FIT3161 Computer Science Project 1
Semester 2, 2019
Monash University

PROJECT PROPOSAL

on GPU Acceleration of Line-integral Convolution using OpenCL

Delivered by
Team 01

Bryan Mayson
Kelvin Butler
Anh Huong Nguyen

WORD COUNT: 6282

TABLE OF CONTENT

1.0 INTRODUCTION

The objective of the project this proposal pertains to is to use Graphics Processing Units (GPU's) in order to accelerate the computation of certain raster filters utilised by cartographers in map making. Eduard is a software tool, created by Monash FIT and ETH Zurich, for creating shaded relief images for cartographers, from elevation maps and for this project we will discuss on how we will optimise the computational process of one specific filter through the use of parallelism and OpenCL. This proposal would mainly discuss on how we will achieve a software product that consists of a modified version of the existing algorithm provided to us with the capability of producing linear convoluted relief images of large elevation maps within optimal time. The proposal will also elaborate on how we will be using OpenCL, a cross platform specification designed to run parallelised code across multiple "compute units" and has been widely commercialised across large technological industries such as Intel, Nvidia and AMD, to achieve said parallelism.

In addition, the proposal below will go into details about our management, product development and methodology, how the team gathers the resources, as well as how we schedule to put the product online for future uses.

The team has elected to accelerate the line integral convolution algorithm as the raster filter to be applied for our elevation maps. This algorithm takes an average of a given point based on the slope at that point and the points around it. As this algorithm requires expensive computations for each points on the map, it is a perfect candidate for parallelization.

Over the course of the Semester 1 of 2020 the team will go about developing and testing the software summarized by this document, the method by which that will be done are outlined later in this document.

2.0 LITERATURE REVIEW

Monash FIT and ETH Zurich have made a collaboration on developing an application which aids cartographers for the creation of shade relief images containing the elevation models of a section of the world. The algorithm currently runs the generation of these shaded relief images within linear time complexity and obtains nonoptimal computational time results when faced with elevation models of enormous size.The objective of our project would be to develop a software product to resolve this issue of nonoptimal processing speed.

Arriving at the end result of the software product may be achieved through various methods however, before elaborate on the steps to be taken in developing the software product, we would briefly look into key concepts to understand what the product should achieve.

2.1 Understanding Parallelism

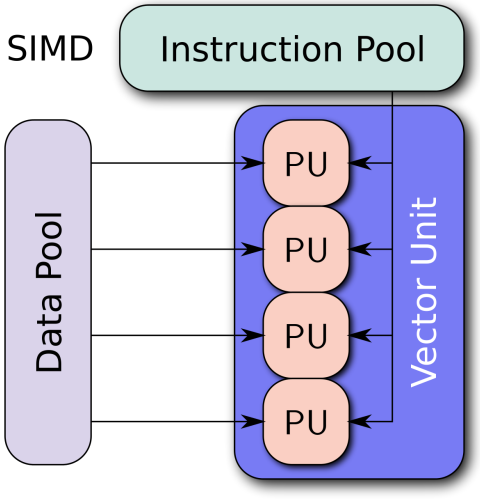
The concept of parallelism revolves around the means of displacing the computational workload of a linear computation to be computed as subprocesses across the multiple processors available within a device's CPUs/GPUs.Its purpose is to accelerate any linear process which fulfills

the Bernstein's Condition for Parallelism (Bernstein, 1996) such that the process would be computed within a much shorter time than if it were to be computed in linearly.

Within a linear algorithm, calculations and processes are then performed sequentially up to the termination of the program. For optimal programs with a small amount of computations or process to be performed the program performs relatively efficient in terms of the time taken in achieving its results.However, if the program were to compute a large problem such as generating a Mandelbrot set image, the time taken for these programs would not achieve optimality than that of a parallel program. This is due to a linear program having to process each computational heavy iteration sequentially which strains the single processor compiling the linear program. In addition, the linear algorithm would only perform computations on a single processor while even though current computers have more multiple processors available.

Parallelism provides the developers the capability of distributing each of these computational heavy iterations of the linear algorithm amongst the available processors the computer has. It allows each small portion of the heavy iterations to be computed simultaneously and compiles them together before termination to produce the same results of the linear algorithm with the main difference being that each heavy iteration would not be dependent on the sequence it is in.

The Single Instruction Multiple Data (SIMD) Model for data parallelism represents the parallel architecture found commonly amongst current parallel applications. Cypher states that the SIMD parallel architecture elaborates on a single instruction being forcefully computed by each of the available processing units within the system simulations (Cypher and Sanz, 1994). For our software product, we would be using this model for the development of our parallel program. A visualisation of this model can be seen below.



Bernstein's Condition for Parallelism consists of three main requisites to act as a guideline in determining whether a linear program is able to be implemented in parallel. The equations of these conditions stated by Bernstein are as follows,

$$I_0 \cap O_1 = \Phi \sim anti\ dependency$$
$$I_1 \cap O_0 = \Phi \sim flow\ dependency$$
$$I_0 \cap O_1 = \Phi \sim output\ dependency$$

where I and O represent the inputs and outputs of their respective processes respectively. If the linear program is able to fulfill these conditions, then the program is deemed to be parallelable.

However, even if a program can be parallelized its does not necessarily mean that parrelization should be performed. The efficiency of a parallel program does not solely depend on the optimality of the workload distribution, yet efficiency is also dependent on the hardware specifications of the computer is being run on.These specifications may consist of the amount of memory available in the system, the clock speed of the current CPU, or even the number of available process.

