ROYAL MILITARY COLLEGE OF CANADA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
EEE457 - COMPUTER ENGINEERING DESIGN PROJECT

DID-03 - Statement of Requirements
Parallelization of the Pixel Shiftmap Estimation Algorithm on a GPU

Presented by:
OCdt Jacob Wiebe
OCdt James Dolman

Presented to:
Capt. A. Lapointe
Dr. D. McGaughey

October 20, 2017

TABLE OF CONTENTS

TABLE OF FIGURES

Table of Abbreviations

GPU	Graphics Processing Unit
CPU	Central Processing Unit
GUI	Graphical User Interface

1 INTRODUCTION

This document outlines the requirements, the goal, and the scope of the design project: Parallelization of the Pixel Shiftmap Estimation Algorithm on a GPU. Section 1 will explain the purpose of this document, as well as the project's background, aim, and scope. Section 2 will discuss the decisions made to determine the project's requirements. Section 3 will identify the project requirements for design, development, and validation. Section 4 will discuss risks to development, and the solutions to mitigate these risks. Section 5 will conclude the document.

1.1 Document Purpose

The purpose of the SOR document is broken into the 3 subtopics below:

a. to explain how this project will improve upon pixel shiftmap estimation methods in order to solve the problem of atmospheric distortion of images captured at long ranges;

b. to detail the project requirements, given its purpose; and

c. to discuss the risk associated with development within these requirements.

1.2 Background

Image distortion occurs, to some extent, in all mediums where light is being captured by an aperture. When capturing long-range images over land, the greatest distortion effect is due to atmospheric turbulence. Pockets of air at varying temperatures in the atmosphere refract light non-uniformly, spreading out what was once a beam of photons into more diffuse light. This phenomena can be better explained using a telescope example in Fig. 1. The effect that this distortion has on a high powered optic is shown in Fig. 2.

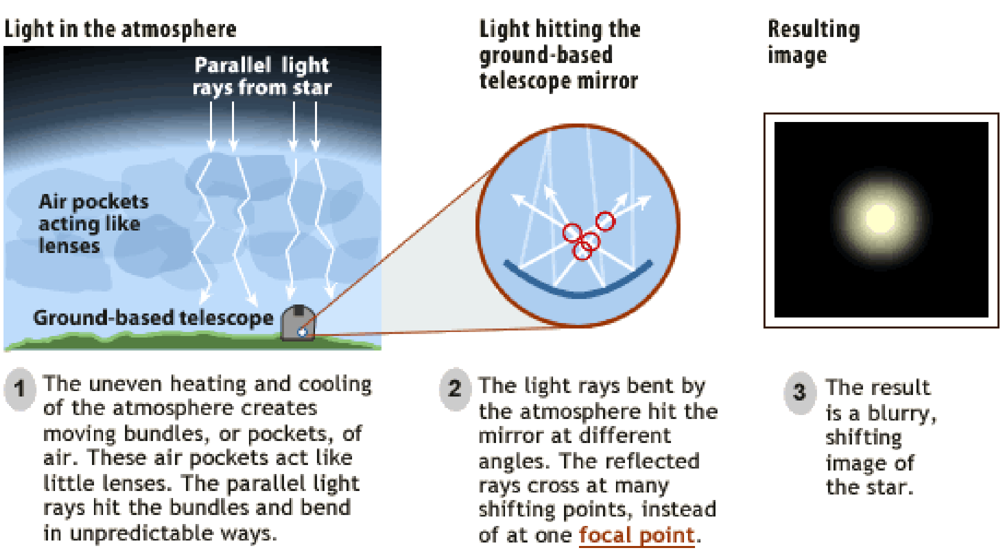

Fig. 1 - Description of atmospheric distortion [1]

Fig. 2 - Effect of atmospheric distortion on an image [2]

As range from the observer to target increases, there are more unpredictable pockets of air that light has to travel through. As a result, images taken at long ranges generally contain non-uniform warping. Due to this, high powered optics in high zoom applications suffer from significantly reduced image clarity. This is an issue to security agencies that require the ability to collect visual data, such as license plate numbers, at long ranges. Take, for example, a helicopter with high powered optics is unable to be used to its full capacity as a surveillance platform if it cannot collect long range, high fidelity images of a target. Therefore, the need for image correction software exists as it gives the user the advantage of concealment, safety, and flexibility. Additionally, there is a need for this to be done as fast as possible. Surveillance optics run in a real time environment, thus image correction computation times should, ideally, be in seconds and smaller time frames.

