A PROJECT REPORT
ON

"IMPLEMENTATION OF SEAM CARVING FOR IMAGE RETARGETING USING CUDA ENABLED GPU"

Submitted to

SAVITRIBAI PHULE UNIVERSITY OF PUNE

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

BY

Fardeen Ahmed	**Exam No: B80434202**
Swapnil Dhage	**Exam No: B80434218**
Kalpesh Dusane	**Exam No: B80434223**
Prathmesh Savale	**Exam No: B80434308**

Under The Guidance Of
Prof. Balasaheb B. Gite

Sponsored By: Persistent Systems Pvt. Ltd. Pune.

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
Sinhgad Academy of Engineering
Kondhawa (Bk.), PUNE - 411048

2014-2015

AFFILIATED TO

SAVITRIBAI PHULE UNIVERSITY OF PUNE

A PROJECT REPORT
ON

"IMPLEMENTATION OF SEAM CARVING FOR IMAGE RETARGETING USING CUDA ENABLED GPU"

Submitted to

SAVITRIBAI PHULE UNIVERSITY OF PUNE

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

BY

Fardeen Ahmed	**Exam No: B80434202**
Swapnil Dhage	**Exam No: B80434218**
Kalpesh Dusane	**Exam No: B80434223**
Prathmesh Savale	**Exam No: B80434308**

Under The Guidance Of
Prof. Balasaheb B. Gite

Sponsored By: Persistent Systems Pvt. Ltd. Pune.

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
Sinhgad Academy of Engineering
Kondhawa (Bk.), PUNE - 411048

2014-2015

AFFILIATED TO

SAVITRIBAI PHULE UNIVERSITY OF PUNE

Sinhgad Academy of Engineering

Department of Computer Engineering
Kondhawa, Pune – 411048

Sinhgad Institutes

CERTIFICATE

This is certify that the project entitled

"IMPLEMENTATION OF SEAM CARVING FOR IMAGE RETARGETING USING CUDA ENABLED GPU"

Submitted By

Fardeen Ahmed	**Exam No: B80434202**
Swapnil Dhage	**Exam No: B80434218**
Kalpesh Dusane	**Exam No: B80434223**
Prathmesh Savale	**Exam No: B80434308**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Engineering) at Sinhgad Academy of Engineering, Kondhwa, Pune under the University of Pune. This work is done during year 2014-2015, under our guidance.

Date: / /

(Prof. Balasaheb B. Gite)
Project Guide

(Prof. Ravindra C. Sanap)
Project Coordinator

(Prof. Balasaheb B. Gite)
HOD, Computer Department

(Dr. Vijay M. Wadhai)
Principal

External Examiner

April 22, 2015

To Whomsoever it May Concern

This is to certify that following students have completed their final year B.E. project naming"Implementation of seam carving for image retargetting using CUDA enabled GPU" at Persistent Systems Ltd. under the guidance of Mr. Pandurang Desai for academic year 2014-15.

Name of Students:

i. Prathmesh Savale
ii. Fardeen Ahmed
iii. Kalpesh Dusane
iv. Swapnil Dhage

For Persistent Systems Ltd.

Kaustubh Bhadbhade
Senior Manager - Human Resource

Acknowledgements

We are profoundly grateful to **Prof. B. B. Gite**, Head of Department of Computer Engineering, for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would also like to thank our mentor **Mr. Pandurang Desai** at Persistent Systems Pvt. Ltd. Pune for his valuable suggestions and guidance from his experience in the industry all throughout our project.

We would like to express deepest appreciation towards **Dr. V. M. Wadhai**, Principal, Sinhgad Academy of Engineering, Kondhawa, and **Prof. R. C. Sanap**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

Fardeen Ahmed
Swapnil Dhage
Kalpesh Dusane
Prathmesh Savale

ABSTRACT

The purpose of this project is to show the difference between executing the seam carving algorithm using sequential approach on a traditional CPU (central processing unit) and using parallel approach on a modern CUDA (compute unified device architecture) enabled GPU (graphics processing unit). Seam Carving is a content-aware image resizing method proposed by Avidan and Shamir of MERL. It functions by identifying seams, or paths of least importance, through an image. These seams can either be removed or inserted in order to change the size of the image. It is determined that the success of this algorithm depends on a lot of factors: the number of objects in the picture, the size of monotonous background and the energy function.

The purpose of the algorithm is to reduce image distortion in applications where images cannot be displayed at their original size. CUDA is a parallel architecture for GPUs, developed in the year 2007 by the NVidia Corporation. Besides their primary function i.e. rendering of graphics, GPUs can also be used for general purpose computing (GPGPU). CUDA enabled GPU helps its user to harness massive parallelism in regular computations. If an algorithm can be made parallel, the use of GPUs significantly improves the performance and reduces the load of the central processing units (CPUs). The implementation of seam carving uses massive matrix calculations which could be performed in parallel to achieve speed ups in the execution of the algorithm as a whole. The entire algorithm itself cannot be run in parallel, and so some part of the algorithm mandatorily needs a CPU for performing sequential computations.
Keywords: Seam Carving, CUDA, Parallel Processing, GPGPU, CPU, GPU, Parallel Computing.

Contents

List of Figures

ABBREVIATIONS

The following are the list of conventions and acronyms used in this document and the project as well:

1. CPU- Central processing unit
2. GPU- Graphics processing unit
3. CUDA- Compute unified device architecture
4. GPGPU – General purpose graphics processing programmability
5. DFD- Data Flow Diagrams

Chapter 1

Introduction

1.1 Background and Basics:

What is Seam Carving?

Seam carving is an algorithm for image resizing, developed by Shai Avidan, of Mitsubishi Electric Research Laboratories (MERL), and Ariel Shamir, of the Interdisciplinary Center and MERL. It functions by establishing a number of seams (paths of least importance) in an image and automatically removes seams to reduce image size or inserts seams to extend it. Seam carving also allows manually defining areas in which pixels may not be modified, and features the ability to remove whole objects from photographs. The purpose of the algorithm is to display images without distortion on various media (cell phones, PDAs) using document standards, like HTML, that already support dynamic changes in page layout and text, but not images.

What is a seam?

Seams can be either vertical or horizontal. A vertical seam is a path of connected pixels from top to bottom in an image with one pixel in each row. A horizontal seam is similar with the exception of the connection being from left to right. The importance/energy function values a pixel by measuring its contrast with its neighbor pixels.

Figure 1.1: Original image and single seam

Figure 1.2: 50 seams carved and removed

Why seam carving?

Effective resizing of images should not only use geometric constraints, but consider the image content as well. Conventional image resizing consists of cropping or evenly down sampling that lead to loss of important features or distortion. Whereas on the other hand seam carving method enables us to remove pixel from uninteresting parts of the image while preserving important content.

Figure 1.3: Original and scaled image (distorted)

Figure 1.4: Cropped and Seam carved image

As seen in Figure 1.4b. We can clearly notice that Seam carving method preserves the interesting features of the image (here castle and person) when resized the image. On the contrary in Figure 1.3b. Scaling introduces geometric distortion. And in Figure 1.4a. Cropping resulted in loss of important features of the image (person on the left).

Seam carving can be used for distortion free image expansion by inserting least energy seams in the image. This can also be extended to object removal and object protection in an image.

What is CUDA?

CUDA (Compute Unified Device Architecture) is an open parallel architecture, introduced in 2007, developed by Nvidia. It is a computational drive, which is used in graphics processors, and is accessible to software developers through the standard programming languages. It mainly uses C as an extensing language for CUDA, along with other languages such as Perl, Python, Java, Fortran and Matlab environment. CUDA gives developers access to a limited set of command and GPU resources. CUDA exploits the potential of the powerful graphics processing units in modern computers. The main difference in the architecture of CPU and GPU is in the proportion of transistors that they designed with, as well as the number of tasks that they can perform in parallel. The first difference can be seen in the Figure 1.5. below. The orange color is highlighted memory part, to control yellow, and green represents arithmetical logical part of the process unit.

Figure 1.5: Architecture CPU vs. GPU

How CUDA works?

Figure 1.6. Shows an example of the use of graphics processor. The operation is divided into four phases:

1. Transfer of data from main memory to GPU memory
2. CPU commands GPU to perform the task
3. Graphics processing unit processes the data
4. Transfer of data from GPU memory to main memory

