



M060072581



NORTH-WEST UNIVERSITY
YUNIBESITHI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT

Graphics Processing Unit (GPU) based Optimized Linear Predictive Coding (LPC) Feature Extraction Algorithm for Automatic Speech Recognition using Compute Unified Device Architecture (CUDA)

Kgotlaetsile Mathews Modieginiane

Student No: 20988540

North West University

(Mafikeng Campus)

DISSERTATION

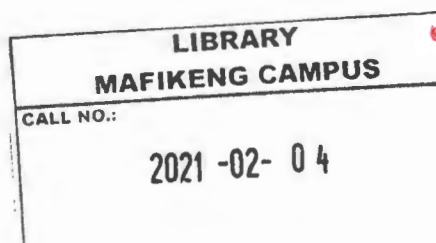
Submitted in fulfillment for the requirements for the degree of MSc in Computer Science.

School of Mathematical and Physical Sciences.

Department of Computer Science

Supervisor: Dr. Naison Gasela

Co-Supervisor: Dr. Zenzo Polite Ncube



APPROVAL

This research work has been submitted for examination with my approval as the candidate's university supervisor.

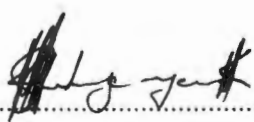
Signature:

Date :

Dr. Naison Gasela

DECLARATION

I **Kgotlaetsile Mathews Modieginnyane** hereby declare that the research work entitled “**GPU based Optimized Linear Predictive Coding (LPC) Feature Extraction Algorithm for Automatic Speech Recognition using CUDA**” is entirely my own work, except where acknowledged, and that it has never been submitted before to any other university or institution of higher learning for the award of a degree.

Signature: 

Date : 29/SEP/2014

Kgotlaetsile Mathews Modieginnyane

ACKNOWLEDGEMENTS

This work would not have been possible without the help, support and patience of my principal supervisor, Dr. Naison Gasela (North West University). The good advice, support and kindness of my co-supervisor, Dr. Zenzo Polite Ncube (Sol Plaatje University), have been priceless throughout my research project, for which I am exceptionally thankful. I would also like to permit my dearest appreciation to Miss. Babedi Betty Letswamotse for her substantial encouragement and support throughout my work.

I would like to acknowledge the financial, academic and technical support of the North West University for all the equipment they have provided, and the Department of Computer Science for their support and assistance since the start of my postgraduate studies.

Again it would not have been possible for me to do this research project without the blessings from GOD and all the support from the kindest people around me, to only some of whom it is possible to give particular mention here. Above all, I would like to thank my mother Jane Modieginnyane, brothers and sisters, who have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice.

ABSTRACT

Recent studies have shown a lot of interest in the field of Automatic Speech Recognition (ASR), and its application. ASR is a technology/process of taking spoken words as input through a speech recognition system or application, and converting/translating them into written text as output. Whilst the study of ASR has attracted a lot of research lately, accuracy in the field of speech recognition remains a great challenge, as relevant features of speech need to be extracted for processing by the speech recognition system. Of particular concern is feature extraction as the most critical phase of automatic speech recognition. This is the process of obtaining the most relevant information from the original (i.e., input speech) data and representing that information in a lower data rate. ASR systems must be accurate in their processes of recognizing speech. In that regard, different approaches as an effort to improve the accuracy of ASR systems exist.

This work implemented an optimized Linear Predictive Coding (LPC) feature extraction technique to acquire efficient extraction of relevant features during this critical phase. The algorithm was implemented on a Graphics Processing Unit (GPU) integrated system using a Compute Unified Device Architecture (CUDA). Experimental results have shown improvement by this version of the LPC algorithm. Achieved results added up to 10 per cent overall performance improvement in reference to the achieved original LPC results, on which the CPU optimized LPC brought in 6 per cent performance improvement.

Key Word: Automatic Speech Recognition, Linear Predictive Coding, GPU, CPU, CUDA

TABLE OF CONTENTS

LIST OF FIGURES	ix
LIST OF TABLES	x
LIST OF EQUATIONS	x
LIST OF ACRONYMS	xi
CHAPTER 1: INTRODUCTION	1
Introduction to the Research	1
1.1 Automatic Speech Recognition.....	1
1.2 Problem Statement and Substantiation	3
1.2.1 Problem Statement	3
1.2.2 Substantiation.....	3
1.3 Research Questions	3
1.4 Research Goal and Objectives	4
1.4.1 Research Goal	4
1.4.2 Research Objectives.....	4
1.5 Expected Results.....	4
1.6 Contribution of the Study.....	4
1.7 Summary of Dissertation	4
CHAPTER 2: THEORETICAL BACKGROUND	6
2.1 Introduction: ASR Systems.....	6
2.2 GPU Systems and the CUDA Architecture in ASR.....	6
2.2.1 GPU Systems	6
2.2.2 The CUDA Architecture	8
2.3 ASR Systems and Approaches.....	9
2.3.1 ASR Using ANNs	10
2.3.2 ASR Using CUDA GPUs	12
2.4 Feature Extraction and Feature Extraction Techniques	13
2.4.1 Feature Extraction	13
2.4.2 Feature Extraction Techniques.....	14