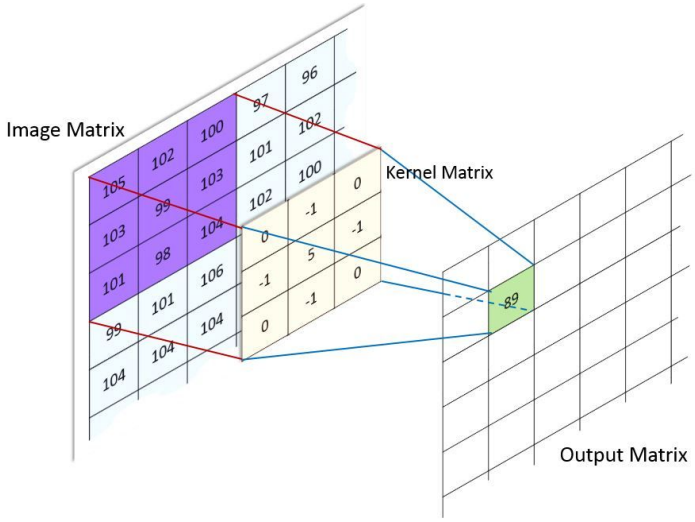
As such, parallelism has been a popular concept amongst modern industries in the computation in mathematical related areas such as medical imaging,image processing, cryptography, cryptanalysis and many more.

2.2 Image Convolution

The term image convolution is described by Ludwig as the general purpose filter effect for images which modifies the spatial frequency within an image to change its characteristics (Ludwig, n.d). This process involves applying a convolution kernel onto an image to produce
the filtered result of the image. A convolution kernel is typically a matrix with a smaller height and width compared to the original image, with each kernel being used for specific filtering effects such as sharpening blurring, edge detection and more. An example of a kernel matrix can be seen as below.

$$Kernel = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

The convolution process involves obtain the image values located by a pixeles of matrix size of the image and applying noises to it to obtain the filter image. The convolution kernel would act as the top layer of the image matrix which acts as the filter. Each image pixel within the symmetric range of the kernel matrix would then undergo the convolution calculation process.This process involves having each of the pixel multiplied with its corresponding kernel matrix values and compiled together to generate the pixel of the output located by the center position of the kernel matrix.The visualisation of this process can be seen in the diagram below.



However, the main issue regarding this process would be the calculations of the border values of the output matrix. This is as some values of the kernel would stand out of the image matrix as there are no values within the image matrix to be applied with the values of the kernel matrix. To resolve this issue we would have to insert ghost values within the image matrix to allow the convolution process of the border pixel to be applied. The convolution process would then be applied throughout the entire pixels within the image size to obtain filtered image.

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

210	89	111			

Output Matrix

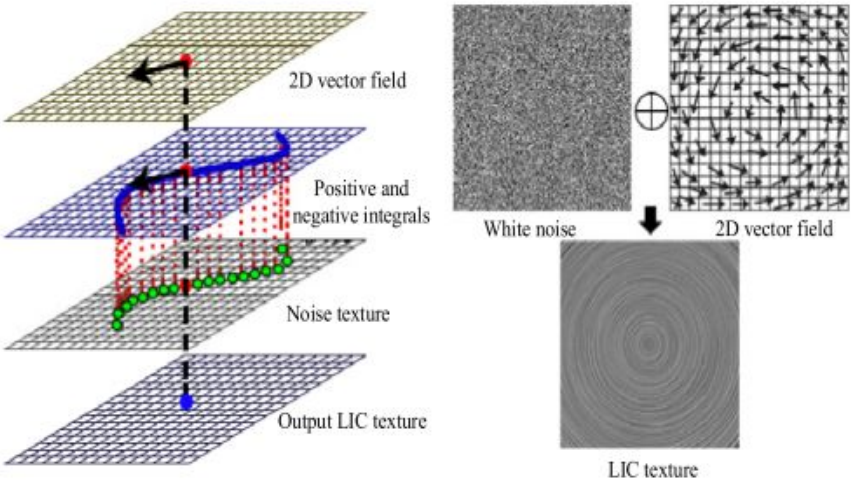
$$0*0+105*-1+102*0$$
$$+0*-1+103*5+99*-1$$
$$+0*0+101*-1+98*0 \quad = 210$$

For this proposal, we have decided to mainly focus on obtaining the acceleration of applying line integral convolution filter onto the elevations map obtained.

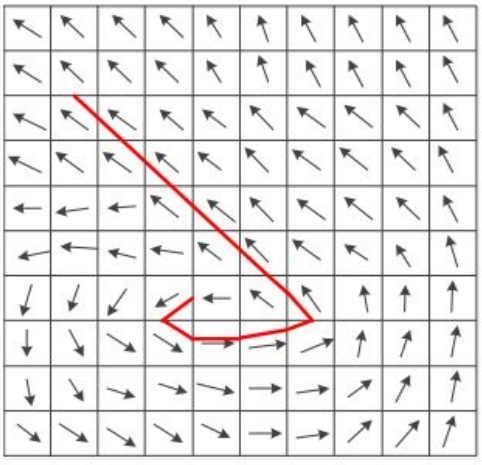
2.2.1 Line Integral Convolution

Line Integral Convolution is a form of image convolution which allows for the visualisation of the direction of vectors within a vector field. It is commonly used to enable its users to observe fluid motions such wind movements in the tornado or even the wave patterns of the ocean.The principle of the process is to generate streamlines and convulate the corresponding vectors onto the streamlines to obtain corresponding value of the filtered image (Qin et al, 2019). The range of the stream line would also vary depending on the extent of the desired filtered effect through the

convolution process. The diagram to represent and visualise this process and results can be from below.



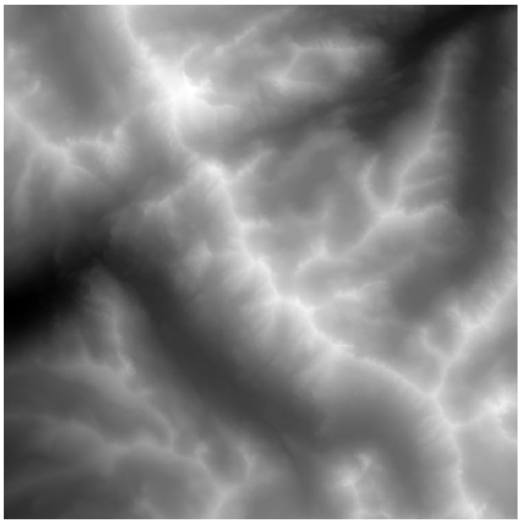
The convolution process of line integral convolution begins with first identifying the location of the pixel within the 2D vector field to be convoluted to form the desired filtered output. It then follows on by obtaining the streamline obtainable from the data of the vector field which involves this location.The streamline would be the data values of each pixel within the range of the specified streamline length, with the same directional vector of the one selected. This simply implies that if the streamline length were to be stated to have a value of 10, the selected point within the 2D vector field would then explore 10 pixels upwards and downwards of pixels which contain the same vector direction. These values would then be applied to a normal distribution and proceed to be compiled. Once the values are compiled, the mean result of these values would represent the filtered output of the pixel of the image located by the specified position of the 2D vector field. The diagram to observe how the streamline could be visualized as, can be seen as below.