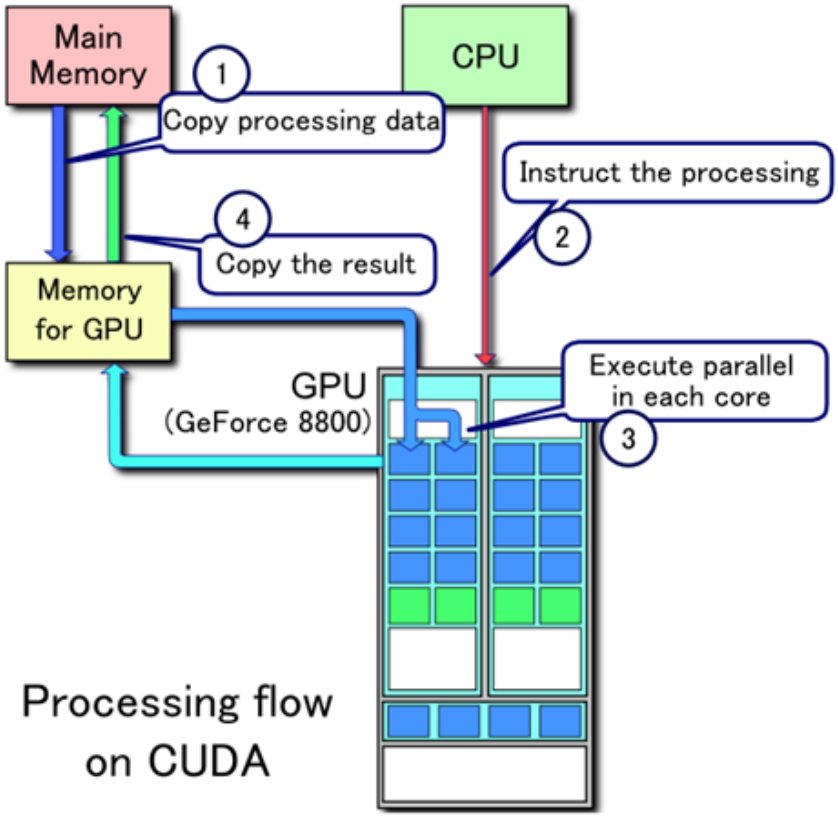

Figure 1.6: CUDA processing flow

CUDA programming model

Software code written for CUDA system is divided into two units, the parallel and the serial code. The serial code is carried out on the CPU and contains mainly commands to transfer data to the GPU and vice versa. Otherwise it may be a serial code which includes a part of the algorithm, which is implemented on the CPU. In CUDA programming model, CPU is known as the host and GPU is known as the device. A piece of code that runs on the GPU is known as thread. A collection of threads, known as a block runs on a single CUDA processor, where all threads within the block run in parallel. The image to be processed is divided into such blocks. A collection of blocks is known as a grid. A C-like function that runs on the GPU is known as kernel. The kernel call includes the specification of gridsize, blocksize and other specifications that are required. Figure 1.7. shows the block diagram of CUDA programming model.

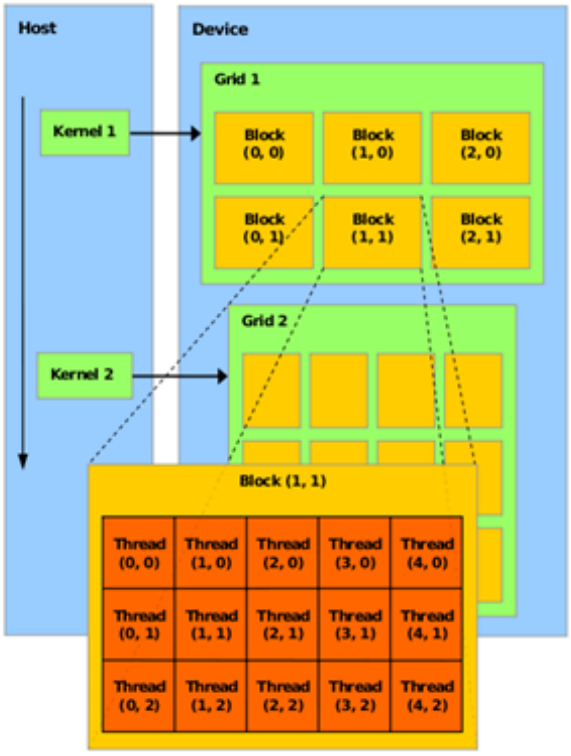

Figure 1.7: CUDA architecture

1.2 Relevance:

Adobe Systems acquired a non-exclusive license to seam carving technology from MERL, and implemented it as a feature in Photoshop CS4, where it is called Content Aware Scaling. As the license is non-exclusive, other popular computer graphics applications, among which are GIMP, digiKam, ImageMagick, as well as some stand-alone programs, among which are iResizer, also have implementations of this technique, some of which are released as free and open source software. Seam carving can also be used in digital and computational photography as well as can be implemented in mobile devices for real-time usage. With the use of mobile GPUs such as TEGRA K-1 powered devices such technology can be also used in video retargeting.

The primary advantage of this technology is for data analytics companies who need to render visual data on different resolution devices faster and in an efficient manner.

1.3 Project Undertaken:

1.3.1 Problem Definition:

To implement Seam carving application, this takes an image file as the input and generates a resized image file as the output. The central idea is to demonstrate the difference in execution time between running the algorithm on a CPU and a GPU.

The scope of our project can be increased to seam insertion or object removal within an image. The overall emphasis of our system is to show the difference between processing speeds of a CPU and a GPU.

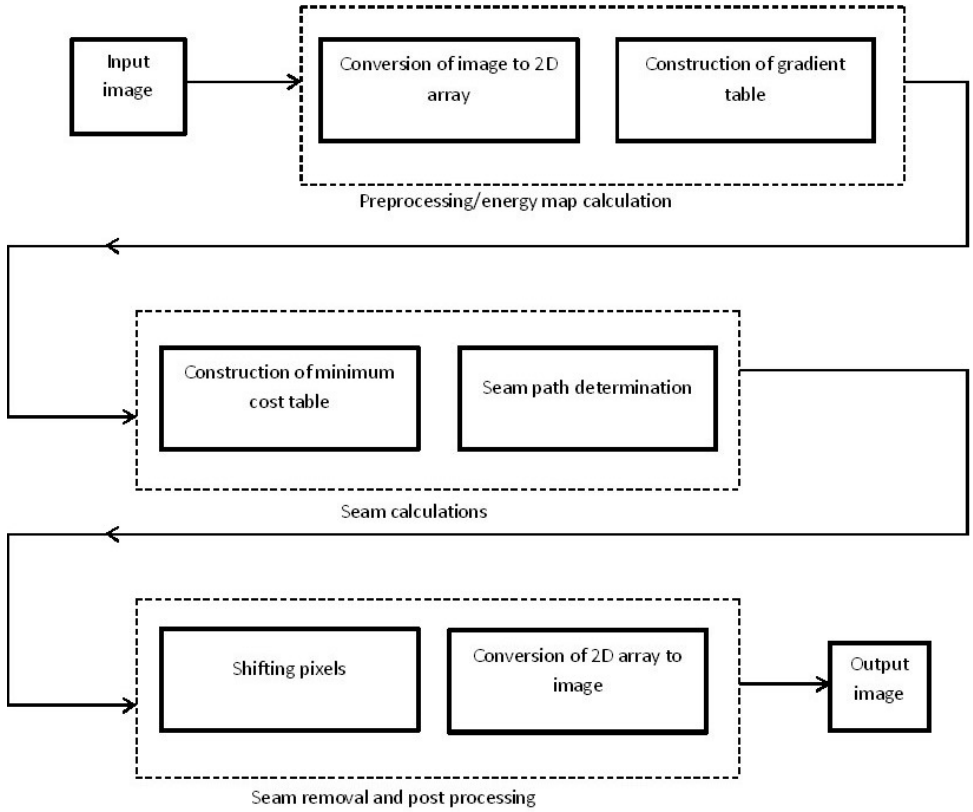

Figure 1.8: Block Diagram of proposed system

1.3.2 Mathematical Model:

Sequential Implementation:-

1) For Energy Calculation,

Complexity:- t = (r*c) where t = total no. of pixels

2) For Each Seam Calculate Minimum cost,

For each seam : Complexity:- (n + n + (n - p)) = (3n - p) here, Identifying min in a row of min

costs:- n iterations ,(n-p) pixels to shift (where p is the Position of the min pixel. For K seams :

Complexity:- = k(3n - p)

TOTAL Complexity:- [t + (3n - p)] * k

Parallel Implementation:-

1) For Energy Calculation,

Complexity:- [n / (P * 32 * 32)]

Here P = No. of CUDA processors.

BlockSize=32 and GridSize=32

2) For Each Seam PATH,

For each seam : Complexity:- [(r * c)/(P*32*32)]

3) Removing Seam,

Complexity:- [r * log2 c]

TOTAL Complexity:-

[n / (P * 32 * 32)] + [(r * c) / (P * 32 * 32)] + [r * log2 c] * k

1.3.3 Scope Statement:

This project is undertaken to establish a system to resize an image file based on the limit specified by the user. This project is being constructed to take the advantage of graphics processing unit to accelerate a sequential algorithm so that even larger size images can be processed in lesser amount of time and more efficiency.

Includes:

- The facility to resize images using content aware resizing technique.
- This can be done sequentially using only CPU as well as in parallel using CUDA enabled GPU.
- Facility to draw seams that are to be carved in parallel.
- Has an interactive GUI for naïve users.
- Has a command line interface for advanced users.

1.4 Intended Audience and Reading Suggestion:

This document is aimed at providing an overview of the entire project to its Developers, Intended Users and Professors reviewing this project and whomsoever it may concern. The document begins with a brief overview of the project and its aim and proceeds to define all strategies the team has planned to incorporate in the development of this project. The reader is advised to start with the introduction and scope of the project and proceed further, for a lucid understanding of the project plan and intended usage.

Chapter 2

Literature Survey

Considerable work has been done in the field of Seam Carving and CUDA, most prominent and relevant papers from which we have gained knowledge are:

1. AVIDAN, S., AND SHAMIR, A. 2007. Seam carving for content aware image resizing. ACM Trans. Graph. 26, 3, 10.

Introduction:

This is the first paper of the original work of Shai Avidan in the field of seam carving. He proposes a new method for image resizing that is content aware in nature and shows the drawbacks of using other image resizing methods such as scaling, cropping etc. We also learn a technique for finding optimal seam path within an image after calculation of energy distribution using dynamic programming, which is used in our algorithm.

Image Energy Conservation:

A comparison of the preservation of content measured by the average pixel energy using five different strategies of resizing.

