.ac.uk/wp-content/uploads/2013/02/Submission-1138832.pdf.

[2] T. McClintock, “Parallelization and optimization of a tax and benefits model,” Master’s thesis, School of Physics And Astronomy, 2012. http://www.epcc.ed.ac.uk/wp-content/uploads/2013/02/Submission-1150361.pdf.

[3] I. C. Hadade, “Novel energy efficient compute architectures on the path to exascale,” Master’s thesis, School of Physics And Astronomy, 2012. http://www.epcc.ed.ac.uk/wp-content/uploads/2013/02/Submission-1146632.pdf.

[4] “Msc and diploma in high performance computing student handbook,” 2012. https://www.wiki.ed.ac.uk/download/attachments/136519733/Programme-Handbook-2012+-+13+-+Final.pdf?version=1.