Implementing this process would be possible for images which contain vector values to be computed however, with regards to normal images, the convolution process would be considered to be impossible to be applied without the addition of a vector field layer on to the normal image. For our software product, we are not required to be concerned about the implementation of a vector field layer onto the elevation maps, this is due to the fact that the elevation maps provided already contains vector values which represents the vector field thus allowing us to perform said convolution.

2.2.2 Elevation Maps

Elevation maps represents a geographical section of the planet which contains values such as longitude and latitude.These information can be stored within an ASC file and can be used to visualize the peak and depth of the geographical location through using applications such as QGIS. The ASC file otherwise known as the Action Script Communication Files are files written in scripts to be used for a specific application or program (ReviverSoft, n.d). The desired contents to be viewed by the file in only accessible to those applications which supports the scripts written within the ASC file.In this case, the visualisation of the elevation model are only visible through applications such as QGIS. A sample ASC file representing and elevation map can be seen as below. The brighter regions of the map represents the peak of the model while the darker regions represents the depth.



2.2.3 Applying the convolution to the Maps

As stated previously, convolution of the elevation maps is possible due to the value within the map's ASC file representing the 2D vector matrix. However, the process of obtaining the convoluted image of the entire elevation map would require a significant amount of computational time. Not to mention the convolution process would have to consider the addition of ghost values in a situation where we have to process convolution with no existent values within the matrix as stated earlier in 1.2.1. As a result, the computational time required for the generation of the convoluted elevation map would scale linearly to the size of the elevation map to be processed. This would then result in longer computational time for elevation maps of a significantly large size. Currently the linear algorithm's performance is entirely equal to what is said above, thus making it unideal for cartographers to use.As such this is the source of our objective to accelerate this convolution process.

Through careful observation and analysis, we were able to identify that the convolution process of each pixel within the image is not is not dependent on the result of the convolution of the previous and following pixels vice versa. As such, we are able to conclude that applying linear integral convolution is parallelizable as it has fulfilled Bernstein's Conditions for Parallelism.

2.3 OpenCL

OpenCL is a standard programming interface developed by the Kronos Group which is used used to enable developers to accelerate their task parallel or data parallel computations easily within a heterogeneous computing computing environment (Stone,Gohara & Shi,2010). The interface also handles the supported programming languages and application programming interfaces (API) executable by the computing devices within the computer. The computer devices may either be the central processing unit of the device (CPU) or even the graphical processing units (GPU) of the device.

The programming language used to allow for processes to be computed by OpenCL is similar in nature to the C programming environment.The term used to describe a function compiled within the OpenCL environment is known as a "kernel".The programs within the OpenCL environment is also designed to be compiled at run time such that it would be able to be applicable to various host devices.

Currently OpenCL is considered to be one of the most common parallelism frameworks used within technological industry due to its applicability amongst the various of the hardware found within the computers of today. The Kronos group have developed multiple open source support libraries which allows the OpenCL environment to run on well known GPUs such as ,Intel Nvidia and AMD Radeon.

2.4 Existing Research

Numerous research regarding the acceleration of an image convolution process through the use of OpenCL can be commonly found throughout the web due to OpenCL's wide commercial usages. An example of such research containing similar objectives with our current project would be "Exploiting SIMD Extensions for Linear Image Processing with OpenCL"(Antao & Sousa ,2010).This research written by Antao and Sousa elaborates on how they were able to use various implementations of OpenCL to accelerate various reference convolution algorithms through the use of Intel and AMD processors. These various reference algorithms vary from each other through being vectored and non vectorized. As a result, they were able to observe that the higher performance could be observed through efficient data organization. They were also able to compare the efficiency between the various algorithms and witnessed that the non vectorized algorithm computes three times more efficiently compared to the vectorized algorithm.Comparison between the usage of an AMD processor and Intel processor was able to obtain, with the observation that Intel processors being much more optimal in performance.

The main difference between this existing research and our current project would be that our project would be mainly focused on optimizing the vector based linear integral convolution algorithm rather than comparing the results between varying algorithms.The current research also lacked the discussion of optimal partitioning schemes however, we were also able to take into considerations the results gathered by this research such as Intel processors producing far better results compared to that of the AMD processor, to optimize the implementation of the software product. As such, future research and hands on experience would be required to obtain the knowledge of an optimal partitioning scheme of the convolution process.

2.5 Conclusion

In summary, this literature review elaborates on key concepts to inform the readers such that they are capable of understanding this propsal's objective in obtaining the parallelization of the linear integral convolution process of an elevation map through the use of OpenCL.

3.0 PROJECT MANAGEMENT PLAN

3.1 Project Overview

As aforementioned, our objective is to further aid the accessibility of the Eduard application. The product is envisioned to use the GPU language, OpenCL in particular, to accelerate the filters used. With OpenCL, the team's goal is to provide a product that provides much better speed up than that of the current Eduard application, within our proposed budget and timeframe.

The project is assumed to be complete within the next 6 months, with the expectation of further supervisions and revisions to improve the product gradually.