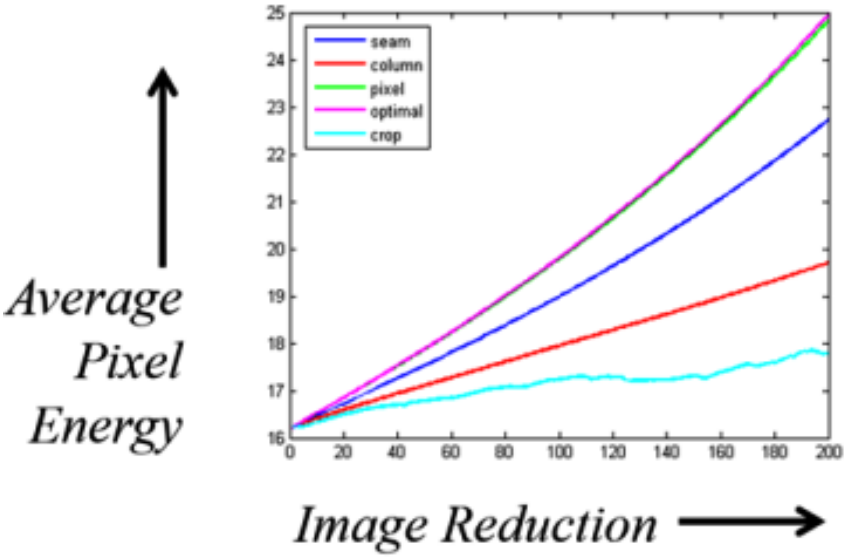

Figure 2.1: Image energy preservation.

Use of dynamic programming:

The optimal seam can be found using dynamic programming. The rst step is to traverse the image from the second row to the last row and compute the cumulative minimum energy M for all possible connected seams for each entry (i, j):

$$M(i, j) = e(i, j) + min(M(i1, j1), M(i1, j), M(i1, j+1))$$

At the end of this process, the minimum value of the last row in M will indicate the end of the minimal connected vertical seam. Hence, in the second step we backtrack from this minimum entry on M to nd the path of the optimal seam. The denition of M for horizontal seams is similar.

This is better described by the figures below. Each square represents a pixel, with the top-left value in red representing the energy value of that said pixel. The value in black represents the cumulative sum of energies leading up to and including that pixel.

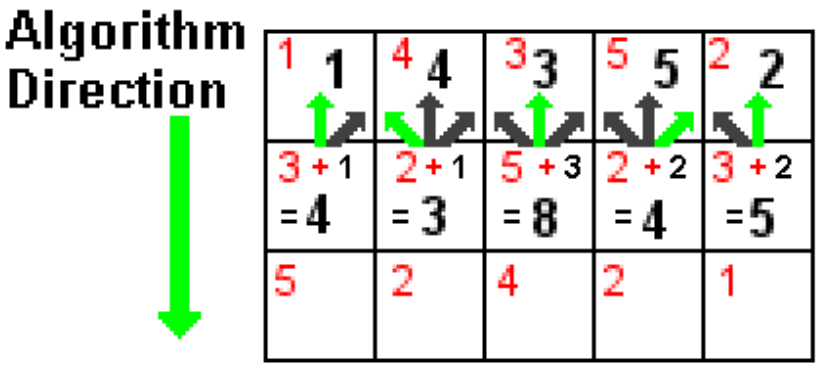

Figure 2.2: Optimal path calculation from 2nd row

The first row has no rows above it, so the sum (black) is just the energy value of the current pixel (red). The second row, if we look at the second pixel for example, we see its energy value is 2 (red). If we look above it, it has a choice of 1, 4, or 3 (black). Since 1 is the minimum number of the three values, we ignore the other two and set the sum of the pixel to its energy value which is 2 (red) plus 1 (black).

After the above operation is carried out for every pixel in the second row, we go to the third row:

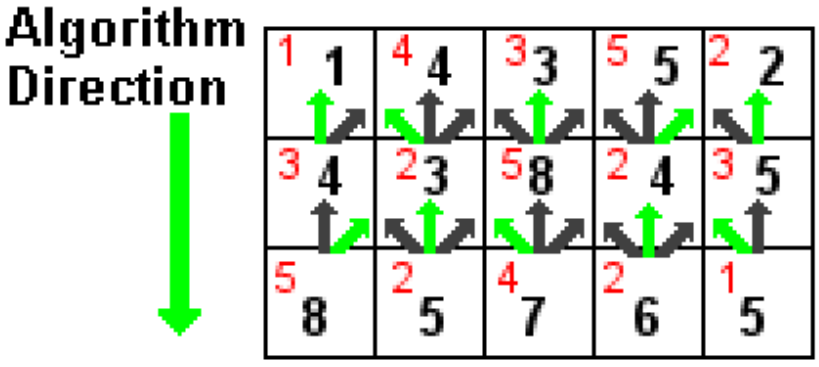

Figure 2.3: Optimal path calculation continued

We repeat the process in row two in row three to end up with the final cumulative sums for the seams/paths. The lowest value or values are the seams with the lowest energy, which would be in this example the seams with '5' in the last row.

To trace the seam/path, work from the last row and follow the green arrows:

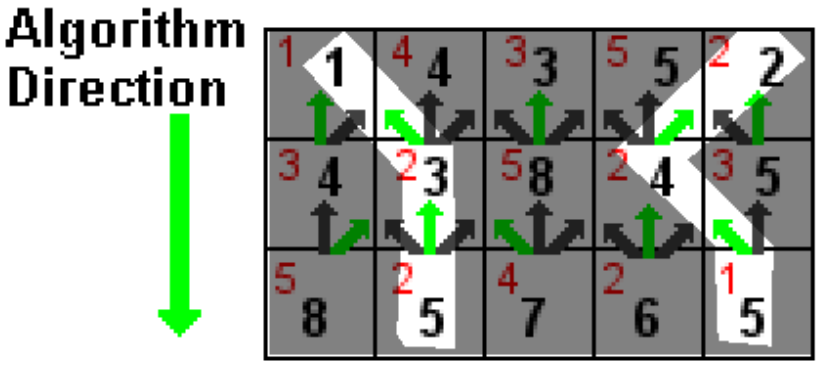

Figure 2.4: Backtracking

Conclusion:

Here the authors provided a solution for seam carving problem, using a slow pre-processing step. With this step we determine the sequence in which the seams can be removed and the information about each seam. As we can imagine this step is very memory consuming. A number of papers were published on the topic of real-time content-aware image resizing.

2. Optimization of a single seam removal using a GPU, Rok Cešnovar, Patricio Bulic, Tomaž Dobravec University of Ljubljana, Faculty of Computer and Information Science, Tržaška c. 25, Ljubljana, Slovenia

Introduction:

This paper provides an idea of using a GPU for the implementation of seam carving algorithm. As seam carving involves massive matrix calculations the idea of using a GPU for offloading the complex calculations from the CPU will help in accelerating the execution speed of the algorithm.

Proposed Framework:

A seam is an 8connected path of low energy pixels crossing the image from top to bottom, or left to right, while only one pixel in a row or column, respectively, can be a part of the seam. In order to determine which pixels belong to the optimal seam, dynamic programming is used. This proposed method can be divided in 4 steps:
1) Energy analysis
2) Cumulative minimum energy computation
3) Backtracking for optimal seam determination
4) Seam removal

In order to reduce/enlarge the image in any dimension for k pixels, all of the above steps have to be repeated k-times in order to achieve the best results. Repeating all the steps is very time consuming, thus real-time application cannot be achieved this way. Instead of focusing on changing the seam carving algorithm, author focused on and optimized implementation of the original single seam removing method on GPUs. The optimization is based on the properties of the GPU. This paper presents the results of this implementation and the results of the optimization steps. The method was implemented on graphics processing units with the CUDA architecture and programming model.

Conclusion:

The approach of using CUDA for performing tasks in parallel is the most efficient and optimised method for accelerating seam carving algorithm.

Others:

3. In "Chen-Kuo C., Shu-Fan W., Yi-Ling C., Shang-Hong L. Fast JDN-Based Video Carving with GPU Acceleration for Real-time Video Retargeting"

Here the authors proposed an efcient improvement of the seam carving method, by detecting local and global seams. This way multiple seams can be removed in each step, instead of the afore mentioned repetitions of single seam removal.

4. In "Huang H., Fu T., Rosin P., Qi C. Real-time content-aware image resizing. June 2010, CUDA Toolkit"

Here the authors also propose a more efcient approach to seam based content aware image resizing, searching the seams through establishing the matching relation between adjacent rows or columns

Chapter 3

Requirement Analysis

3.1 Hardware Requirements

The Hardware requirements are as follows

1	Processor	Minimum Requirement: Intel Pentium 4 or above/AMD Athlon processor range or above. Recommended Requirement: Intel i3 2nd Generation or above/AMD Phoenix or above.
2	Memory	Minimum Requirement: 100MB or above. Recommended Requirement: 250MB or above.
3	Hard disk space	Minimum Requirement: 120MB or above. Recommended Requirement: 200MB or above.
4	Others	CUDA enabled GPU's manufactured by NVIDIA present in PC's OR Laptops

3.2 Software Requirements

The software requirements are:

1	Operating System	Windows (XP and above)
2	Others	.NET 3.5 and up, visual c++ runtime environment.

Chapter 4

Analysis and Design

4.1 Use-Case Diagram

The following diagram shows 3 actors interacting with the seam carving system; the primary actor (user) triggers the execution inside the system by selecting the image to be carved and then initiating the system for execution. After this the CPU will execute the sequential part of the system and the GPU will execute the parallel part of the system.