2.4.2.1	Mel Frequency Cepstral Coefficients (MFCC).....	15
2.4.2.2	LFCC Speech Features (LFCC-FB40).....	15
2.4.2.3	Linear Predictive Coding (LPC)	16
2.4.2.3.1	The LPC Algorithm	16
2.5	Feature Extraction Performance Analysis.....	17
CHAPTER 3: LITERATURE REVIEW		19
3.1	Introduction.....	19
3.2	Components of an ASR System.....	21
3.2.1.	Speech Signal Acquisition	22
3.2.2.	Feature Extraction Concept.....	22
3.2.3.	Acoustic Model	22
3.2.4.	Language Model	22
3.2.5.	Lexical Model	23
3.2.6.	Recognition Process.....	23
3.3	Hidden Markov Model (HMM) for ASR.....	24
3.4	Critical Analysis and Discussion	26
3.5	Review Conclusions.....	27
CHAPTER 4: METHODOLOGY AND EXPERIMENTAL SETUP		29
4.1	Introduction.....	29
4.2	Outline of the Methodology and System Setup	29
4.2.1.	LPC System Survey	29
4.2.2.	Experimental Setup	30
4.2.3.	Validation of Results: Proof of Concept	30
4.3	Details of Used Architectures and Developmental Tools	30
4.3.1.	Architecture Used	30
4.3.1.1.	Architecture Description	30
4.3.1.2.	The CPU Core.....	31
4.3.1.3.	The GPU Core.....	32
4.3.2.	Developmental Tools	33
4.3.2.1	CUDA Developmental Model	33
4.3.2.2	Speech Corpus	33

4.3.2.3	The HTK	33
4.4	System Testing.....	34
CHAPTER 5: RESULTS AND INTERPRETATIONS.....		35
5.1	Introduction.....	35
5.2	LPC Speech Feature Extraction Performance Survey	35
5.3	Methods for Computing LPC Speech Feature Extraction.....	37
5.3.1.	The Autocorrelation Method.....	37
5.3.2.	The Covariance Method.....	38
5.3.3.	The Levinson Durbin Algorithm Method	40
5.4	CPU and GPU Based LPC Implementation.....	40
5.4.1.	CPU Based LPC Implementation	41
5.4.1.1.	CPU Processed LPC Speech Waveform and Power Spectrum	42
5.4.1.1.1.	CPU Processed Speech Waveform	42
5.4.1.1.2.	CPU Processed Power Spectrum	42
5.4.2.	GPU Based LPC Implementation	43
5.4.2.1.	GPU Processed LPC Speech Waveform and Power Spectrum.....	45
5.4.2.1.1.	GPU Processed Speech Waveform	45
5.4.2.1.2.	GPU Processed Power Spectrum.....	45
5.5	PSNR (Peak Signal to Noise Ratio) Signal Analysis.....	47
5.6	Analysis and Interpretation of Results	49
CHAPTER 6: SUMMARY AND CONCLUDING REMARKS.....		52
6.1	Introduction.....	52
6.2	Proposed Goal and Objectives	52
6.2.1.	Proposed Goal.....	52
6.2.2.	Proposed Objectives.....	52
6.3	Analysis of Research Achievements.....	52
6.4	Research Challenges	53
6.5	Future Work.....	53
6.6	Summary of the Research	54
REFERENCES.....		55
BIBLIOGRAPHY		59

APPENDICES63

Appendix A.....63

Appendix B65

Appendix C67

LIST OF FIGURES

Figure 1.1: Major Components of an ASR System [2].....	1
Figure 2.1: Architecture of the GeForce GTX 8800 [7].	7
Figure 2.2: CUDA Architecture.....	9
Figure 2.3: Overview of ANN Speech Recognition Process.....	11
Figure 2.4: General processes of feature extraction.	14
Figure 2.5: Stages of the MFCC feature extraction technique [16].	15
Figure 2.6: The LPC processor [19].	16
Figure 3.1: Speech recognition models [28].	24
Figure 3.2: HMM for an isolated speech recognition system [29].	25
Figure 3.3: Network of connected or continuous word recognition [29].	26
Figure 4.1: Heterogeneous accelerated processing system [AMD Developer Central].	31
Figure 5.1: Original LPC Speech Signal [31].	36
Figure 5.2: Voice Excited LPC Speech Signal Using DCT [31].	36
Figure 5.3: Voice Excited LPC Speech Signal non-DCT [31].	37
Figure 5.4: CPU Optimized LPC Processes for reading and loading speech samples.	41
Figure 5.5: CPU Processed LPC Speech Waveform	42
Figure 5.6: CPU Power Spectrum at Specified Frequencies.	43
Figure 5.7: GPU Optimized LPC Processes for reading and loading speech samples	44
Figure 5.8: GPU Processed LPC Speech Waveform	45
Figure 5.9: GPU Power Spectrum at Specified Frequencies	46
Figure 5.10: GPU Runtime Graph for Original CPU, Optimized CPU and Optimized GPU.....	49