3.2 Project Scope

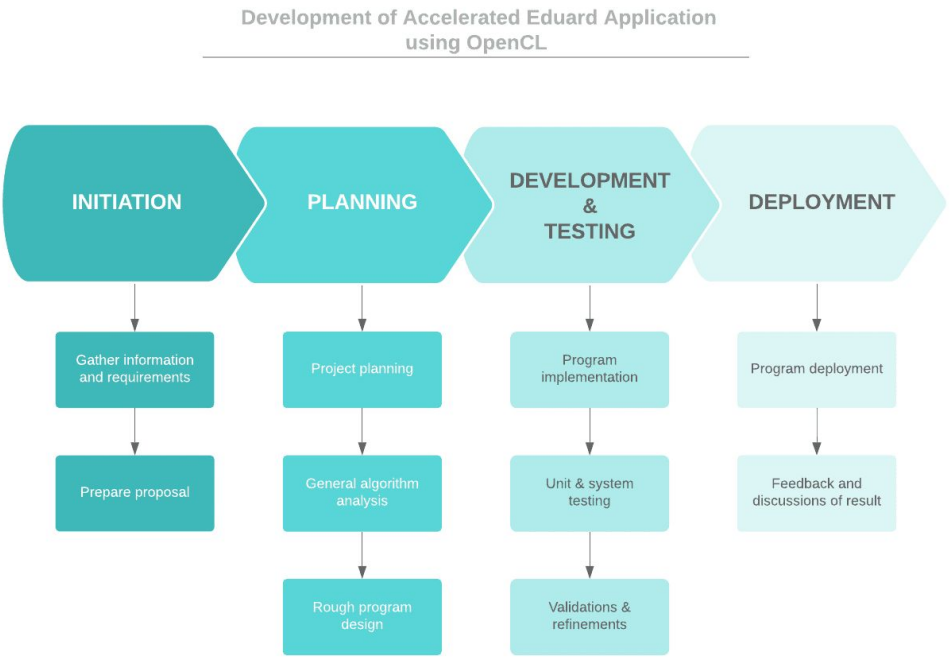
Project Scope Statement

Project Title: GPU Acceleration of Raster Filters using OpenCL **Prepared by:** Team 01 of FIT3161 Computer Science Project 1 at Monash University
Project Justification: The aim is to further accelerate the current Eduard application by developing it utilizing OpenCL and the concept of parallel computing. Our hope is to encourage more users by improving the application's performance, as well as proving that parallelism contributes greatly to the modern technology. Any information and development achieved in this project is expected to be used in the future.
Product Characteristics and Requirements: In Scope: 1. Accessible from Java Virtual Machine 2. Improve current speed up of raster filters in Eduard using GPU 3. Utilize line-integral convolution filter Out of Scope: Onboarding material for maintenance
Summary of Project Deliverables **Project management-related deliverables:** scope statement, WBS, schedule, status reports, final project presentation, and final project report **Product-related deliverables:** 1. Fully-functioning program 2. Software documentation
Project Success Criteria: - Improve the performance of Eduard application

- Encourage more users of the application
- No environmental/functional issue
- Product is delivered within timeline (6 months)
- Receive good/non-negative publicity

3.3 PROJECT ORGANISATION

3.3.1 Process Model



The team has opted for waterfall approach for our software development cycle, based on the assumption that the requirements for this project will not be altered under any circumstances, as well as considering that this is a short-term small-scale project with clear objectives and technology. With this approach, the team wants to simplify the actions undertaken during each stage, and makes it certain that we focus entirely on our objective. This approach also prevents loss of data and information between phases, because we need to complete one to move on to another. Moreover, the project does not exactly require client's involvement, since this is already a published application and our sole purpose is to improve its performance, thereby only calling for the development team during testing phase before product deployment. The waterfall method, in conclusion, is straightforward, and suits the team's requirement perfectly.

As mentioned above, the project is divided into two phases, each consisting of two stages. To initiate the project, we have gathered the necessary information and requirements through adequate research based on the brief given. Following up, we prepare a project proposal that clearly states how we understand the project, as well as what the team's intentions are to deliver it successfully. Our team will continue to move on to project planning, while simultaneously analyze the algorithm that

was assigned to us to prepare for the next important phase. A rough design will also be visualized towards the end of the Planning stage.

The second phase might be considered the success determinant of our project, considering that the team will start developing the ideas stated in this proposal. Further testing and validations, as well as peer-review comments from our supervisors will definitely be required, so we can refine the program to satisfy the project requirements. Consequentially will be the deployment stage, when we release the program, and collect feedback for further discussion.

3.3.2 Responsibility Allocation

Each member of the team is allocated to certain responsibilities to ensure project success. More details are given using the RACI matrix below. By utilizing this matrix, the team ensures that no team member is overburdened with several tasks and that no task is unallocated, as well as providing the supervisors a quick glance towards the team's organization.

Notes: R - Responsible / A - Accountable / C - Consulted / I - Informed

Project Tasks or Deliverables	**Project Manager**	**Developer**	**Quality Assurance**
Initiation Stage			
Project Kickoff Meeting	R	I	I
Information and Requirements Gathering	R / A	R / C	I
Requirement Documentation	R / A	C / I	R
Project Proposal	A	C / I	R
Project Proposal Presentation	R / A	C / I	R
Planning Stage			
Finalize Process Model	C / I	I	R / A
Risk Identification and Management	C / I	I	R / A
Finalize Schedule	C / I	I	R / A
General Design Walkthrough	C / A	R	I
Development & Testing Stage			
Start Implementation / Build	C / I	R / A	C / I

Deliverables			
Create Test Cases	I	R / C	R
Execute Test Cases	I	R / C	R
Request Feedback from Supervisors	R	I	C / I
Create Status Report	C / I	I	R
Create Change Requests	C / I	R	C / I
Perform Change Requests	R / C	R	I
Deployment Stage			
Deploy Program	A / C	R	I
Create Project Closure Report	R	C / I	C / I