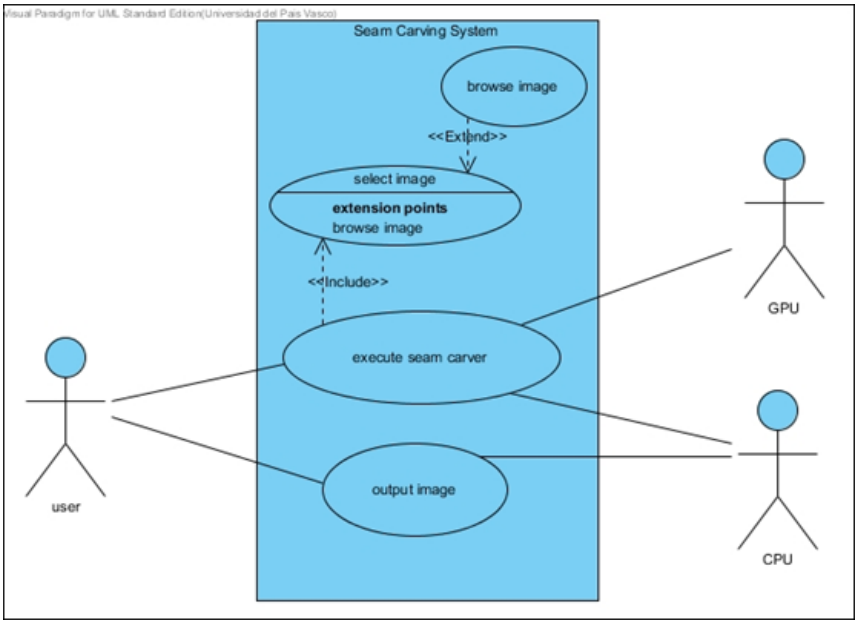

Figure 4.1: Use Case

USE CASE NAME	SEAM CARVE SYSTEM
ID	1
BRIEF DESCRIPTION	OVERALL SYSTEM DESCRIPTION
PRIMARY ACTORS	USER

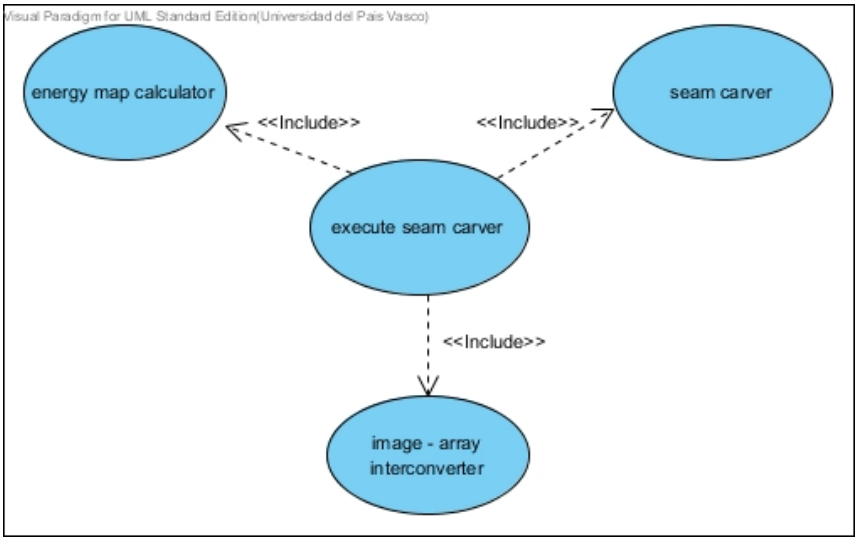

Figure 4.2: Extended Use Case

USE CASE NAME	EXTENDED USE CASE FOR SEAM CARVER
ID	2
BRIEF DESCRIPTION	OVERALL SYSTEM DESCRIPTION

4.2 Class Diagram

From the following diagram it is clear that our system consists of 4 classes, each performing its own job. The ?image? class is responsible for accepting the image to be carved from the user and then convert it into a 2D array for further processing, the seam carve system is an aggregation of 2 different classes viz. ?energymap? and ?seamcarver?, where energymap class performs the energy calculations of the image and the seam carver class is responsible for finding optimal seams and removing them.