Due to the highly mathematical and advanced nature of image correction theory, the creation of a unique image correction implementation would be too great a scope for this project. As a result, this project will use an existing optical flow algorithm for the generation of a pixel shiftmap based on the work of Brox *et al.* [3] This algorithm uses two nested fixed point iterations to calculate the spatial gradient between two images. The spatial gradient is computed via the first order derivatives of the x and y components of the optical flow of two images (density mapping). This method gives significantly smaller angular errors than previous optical flow estimation techniques. [4]

Currently, the serial MATLAB implementation of Brox's algorithm requires a very long processing time (one to two hours with typical parameters). A GPU can increase the speed of Brox's algorithm by running this computation on multiple pixel arrays at the same time. In direct contrast to the few large cores in a CPU which are optimized for fast sequential processing, GPUs have a massively parallel architecture that consists of thousands of smaller cores, optimized for the efficient handling of multiple small tasks simultaneously. [5] Brox's optical flow estimation algorithm is ideal for GPU parallelization as there are many accesses to data and simple computations.

1.3 Aim

The aim of this project is to design a program that will create a pixel shift map by leveraging the parallel processing capabilities of a GPU, with the intent of minimizing its computation time. The finished program will be applicable for image correction systems with the effect of significantly increasing the speed of image dewarping.

1.4 Scope

The project will present the implementation of Brox's optical flow algorithm, generating a pixel shiftmap on a GPU. It will not perform a complete image correction. Instead, it will generate the estimated pixel shiftmap; being the most computationally intensive procedure of image dewarping algorithms. The generation of the shiftmap consists of many data transfers and simple operations. It is therefore ideal for implementation on a GPU.

The project is limited to the use of a single, specific GPU as this is what is available to the project team. In a more robust design scenario, the ability to run the software on multiple GPU configurations would allow for greater flexibility or performance. Ideally, the image processing time would be reduced to milliseconds so that the corrected images could be used in real time applications. As this project is limited in the hardware available, the project's design performance has been scaled down to longer than real time applications would permit.

The design itself will consist of first converting the current MATLAB implementation of Brox's algorithm into C++ while keeping in mind the goal of parallelization. After this, an iterative design approach will be used to parallelize increasingly greater sections of the C++ program. Testing will be performed upon completion of each design iteration. To test for product accuracy, error calculations will be made from the discrepancy between control image sets with known pixel shiftmaps, and the program's outputs.

2 REQUIREMENT DEFINITION ACTIVITIES

To formulate the product requirements, meetings with the project supervisor were conducted. Using the current implementation of Brox's Algrorithm, a MATLAB speed benchmark was decided on as the baseline performance requirement. The inputs and outputs for the program are well defined as they correlate to Brox's algorithm. A complete image correction algorithm, including application of the shiftmap, was considered. However, it was decided to be outside of the scope of this project, given the lab team's limited experience in both parallel programming and image processing. An error of 10% was selected based on the potential data loss of converting from floating to fixed point numbers. NVIDIA's CUDA Toolkit will be used due to its popularity and availability. It requires the use of the C or C++ programming languages in order to interface with the GPU, therefore C++ was decided on due to its greater functionality, particularly from an object-oriented perspective.

3 PRODUCT REQUIREMENTS

3.1 Functional Requirements (FR)

FR-01: <u>Program Input</u> - A reference image and an image distorted with a known pixel shift, will be used for the program's inputs.

FR-02: <u>Program Product</u> - The program will produce a pixel shift map in the form of two arrays: the x component of the shift and the y component of the shift.

3.2 Performance Requirements (PR)

PR-01: <u>Processing Speed</u> - The pixel shift map will be generated, at a minimum, in equal time as a benchmark of the original pixel shift map algorithm written in MATLAB.