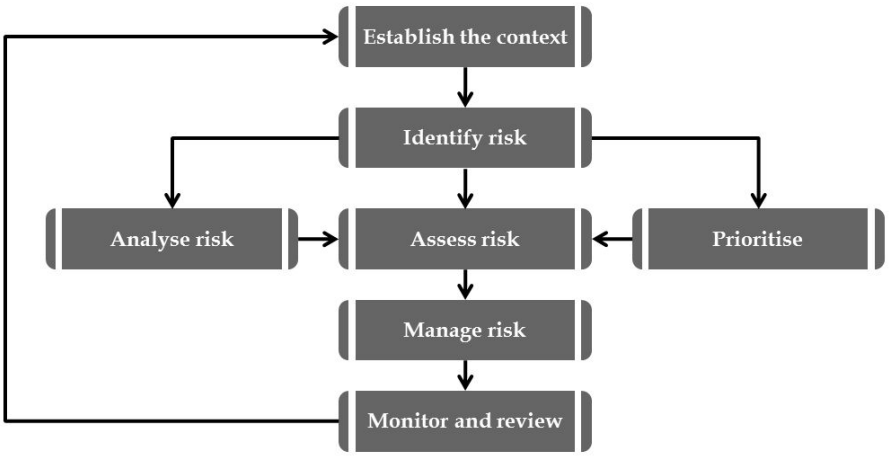
3.4 MANAGEMENT PROCESS

3.4.1 Risk Management

Purpose of Risk Management Plan

Risk has many definitions, but it is implicitly stated to consist of two characteristics: uncertainty (an event may or may not occur) and loss (an event has undesired effects) (Wallmuller, 2002). It is a potential issue that may directly compromise the success of the project. To secure successful delivery, it is essential that we identify such imposing risks, as well as a clear plan to cope with them. This is the main purpose of a risk management plan.

Risk Management Procedure



There are many risks posed to an IT project, including technical and project-wise. Throughout the process of the project, the project manager will need to continuously identify, analyze and mitigate risks. It is also vital that team members actively communicate and give feedback about risks mitigation methods to ensure a successful project.

Risk Identification

Risks can be identified either by brainstorming or using risks checklist. Our team has decided to use the former for our risk identification process. Brainstorming possible issues help increase the team's creativity, as well as raise awareness of certain risks, and help the team come up with mitigation plans more effectively. Since our team only has three members, there will hardly be any organizational risk involved. On the other hand, this is a heavily technical project, from which several functional and software issues might stem, and critically affect the delivery of the project.

Risk Analysis

After identifying the risks, the team will proceed to analyze such risks to measure the level of impact those risks imposed on the whole project. Through risk analysis, we will be able to determine which risks have higher impact than the others, and thereby requiring further attention.

Qualitative Analysis

Qualitative analysis is done to mark the risks that are of more importance. In the risk register, risks significance will be illustrated according to a scale of "Low - Moderate - High" measurement, which represents their impacts and probability of happening.

Impact:

Low - Has little impact on the project and product delivery.
Moderate - Has average impact on the project and product delivery
High - Has critical and important impact on the project and requires immediate attention.

Probability of occurrence:

Low - Less than 30% chance
Moderate - About 30% to 70% chance
High - More than 70% of happening and requires immediate attention

Quantitative Analysis

Quantitative analysis follows after qualitative analysis, and only applies for risks that are marked necessary for such analysis. Effect and cost, schedule contingency, etc. will be applied to those risks, along with a numerical rating.

Risk Response

There are four most common strategies in risk response:

- Avoidance: Reduce the risk's possibility to zero by focusing on eliminating the cause.
- Transfer: Transfer the risk's responsibility to another party
- Mitigation: To reduce the risk's impact since it is unavoidable.
- Acceptance: Accept the risk's impact and deal with the consequences.

The team's objective is to respond to risks mostly by avoiding or mitigating them. Further insights will be given in a specific risk register custom-tailored for our project, referred to in Appendix.

3.4.2 Monitoring and Controlling Mechanism

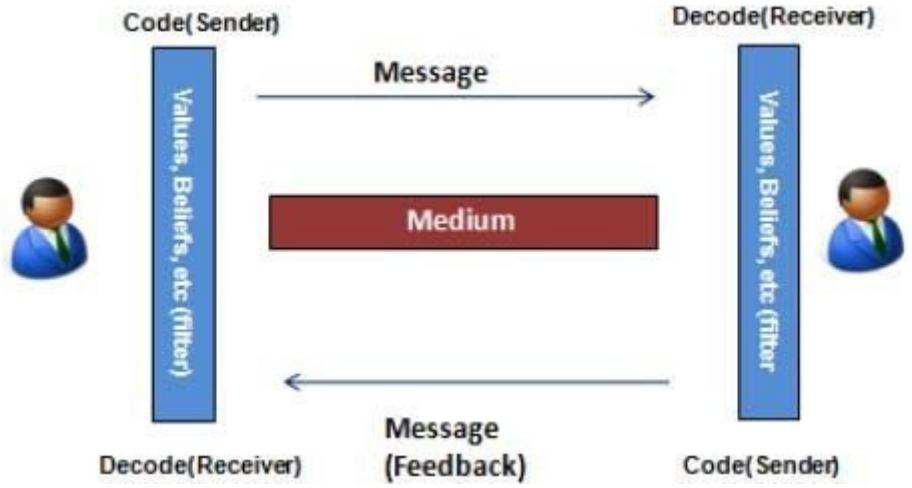
According to the PMBOK Guide (PMI, 2008), the three basic steps in monitoring and controlling process include:

- Track, review, and regulate the progress and performance of the project
- Identify any areas in which changes to the plan are required
- Initiate the corresponding changes

Despite detailed planning, changes during phases are inevitable for any project. This has raised the need for Change Requests (CR). Most of the time CRs would lead to a change of scope, but in our case, the changes we expect would mainly be related to how we utilize the provided algorithm rather than the project itself. Since this is a small-scale project with limited panel members, we do not expect a lot of CRs, but if there is, it would help the Project Manager to monitor the development more efficiently. CRs would also help with our version controls of the program, in addition to GitHub sharing.