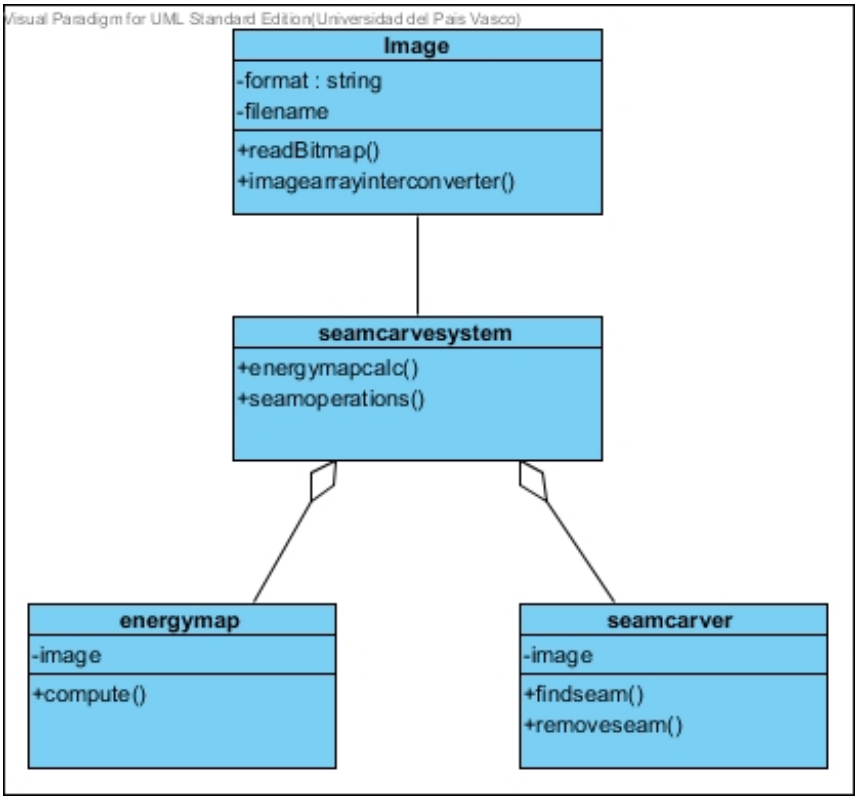

Figure 4.3: Class Diagram

4.3 Data Flow Diagrams

Figure 4.4: Level 0 DFD

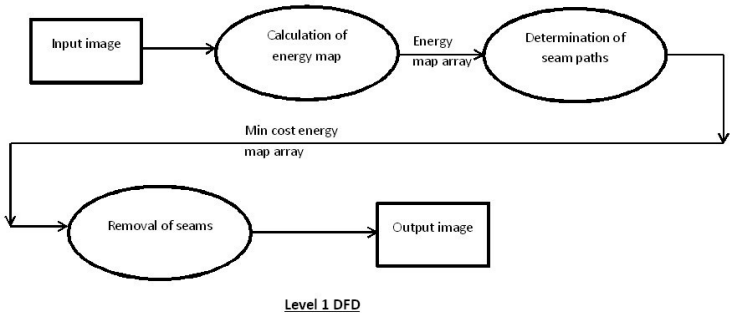

Figure 4.5: Level 1 DFD

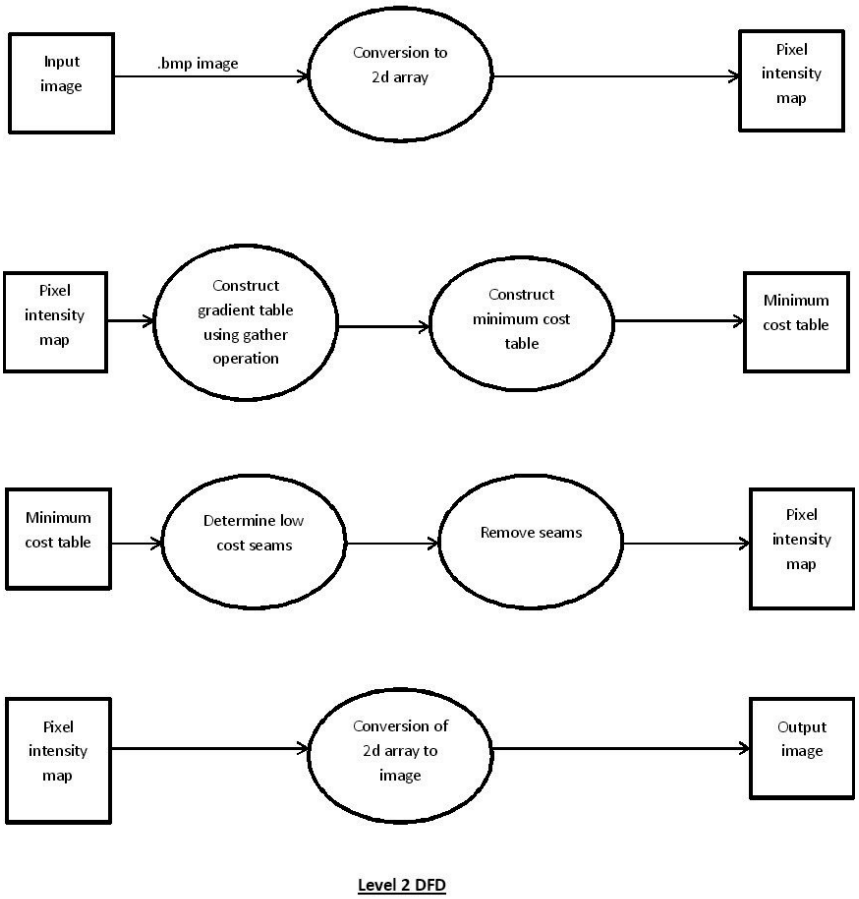

Figure 4.6: Level 2 DFD

4.4 Analysis Activity Diagram

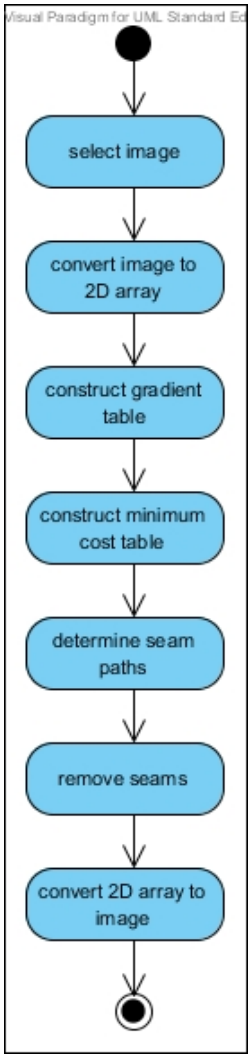

Figure 4.7: Activity Diagram

4.5 Analysis State Chart Diagram

The following diagram shows specifically the different states that the system will be in while execution. Initially the system is in an idle state and later the user selects the image to be carved, after this the user enters the number of seams to be removed and then presses enter to continue execution, the system then executes on to remove the seams to present the required output.

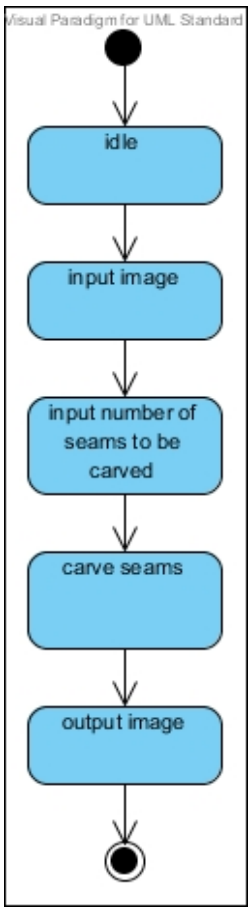

Figure 4.8: State Chart Diagram

4.6 Component Diagram

The following diagram consists of all the components and modules inside our project seam carving system, it can be clearly seen from the diagram that we are using 2 different types of programming paradigms one as parallel which will be done in CUDA-C (.cu extension) and executed on the GPU, and the another is sequential which is done in C++ (.cpp extension) and will be executed on a CPU.

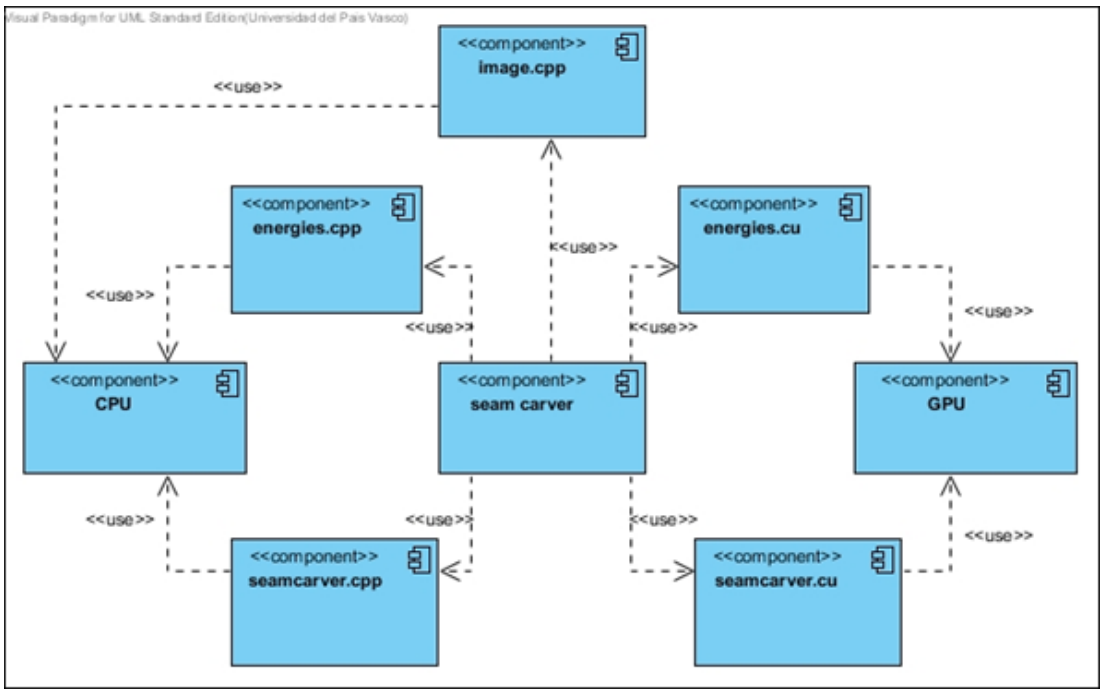

Figure 4.9: Component Diagram

4.7 Sequence Diagram

The following diagram shows the different classes of the seam carving system and how they are interacting with each other following a hierarchy of execution . after the user selects the image the redBitmap() function of the Image class reads the image, futher the image class converts this image into a 2D array bassed on the pixel intensities and RGB values, futher the aggregation class seamcarvesystem uses compute() function of energymap to calculate the gradient table, after this the minimum cost table is calculated and later the calculatted seams are removed from the image. After this the carved image is converted from the 2D array back into an image.

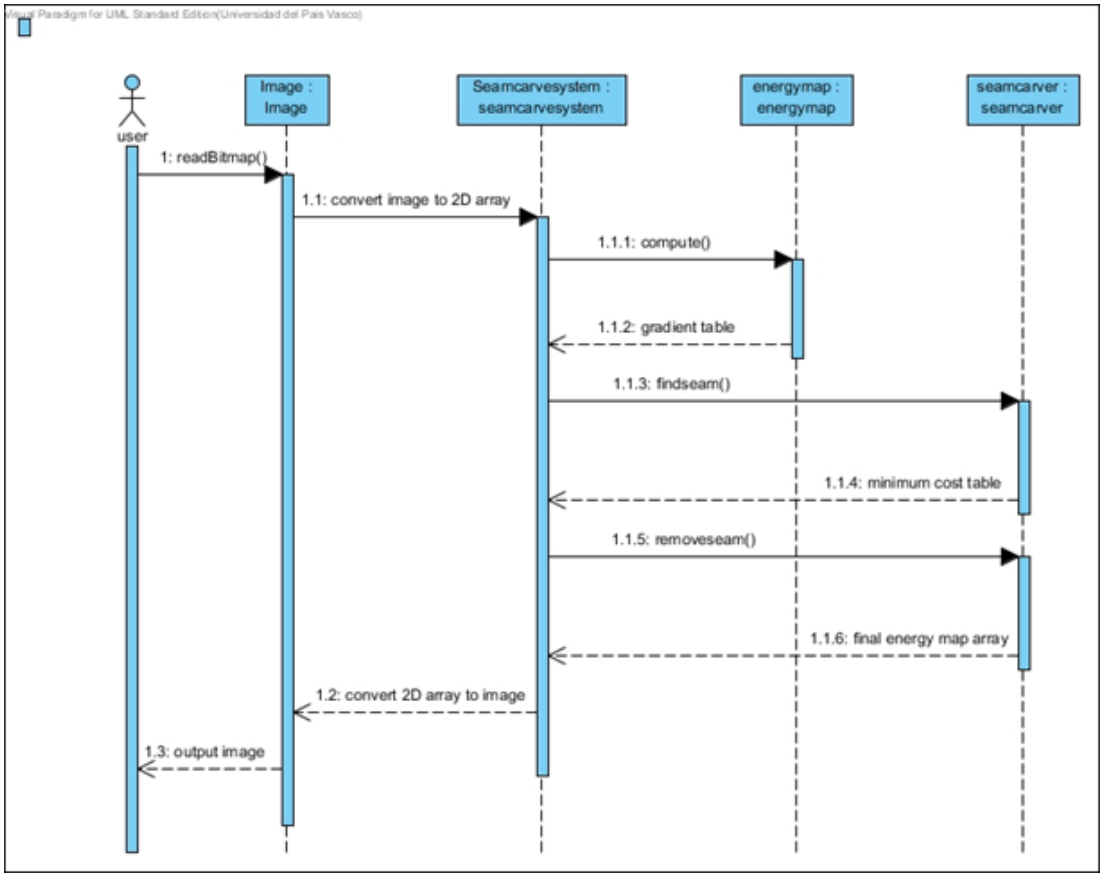

Figure 4.10: Sequence Diagram

Chapter 5

Project Planning and Management

5.1 Estimation and Efforts

Cost of the system is calculated from the cost of various software components used to develop it. As the development done is open source and done using open source technologies cost for implementation is none. Else, the efforts and time utilized add up to the cost.

The project development should take about 6 months. The time needed to study the syntax and commands are included. Testing is done for software to see whether all the functionalities are working correctly. Time is needed to test the code, using unit testing and system testing.