PR-02: <u>Program Error</u> - The converted Brox algorithm must have a mean squared error of less than 10%.

3.3 Interface Requirements (IR)

IR-01: <u>Implementation of Parallel Code on GPU</u> - The program will interface with a GPU in order to perform calculations in parallel.

IR-02: <u>CUDA GPU Communication</u> - NVIDIA's CUDA Toolkit shall provide the interface for sending commands and data between the programmer and GPU.

IR-03: <u>Program User Interface</u> - This program shall have an expert level user interface (i.e. the user will run a C executable file in the command line).

3.4 Implementation Requirements (IR)

ImpR-01: C++ Implementation - The product shall be written entirely in the C++ programming language.

ImpR-02: CUDA Implementation - Development of the parallel algorithm shall be accomplished using NVIDIA's CUDA parallel computing platform.

ImpR-03: GPU Specification - The program parallelization will be implemented on, and developed for, the Nvidia Quadro K2000D.

ImpR-04: MATLAB Benchmark - The benchmark of the Brox Optical Flow MATLAB code will be run on a Intel Xeon E5430 CPU.

ImpR-05: Data Representation - The program will convert 64-bit variables used by MATLAB into 8-bit variables for use in C++.

3.5 Schedule Restrictions (SR)

SR-01: First Prototype: The first functional prototype of initial Brox Optical Flow Computation Algorithm translation into C++ shall be available for testing no later than January 8, 2018.

4 RISK ASSESSMENT

4.1 Translation of Code

Due to its criticality in the implementation of the project, the translation of Brox's Optical Flow Computation Algorithm from MATLAB into C++ code is essential. This will be the first design challenge of the project. As there are differences between MATLAB and C++, the translation will not be direct. Design considerations must be taken as to what data structures the C++ program will use and how it will represent floating point values in fixed point variables. Additionally, decisions will have to be made about how the images will be loaded and how it will handle different file formats of images. The likelihood of issues occurring from this is high, however, the risk to the project is low, and it will be mitigated by putting a large focus on the translation of code as part of the design process. Ample time should be available to overcome any translation inconsistencies without falling behind schedule.

4.2 Implementation on GPU

The heart of the project will be to build the C++-implemented Brox algorithm that enables computations to be executed in parallel on a GPU. Time must be taken to dissect how to accurately and safely handle and store data, both on and off the GPU. The data structures which best utilize the computational potential of the GPU for image partitioning and merging also need to be determined. This is an area of high impact as this parallelization of the sophisticated Brox algorithm is the core design component of this project. To mitigate this risk, an engineering process shall be employed where functionality is implemented and tested on the GPU incrementally in subsequent revisions. A challenge will be to develop a release schedule so that project momentum and direction can be focused and maintained.

5 CONCLUSION

This preliminary document introduced and detailed the product requirements of the the Parallelization of the Pixel Shiftmap Estimation Algorithm on a GPU project. Additionally, the aim, scope and risks associated with this project were discussed. This document serves to set the intent and direction for the preliminary design of the project.

REFERENCES

[1] S. T. S. Institute, “Why stars twinkle: Atmospheric distortion of light,” 2008. [Online]. Available: https://amazing-space.stsci.edu/resources/explorations/groundup/lesson/basics/g18a/. [Accessed: 28-Sep-2017].

[2] B. Mahar, “smc pentax a* 300mm vs soligor 650mm mirror lens,” 2012. [Online]. Available: http://blog.trafficshaper.com/?p=569. [Accessed: 28-Sep-2017].

[3] N. P. Thomas Brox, Andrés Bruhn and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” *8th European Conference on Computer Vision*, vol. 322, pp. 25–36, 2004.

[4] N. M. Javier Sánchez and A. Salgado, “Robust optical flow estimation,” vol. 2013, no. 21, 2012, pp. 242–261.

[5] NVIDIA, “What is gpu-accelerated computing,” 2017. [Online]. Available: http://www.nvidia.ca/object/what-is-gpu-computing.html [Accessed: 05-Oct-2017].