3.4.3 Communication and Reporting Procedure

Communication process requires a sender and a receiver. The sender "encodes", or creates a message, which is meant for the receiver. Meanwhile, the receiver "decodes", or interprets the message, and send feedback back to the sender. An effective communication process means that every message sent will have a feedback reflected, indicating that the message has been received. Such message and feedback can be positive, negative, or neutral, depending on the values, emotions, and beliefs of the senders and receivers. The communication process always requires a medium, e.g. meetings, emails, presentations, etc., and settings such as time and location, which has immediate effects on the results.



There are several types of communication and reporting within the team and to the supervisors of this project. To encourage better communication and more effective project management, we have set up several meetings, as well as regular check-ins among members, ensuring that the project is still going according to schedule. Since we are a small group, the main communication and reporting mechanism used has been informal methods such as messages, social media, and informal meetings. Below is a communication matrix to demonstrate our mechanism and plan in more details.

	Purpose	Medium	Frequency	Audience
Kickoff meeting	- Introduce project. - Confirm objectives, goals, and deliverables needed.	- In-person meeting on campus	- Once at start of the project	- Project team (PM, QA, and Developer) - Unit supervisor
Project team meetings	- Review status of project.	- In-person meeting on campus	- Every Monday at 11AM starting from Sep 16	- Project team
Meeting minutes	- General summary of each meeting	- Written minute	- After every project team meeting	- Project team
Check-ins/ meeting recap	- Update project status among project members	- Social media group	- Every Friday starting from Sep 20	- Project team
Project status meetings	- Update and report to supervisors on current progress	- In-person meeting	- Once on Oct 8	- Project team
Technical demonstration meeting	- Developer demonstrates further details of the program	- In-person meeting	- Once on Monday 11AM Oct 14	- Project team
Status report	- Report current program performance against performance mesurement baseline	- Written report	- Once every week starting semester 1, 2020	- Developer (main reporter) - PM & QA (informed)
Progress report	- Update current progress of the project	- Written report	- Once every week starting semester 1, 2020	- QA (main reporter) - PM (informed)

3.4.4 Review and Audit Mechanism

The Review and Audit Mechanism for our project can be alternatively interpreted as the Software Quality Assurance (SQA) process, which is essential to evaluate whether the current software process is following the predefined standard process. The main benefits of conducting SQA are that it helps monitor and control the process, ensure that procedures are followed correctly, as well as avoiding certain quality issues.

The initial part of any SQA process would be developing a detailed SQA plan, which will indicate certain testing level and schema used. It is vital that the QA is involved from the initiation of the project, such as gathering information and requirement, to keep the design away from bug prone areas, which might compromise the final product. To facilitate the review and audit process, proper documentation is also required.

The team will use GitHub as the main version control method. However, by conducting SQA during implementation and testing phases, it would also help with maintenance and improvement of the final product.

3.5 Schedule and Resource Requirements

3.5.1 Schedule

Referring to the figure presented in the Process Model section, the project will be divided into four clear stages of initiation, planning, development & testing, and deployment. The first two stages are expected to be completed during Semester 2 of 2019, while the rest will be carried out during the course of Semester 1 of 2020. In the appendix are the Gantt charts prepared to summarize what the team has planned so far, as well as how we intend to carry out the second phase of this project.

There are two critical milestones for this project. The first one concludes the first two stages of Initiation and Planning at the end of Week 12 of Semester 2, 2019. The team must complete the proposal and be able to clearly present the ideas stated in said proposal to the supervisors to gain approval, so the project can carry on to the latter phase.

The second pivotal milestone is at the end of Week 10 of Semester 1, 2020, when the team is scheduled to complete the program development, during which phase, we will also begin entering the testing and validation starting from Week 4. Since this has direct impacts on the final delivery, any necessary changes will have to be reported promptly to prevent any undesirable delay. The team wants to emphasize more on the testing phase, as well as receiving feedback simultaneously, so we can refine the program more efficiently.

3.5.2 Resource Requirements

Even though it is technically heavy, the project, in fact, does not require a large panel of members. The team simply needs three people - Project Manager, Developer, and Quality Assurance - to look after the whole process, during the estimated schedule of 6 months.

Algorithms for image processing are considered expensive, which consequently requires hardware that allows fast processing, with an optimized memory architecture to reduce memory access latency. Due to the nature of the application and requirements of the project, it is essential that the computer systems utilized are fully functional with CPUs and GPUs integrated. As we are developing with OpenCL, it is preferred that the systems come with AMD, NVIDIA, or Intel graphics cards that support OpenCL SDK.

The most crucial software would be the OpenCL SDK, as well as a Java Compiler and an IDE, since the provided algorithm is in Java. To facilitate version control, an IDE such as Eclipse, that allows pulling and committing group work on GitHub would be more ideal. As mentioned, OpenCL SDK must be manually installed accordingly to the system's specific graphic cards, since most computers do not have OpenCL preinstalled.

4.0 EXTERNAL DESIGN

4.1 External Packages

There are Two main code packages that will be used in the development of this Project:

- JNI - The Java Native Interface is an interface that allows the Java Virtual Machine JVM to Interact with libraries and code native to the system (Oracle, n.d). This allows java code to be run from native applications, or allow Java to run native code to run native applications, by wrapping the native functions with methods that handle adat interchange and access. Through this library, the OpenCL C code will be exposed to Eduad so that it can harness the computational power of the GPU.
- OpenCl SDK - This SKD is imperative to writing parallel code with OpenCl. It consists of a set of C libraries that let C applications interface with the hardware through OpenCL. There is also a set of drivers that are needed to run OpenCl applications, but most modern GPUs and CPUs have them automatically installed.

4.2 Data Sets

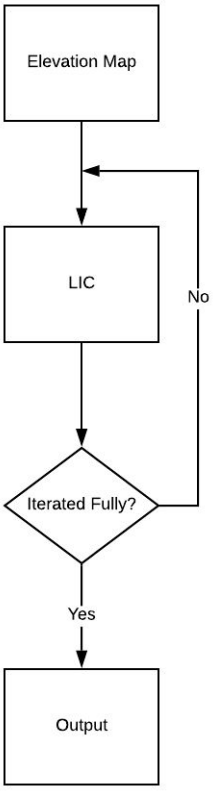
As this project relies on very few external systems, the only significant datasets needed are those used for testing purposes. This test data will be gathered from a variety of sources: Several elevation maps of varying sizes were provided with the source code for Eduard, elevation data is freely and widely available online, and elevation maps may even be generated through many different techniques (perlin or simplex noise for example).