5.2 Project Scheduling

The project scheduling is all about the planning of the project completion steps. The progress of project work is as shown in the following Table:

Work to be done	Start Date	End Date	Duration (in days)
Group Formation and Selection of project topic.	01-07-2014	31-07-2014	30
Getting familiarized with the project domain	01-08-2014	14-08-2014	13
Literature Survey	15-08-2014	30-08-2014	15
Data Gathering	01-09-2014	15-09-2014	14
Manual Analysis	16-09-2014	17-10-2014	31
Coding for Sequential code	18-10-2014	02-01-2015	76
Coding for Parallel code	03-01-2015	03-03-2015	59
GUI Design	04-03-2015	15-03-2015	11
Testing	16-03-2015	01-04-2015	16
Documentation	02-04-2015	15-04-2015	13

Figure 5.1: Time Chart

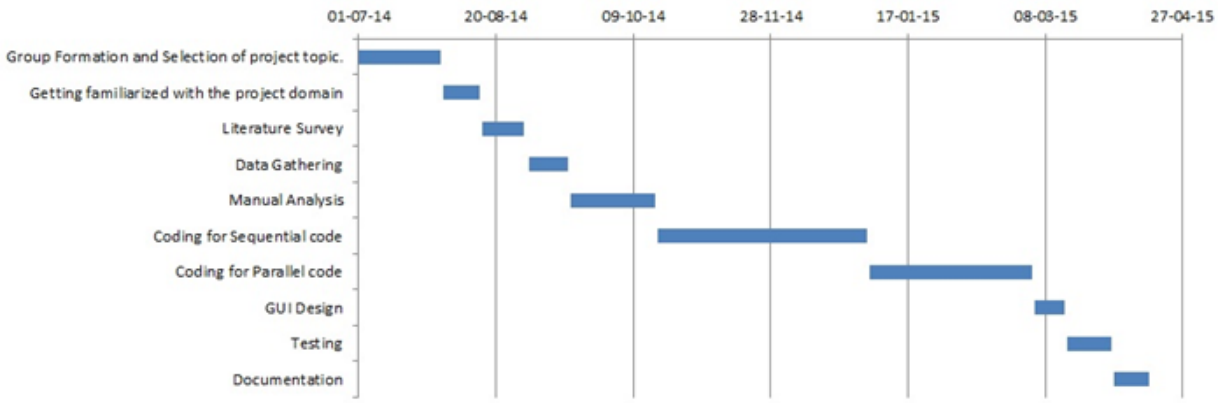

Figure 5.2: Time Line Chart

Chapter 6

Implementation

6.1 Coding

Baseline Sequential Implementation

For comparison and correctness, a sequential implementation of seam carving in C++ has been written. A Driver class will utilize Image , Energy, and SeamCarver classes to load an image, shrink it horizontally, and output the shrunk image to a file.

The first step of the sequential implementation is to load the image from a file. Images are stored as bitmaps, so they can be considered as a two dimensional array of pixels. The image is loaded and saved in the Image class. The width and height are also identified as properties. In the next step, the energy of each pixel is computed in the Energy class. A gradient calculation is used for energy. A difference between two pixels is calculated as the sum of squared differences between the RGB components. The gradient of any given pixel is equal to the sum of its difference from its bottom and its right neighbors.

After computing the gradients, the next step is to identify the vertical path of minimum importance through the image. By removing the pixels in this path, the image can be shrunk horizontally. The identification and removal of this seam occurs in the SeamCarver class. In fact, this class can remove a variable number of seams from the image, as instructed by the Driver.

The minimum vertical path is computed as follows. First the gradient table is converted to a minimum cost table. The minimum cost of any given pixel is equal to its gradient value added to the minimum of the gradient values of the three adjacent pixels from the above row. The creation of the minimum cost naturally proceeds row

by row from top to the bottom of the image. At the conclusion of the creation of the minimum cost table, the bottom pixel of the minimum seam is simply the pixel in the bottom row with the minimum cost value. After this pixel is identified, the seam can be built by tracing up through the image. The minimum of the three adjacent pixels from the above row is also a member of the minimum seam.

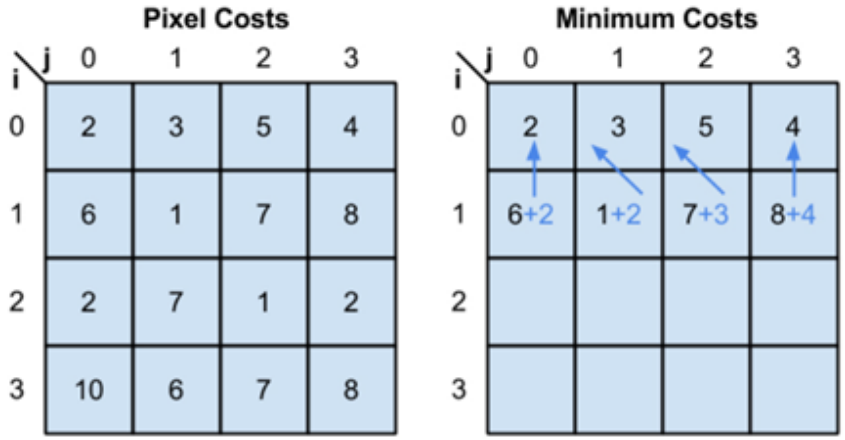

Figure 6.1: Optimal seam path calculation

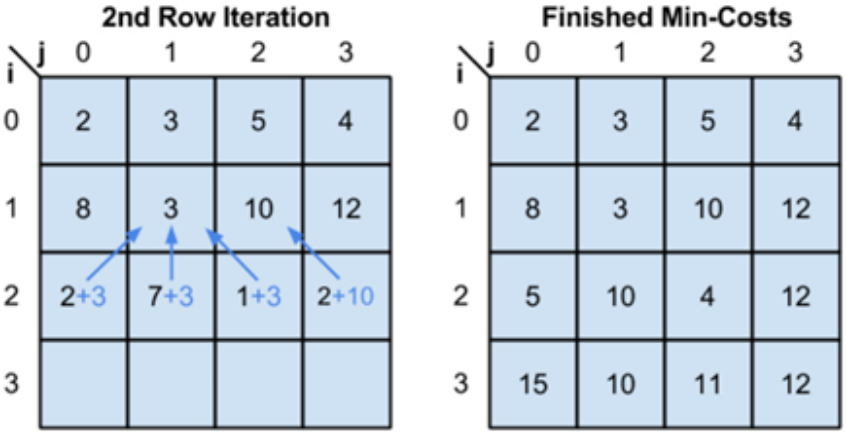

Figure 6.2: Optimal seam path calculation (contd.)

It is worth noting that this implementation is an example of bottom up dynamic programming. The minimum cost of each location in the image is computed only once and used by multiple pixels in the row below. This is also a good example of the use of backtracking to construct the solution to a problem.

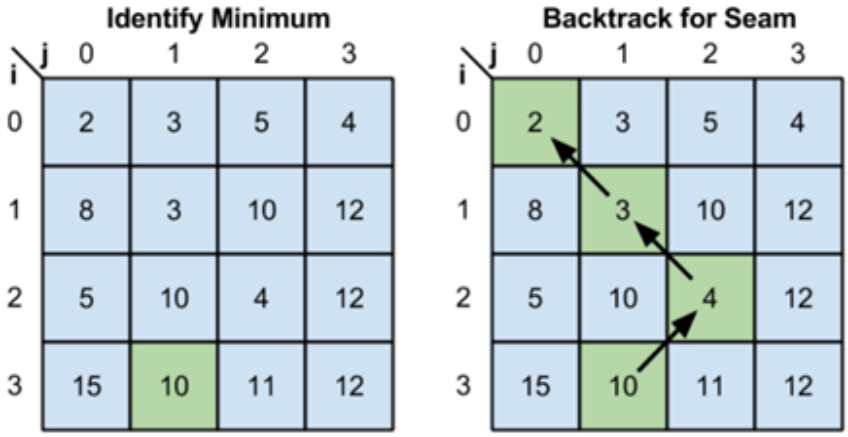

Figure 6.3: Optimal seam path calculation (contd.)

Finally, the resized image is saved to a file utilizing a method in the Image class. This is the conclusion of the algorithm.

Optimized GPU Implementation

GPU Optimization 1: Compute Energies on the Kernel

In the first optimization, the CUDA kernel is utilized to compute the energy of each pixel. A single thread is responsible for computing the energy of each specific pixel. The kernel function is called with 32 x 32 thread blocks. Enough blocks will be created to provide a thread for each pixel in the image.

To compute the energy of a pixel, the pixel values from below and to the right must be fetched and compared to the value of the current pixel. To avoid costly DRAM operations when fetching the pixels, each pixel will be loaded into shared memory by the thread that will be responsible for computing its energy. After synchronizing the threads, each thread can obtain the values it needs from the shared memory array. Boundary checks are implemented for threads on the edge of the block that may need to access neighboring pixels through DRAM.

After computing the energies, the resulting array is copied back from device memory to host memory.

GPU Optimization 2: Compute Minimum Cost Table on the Kernel

Another kernel function will be used to compute the minimum cost table. The minimum cost table is a representation of the cost of taking potential paths through an image. The minimum cost of each pixel is equal to its energy added to the minimum of the costs of the three neighboring above pixels. This is naturally an operation that occurs row by row.

In the call to this kernel function, threads will be created for each column. The minimum cost table will be created row by row, with each thread computing the minimum cost for one pixel in the row. Since each row of the minimum cost table depends on the above row, the threads will be synchronized after each row. The simplicity of this synchronization is a significant advantage of using the GPU.

While each row of the minimum cost table will be stored in DRAM, the previous row of the minimum cost table will be maintained in a shared memory vector array. This will ensure that, when using the above row to compute the next minimum cost, fetches are from fast shared memory rather than slow DRAM.

After the computation is complete, the resulting array is copied from device memory to host memory.

GPU Optimization 3: Performing a Minimum Reduction on the Kernel

After computing the minimum cost table, the minimum value in the bottom row of the table must be located. Starting at this pixel, backtracking will be used to discover the minimum seam in the image. This is the seam that will be removed.

Instead of searching for the minimum linearly, a minimum reduction will be performed on the GPU to identify the bottom of the minimum seam. The bottom row is stored in shared memory on GPU to ensure that the operations necessary for the reduction occur without costly fetches to DRAM. It is important to note that this reduction must track both the minimum value and the index of that minimum value.

Once the minimum index has been determined, it is copied back to the host for use in backtracking the minimum seam.

6.2 GUI Design

The GUI design comprises of a simple interface which uses browsing of the desired image, selecting the number of seams to be carved and selection of the type of execution to be performed which is either sequential or parallel.

6.3 Implementation Constraints And Dependencies

Assumptions:

- The numbers of pixels to be removed are less than the actual size of the image.
- CUDA powered NVidia GPU available for parallel execution.
- Carving of only vertical seams is being considered.

Constraints:

- As the algorithm is content aware the amount of resizing that can be done depends on the number of important objects within the input image.

6.4 Algorithm

1) We start with an image such as:

Figure 6.4: Input image

2) We then calculate the weight/density/energy of each pixel. This can be done by various algorithms: gradient magnitude, entropy, visual saliency, eye-gaze movement. In this example gradient magnitude gives 'satisfactory results.'

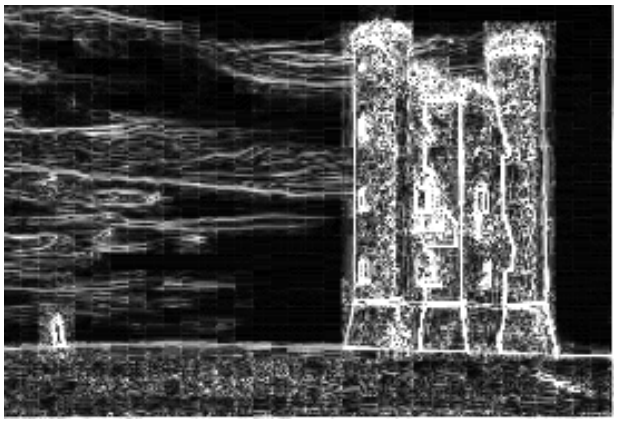

Figure 6.5: Calculation of Energy Gradient

3) After we have the energy of the image, we generate a list of seams. Seams are ranked by energy, with low energy seams being of least importance to the content of the image. We can choose to calculate seams via the dynamic programming approach.

Seams shown with the energy function:
(Notice that the seams pass through area having leaast energy i.e. dark area)

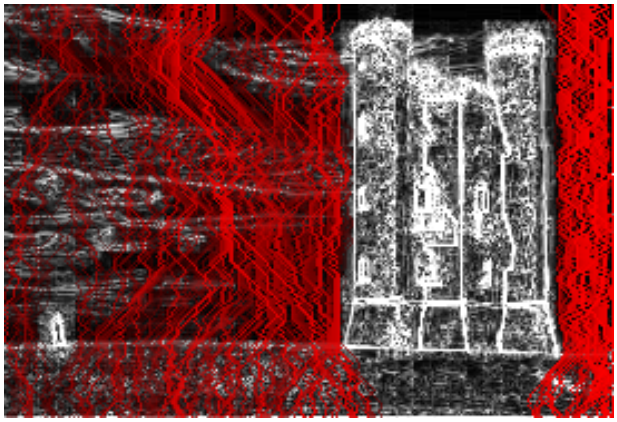

Figure 6.6: Seam paths with energy distribution

Seams shown with the original image:

Figure 6.7: Seam paths on original image

4) We then remove the seams from the image, reducing the size of the image as a result:

Figure 6.8: Seam carved image

Chapter 7

Screen-shots of Project

7.1 The Graphical User Interface

Given below in Figure 7.1 is the main interface of our application the first textbox is to select the image path either by browsing or entering manually. The second text box is for accepting the number of seams that are to be carved manually in the form of an integer number.

Figure 7.1: Screen 1

Figure 7.2 shows the working of error checking in case invalid seam number is specified.

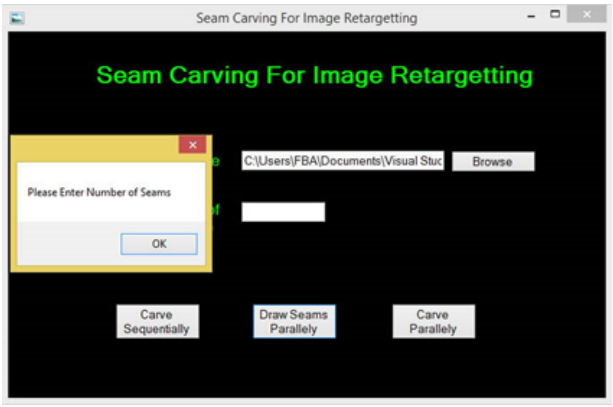

Figure 7.2: Screen 2

Figure 7.3 shows the working of error checking in case invalid image path is specified.

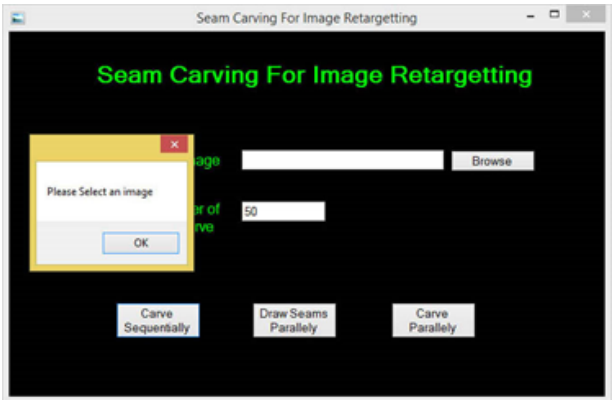

Figure 7.3: Screen 3

Figure 7.4 shows the time of execution after parallel execution is completed

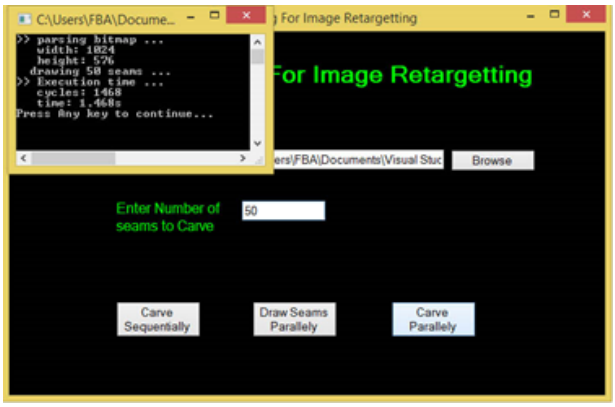

Figure 7.4: Screen 4

Figure 7.5 shows the image browsing screen.

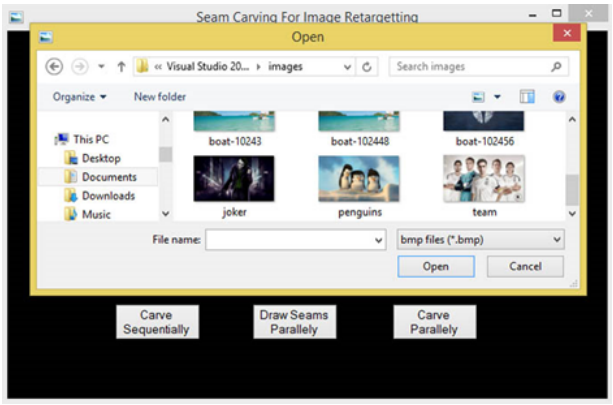

Figure 7.5: Screen 5

Figure 7.6 shows the time of execution after sequential execution is completed

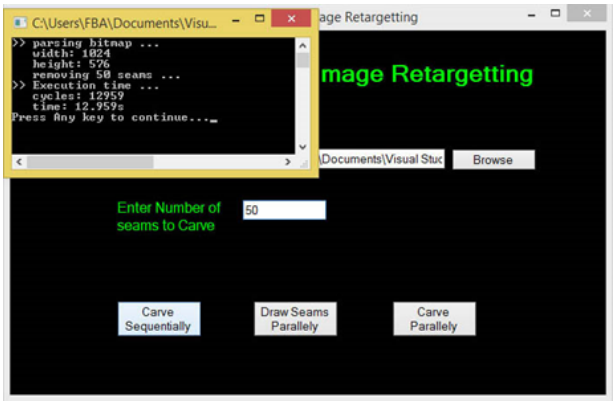

Figure 7.6: Screen 6

Chapter 8

Testing

This test plan for the Seam carving application and deals with the following objectives:

- Define the activities required to be prepared and conduct unit, system and acceptance testing.
- Communicate to all responsible parties, the test strategy.
- Communicate to all responsible parties, the test results.

8.1 Unit Test Plan, Test Results

The principle goal for unit testing is to ensure that each individual software unit is functioning according to its specification. This test will test the smallest individual operation that can be done in project. This tests the GUI as well as the seam carving system.