5.0 METHODOLOGY

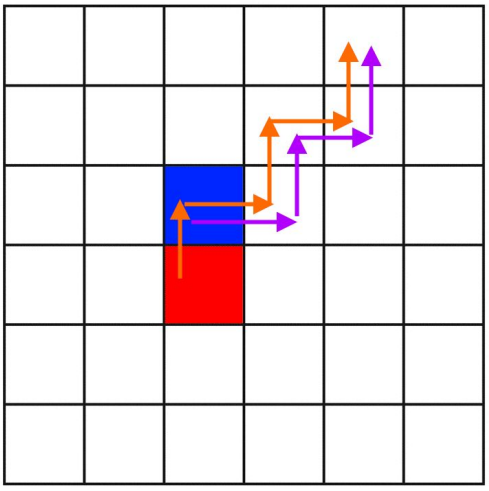
The goal of this project is to utilise GPU hardware in order to accelerate the calculation of the line integral convolution filter. As GPU's excel in running parallel code, a major hurdle in completing this project will be to devise a partitioning scheme that describes how the problem should be broken up in such a way that each individual part of the problem can be run simultaneously. In general, Once a suitable partitioning scheme has been discovered, the effort needed to implement it is low in comparison. For this reason, this Proposal will not provide a specific scheme or implementation for the algorithm, but will briefly outline some possible ways to partition the problem, as well as deal with the integration between the existing software and the functionality to be implemented.

5.1 Algorithm

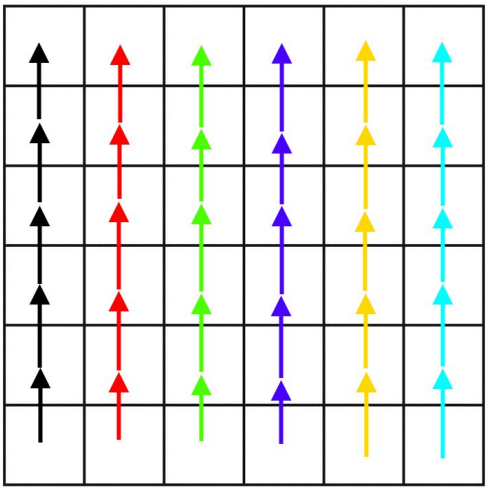
In the case of this project, when applying the Line Integral Convolution filter, instead of using a noise image, the elevation map is used as both as the image to be filtered, and the source of the vector field. Additionally, rather than taking the mean value, the filter takes the summation of the values once they have been passed into a gaussian function. This has the effect of softening and removing the slope lines and ridges from the map, rather than revealing them. The filter is iteratively applied to the result multiple times in order to produce the desired effect of removing detail and small ridges from the map while still keeping the larger features intact.



The implementation of LIC given with Eduard functions by iterating over each point and following the slope uphill and downhill from to calculate the stream line around that point and then applying the appropriate weight before adding them all together. The obvious way to partition the problem is to assign each work item one point and to load the entire map into global memory. This may prove to be quite slow, so it may be worth dividing up the map into separate chunks that each work for particular work groups to work with. It also results in a large amount of repeated work, as a point will have an almost identical stream line the points near it in the stream line.



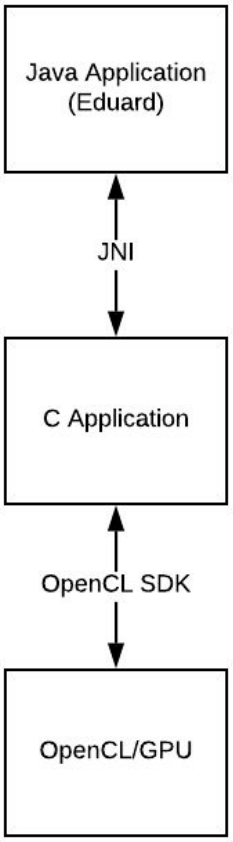
One possible way to reduce the number of repeated calculations is to follow stream lines through the map and evaluate points as it passes over them. Although it is not without its own potential issues, It will be explored during the development process along with other partitioning schemes.



5.2 Integration

The actual algorithm will be written in the C implementation of OpenCl, as it is the standard way of writing OpenCl applications and provides the speed and control that developing in C affords. This decision raises the problem that the existing Java application can't run C code natively, as well as a question of portability: while Java code can run on any platform that has a virtual machine, C must be compiled individually for a specific platform.

The first problem can be easily solved by using the aforementioned Java Native Interface. Integration through the use of JNI will require the writing of wrappers for the functions and methods of the C code that needs to be exposed to the java application. This method gives great control over what data will be shared between the different parts of the application, which can be used to provide a clean API structure for future developers. Part of this integration will be making sure the data is correctly formatted for whichever section is about to use it, for instance, OpenCL kernels have no native support for 2D arrays, so there will have to be some configuration done to make sure that the data going into the kernel is readable.

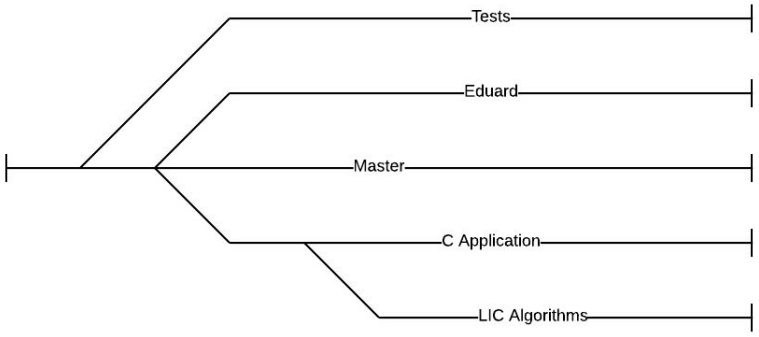


The second problem does not have such an elegant solution. OpenCl is designed to be platform independent, which means that the bulk of the code will not have to change between platforms, however there will still be a need for some checks for platform within the code to be able to run correctly, as well as a separate version compiled for each major platform. At this juncture the plan

is to support Windows and Mac, as they are two of the most popular platforms, as well as being the operating systems used by the development team.

5.3 Version Control

The primary version control tool that will be used is GitLab, as the university provides access to all students, and it has a simple to use interface, potentially reducing the number of human errors that can come about when using command line based solutions. There repository will have three main branches splitting off the master branch: one for Eduard, one for the C application, and one for Testing tools. The C application branch will then have it's own branch for LIC implementations and kernels, and from all these branches there may be sunbranches as parts of each section are developed. Regular code review will be performed in order to ensure that the code being submitted by the team is of a satisfactory level of quality.



6.0 TEST PLANNING

It will be necessary to create a suite of tests that can quickly can be run to compare multiple versions of the software and provide comparisons over a range of test Data. The main goal of the project is to provide an increase in the speed of computation for the filter, therefore the main properties to be tested for will be the accuracy of the implementation – how well it mimics the results of the given implementation, and the speed of the filter – how quickly it can perform the computations. It should be noted that due to floating point errors and the fact that the filter is designed to provide and image for artistic reason the accuracy of the implementation may not have to be completely precise.

Fortuitously there already exists an implementation of the functionality this project seeks to create as well as several examples of input data, which makes it very easy to compare the created implementation against the existing one, as well as generate new cases to test it against. As there may be several competing versions of the algorithm it very important that the test data is plentiful and varied, so that the best algorithm can be selected. The test result data will be categorised by implementation and by version. The same applies to the Implementation of the interface between C and Java, although there will probably be less variation it is important to make sure how much overhead it is causing and whether it is accurately transmitting data.

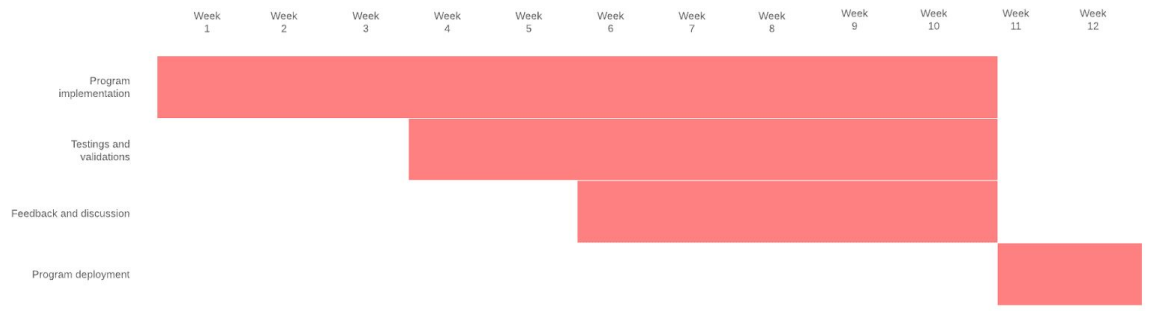
7.0 BIBLIOGRAPHY

1. Antao, S., & Sousa, L. (2010). Exploiting SIMD extensions for linear image processing with OpenCL. 2010 IEEE International Conference on Computer Design. doi: 10.1109/iccd.2010.5647672
2. Bernstein AJ (1966) Analysis of programs for parallel processing. IEEE Trans Electron Comput EC-15:757–762
3. Cypher, R., & Sanz, J. L. C. (1994). The Simd model of parallel computation. New York: Springer-Verlag.
4. Hayes Munson, K. A. (2012). How do you know the status of your project?: Project monitoring and controlling. Paper presented at PMI® Global Congress 2012—North America, Vancouver, British Columbia, Canada. Newtown Square, PA: Project Management Institute.
5. Ludwig, Jamie (n.d.). Image Convolution. Portland State University
6. Project Management Institute (2017). A guide to the project management body of knowledge (PMBOK® Guide)— Sixth edition. Newtown Square, PA: Author.
7. Rajkumar, S. (2010). Art of communication in project management. Paper presented at PMI® Research Conference: Defining the Future of Project Management, Washington, DC. Newtown Square, PA: Project Management Institute.
8. ReviverSoft. (n.d.). File Extension Search. Retrieved from https://www.reviversoft.com/file-extensions/asc.
9. Stone, J & Gohara, D & Shi, G. (2010). OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. Computing in science & engineering. 12. 66-72. 10.1109/MCSE.2010.69.
10. Wallmuller E., (2002), Risk management for IT and software projects, Business continuity: IT Risk management for international corporations
11. Oracle. (n.d). Java Native Interface Specification. https://docs.oracle.com/en/java/javase/11/docs/specs/jni/intro.html

8.0 APPENDIX

Gantt charts

First phase consisting of first two stages during Semester 2 of 2019



Second phase consisting of the last two stages during Semester 1 of 2020

Risk register

ID	Rank	Category	Description	Consequence	Probability	Impact	Risk Response Strategy	Response Implementation Plan	Risk Owner
1	*1*	Personnel	Member's personal schedule leads to delay in project schedule.	Project phases are consequentially delayed, and might cause several complications.	High	High	Mitigate	Communicate in advanced to other team members so schedule can be revised in a proper manner.	Project Manager - Bryan
2	3	Personnel	Member's productivity drops due to lack of motivation.	Project is delayed, or final product might not be up to par.	Medium	High	Avoid	Motivate among team members occasionally check in on the progress weekly.	Project Manager - Bryan
3	2	Technical	Due to technicalities & complexities, the program design is compromised.	Final product is directly impacted.	Medium	High	Avoid	Check requirements thoroughly, check in on other members for further assistance.	Developer - Kelvin
4	4	Technical	Insufficient time for QA process.	Final product is not validated on time.	Low	Medium	Avoid	Carefully develop the SQA plan beforehand.	Quality Assurance - Anh