Test Cases:-

Case 1: GUI behavior

Step No.	Action	Input Required	Expected	Result
1	Browse valid Image	Image path	Path displayed in Textbox	Pass
2	Browse Invalid Image	BLANK or Invalid path	Error message displayed	Pass
3	Enter Number of seams	Integer number	Number entered in textbox	Pass
4	Enter Invalid data for Number of seams	Floating point, BLANK, alphabet or other invalid input	Error message displayed	Pass
5	Click on Carve Sequential	Valid image and number of seams	Carved image displayed with time of execution	Output generated
6	Click on Carve Parallely	Valid image and number of seams	Carved image displayed with time of execution	Output Generated
7	Click on Draw Seams Parallely	Valid image and number of seams	Seam paths marked image displayed with time of execution	Output Generated
8	Click on Close		Close the GUI	Pass

Figure 8.1: GUI Testcases

Case 2: Application behavior

Step No.	Action	Input Required	Expected	Result
1	Number of seams entered exceeds image width	Valid image and invalid number of seams	No output generated	BLANK image rendered
2	Carving parallely in absence of GPU/CUDA	Valid image and valid number of seams	Error message displayed	Pass

Figure 8.2: Application Testcases

8.2 System Test Plan, Test results

System testing is the testing of behavior of a complete and fully integrated software product based on the software requirements specification (SRS) document. In main focus of this testing is to evaluate Business / Functional / End-user requirements. This is black box type of testing where external working of the software is evaluated with the help of requirement documents it is totally based on Users point of view. For this type of testing do not required knowledge of internal design or structure or code.

8.3 Summary of Testing

Experiments On different Images:

Original Energy Map Seams Final

Figure 8.3: Test cases

Chapter 9

Result Analysis

The seam carving application is evaluated based on the amount of speed up achieved for a particular image, following is the image which we use for our result analysis as well as given below are the results for various computations done on this image.

Figure 9.1: Input image (1600x900)

Figure 9.2: 50 Seams Highlighted

Figure 9.3: 1000 Seams Removed

The following figures represent the execution timings for the image of size 1600x900 pixels:

No of seams	Sequential	Parallel
50	43.596	7.091
100	72.394	11.603
200	161.998	20.652
500	345.105	44.819
1000	548.305	76.117

Figure 9.4: Execution time Sequential vs. parallel

The line graph showing the difference in speeds of execution is as follows:

Figure 9.5: Difference in exection Sequential vs. Parallel

The following figures represent the execution timings for an image of size 500x500 pixels, the succeeding images are the outputs we obtain from removing n seams each where n is the number of seams in the table below.

No of seams	Sequential	Parallel
50	7.859	2.35
100	14.173	3.761
200	23.876	5.899
300	31.046	7.419

Figure 9.6: Execution time Sequential vs parallel

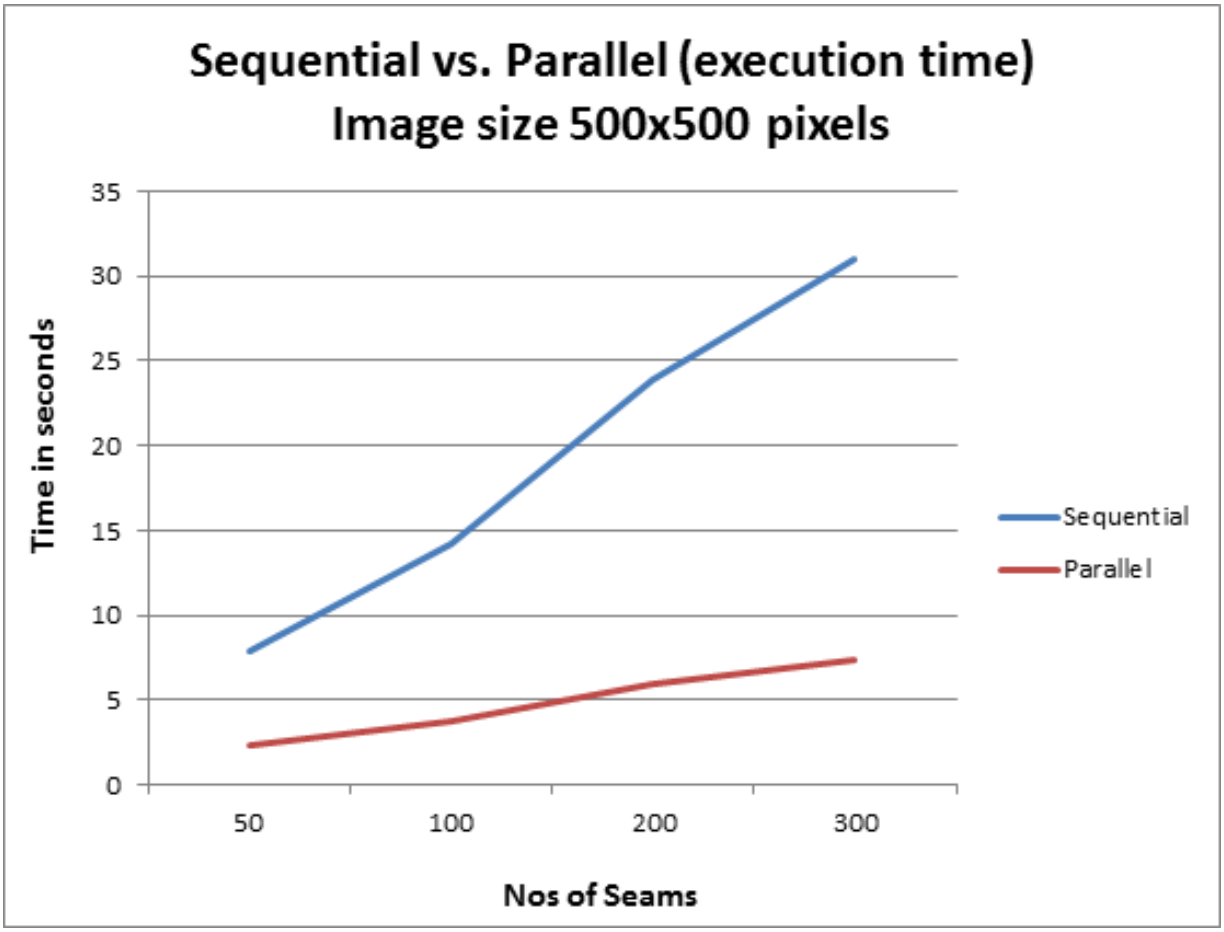

Figure 9.7: Difference in exection Sequential vs. Parallel

Chapter 10

Conclusion , Future Scope and Industrial applications

10.1 Conclusion

This project aimed at showing the difference in execution speeds between execution of the image processing algorithm seam carving on a traditional CPU and a modern GPU. The development and implementation of this system has given us a great satisfaction. Through our efforts, we have incorporated into the system several features such as User-friendly interface, Scalability, Performance, Testability, and Usability.

Appropriate User interface is designed in Visual Studio. It provides the user to browse a particular image as well as enter the number of seams that must be carved, at the same time it provides the user with 2 separate functions either to use CPU or the GPU for execution.

Usability is the ease of use and ability to learn. System is made very user friendly using interactive GUI.

The primary objective of this system was to observe speed up in execution for parallel system as we can see in Figure 9.5 and Figure 9.7 that the line graph for sequential is more or less linear in nature whereas that for parallel execution is somewhat exponential, this also tells us that as the amount of work to be done increases more of it can be done in parallel and so more amount of speed up can be achieved in execution.

10.2 Future Scope

The future is bright for Parallel Computing as more and more processing is required in less time ,also for seam carving which is image processing algorithm required huge amount of work to be done so by using parallel processing we can do this efficently.The future scope for such aids cannot be wholly enumerated, but to enlist a few, here are what this technology could contain in the future:

- Seam insertion.
- Object removal.
- Video retargeting.
- Real-time implementation of the above mentioned content aware features as well as seam carving on TEGRA K1 (192-core) powered mobile devices.

10.3 Industrial applications

Accelerated Seam-Carving can be used to retarget statistical data for devices of different resolutions. Also, seam carving can be used for the following applications:-

- In Adobe Photoshop (content aware resizing).
- Statistical data representation for different resolution devices.
- In computer gaming industry.
- In digital and computational photography.

References

[1] *Seam Carving for Content-Aware Image Resizing*: Shai Avidan and Ariel Shamir. , The Interdisciplinary Center and MERL.

[2] *Run-time Image and Video Resizing Using CUDA-enabled GPUs*: Ronald Duarte and Resit Sendag. , Department of Electrical and Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI

[3] *Context-Aware Saliency Detection*: Stas Goferman, Lihi Zelnik-Manor and Ayellet Tal.

[4] *Detection of Seam Carving and Localization of Seam Insertions in Digital Images*: Anindya Sarkar, Lakshmanan Nataraj and B. S. Manjunath , Vision Research Laboratory University of California, Santa Barbara Santa Barbara, CA 93106

[5] *Context-Aware Saliency Detection*: Michael Rubinstein, Ariel Shamir and Shai Avidan. , ACM Transactions on Graphics, Vol. 27, No. 3, Article 16, Publication date: August 2008.

[6] *Optimized Image Resizing Using Seam Carving and Scaling*: Weiming Dong, Ning Zhou, Jean-Claude Paul and Xiaopeng Zhang , ACM Transactions on Graphics 29, 5 (2009) 10 p.

[7] *A Comprehensive Performance Comparison of CUDA and OpenCL*: Jianbin Fang, Ana Lucia Varbanescu and Henk Sips, Parallel and Distributed Systems Group Delft University of Technology Delft, the Netherlands.

[8] CUDA Toolkit Documentation `http://docs.Nvidia.com/cuda/cuda-c-programmingguide/#axzz3GnJjIKFk`

[9] `http://en.wikipedia.org/wiki/Seam_carving`

[10] `http://en.wikipedia.org/wiki/CUDA`