LIST OF TABLES

Table 4.1: Essential CPU Core Specifications..... 32

Table 4.2: Essential GeForce GTX 8800 Specifications. 32

Table 5.1: Performance Table for the Used LPC Implementations..... 46

Table 5.2: PSNR Values for Speech Sample A. 47

Table 5.3: PSNR Values for Speech Sample B. 48

Table 5.4: Original CPU, CPU Optimized and GPU Optimized LPC Speedup..... 48

LIST OF EQUATIONS

Equation 1: The LPC Algorithm Equation. 17

Equation 2: Error Prediction Equation..... 17

Equation 3: Word Probability Equation..... 23

Equation 4: Word Likelihood Summation Equation. 25

Equation 5: PSNR Equation..... 47

Equation 6: Mean Square Error Equation. 47

LIST OF ACRONYMS

ALU	Arithmetic Logic Unit
ASR	Automatic Speech Recognition
CD-DNN-HMM	Context-Dependent Deep-Neural-Network Hidden Markov Model
CPU	Central Processing Unit
CSR	Command Success Rate
CUDA	Compute Unified Device Architecture
DSP	Digital Signal Processing
DTW	Dynamic Time Warping
FPGA	Finite Programmable Gate Array
FST	Finite State Transducers
GB	Giga Byte
GMM-HMM	Gaussian-Mixture Model HMM
GPU	Graphics Processing Unit
HLSL	High Level Shader Language
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
ICS	Intelligent Call Steering
ISA	Instruction Set Architecture
LFCC	Linear Frequency Cepstral Coefficients
LPC	Linear Predictive Coding
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficients
PLP	Perceptual Linear Prediction
PTX	Parallel Thread Execution
RTF	Real Time Factor

SDK	System Development Kit
SLI	Scalable Link Interface
SM	Streaming Multiprocessor
SWER	Single Word Error Rate
WER	Word Error Rate

CHAPTER 1: INTRODUCTION

Introduction to the Research

This chapter discusses the general background around automatic speech recognition, the problem statement, aims and objectives of this research, the research methodology, justification of the study and a summary of the dissertation.

1.1 Automatic Speech Recognition

Speech Recognition (also called Automatic Speech Recognition (ASR)), is the technology/process of taking spoken words as input through a speech recognition system or application, and converting it into written text as output. Recent research in this field has shown that speech recognition is faced with a lot of challenges, especially with regard to achieving great results in speech processing, particularly the accuracy of speech recognition [1]. Thus a lot of work is required to improve the actual processes of speech processing. This can only be achieved by developing speech recognition systems that are efficient with few errors in their processing of speech. Figure 1.1 illustrates major components of a speech recognition system.

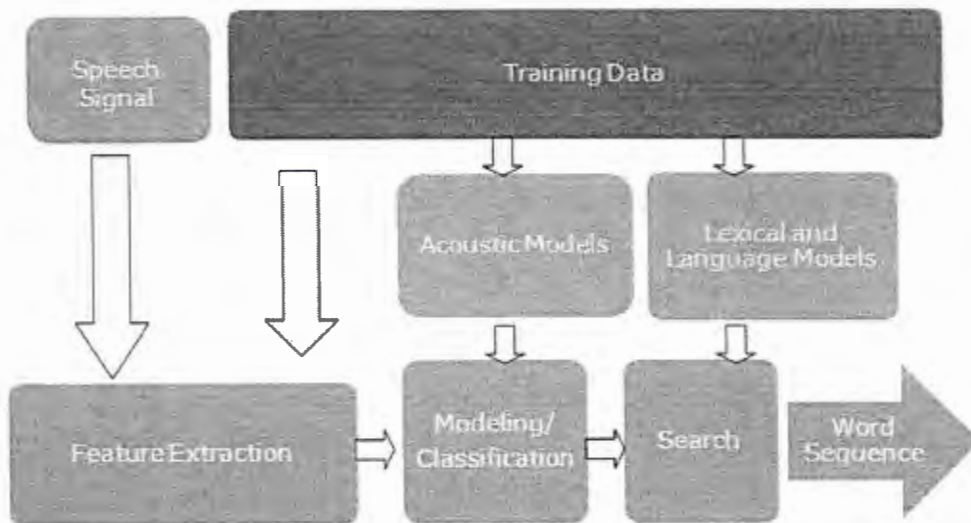


Figure 1.1: Major Components of an ASR System [2].

When speech is articulated to a speech processing system, speech features are extracted for processing by the system's speech models. The system must also be able to detect whether the

recognized audio signal is voiced or unvoiced. To learn how a phoneme sounds, the system passes a training tool of hundreds of recordings of that phoneme as well as figuring out the correct grammar of the particular phoneme. This is done by the acoustic, lexical and language model of that particular speech recognition system. Relevant searches are performed by the search algorithm as an aid to produce or match the observed features or probable word sequences.

Speech recognition is a very complex process, and requires a suitable environment for proficient and accurate speech processing, since there are a lot of factors affecting the actual speech recognition. An efficient speech recognition system must be accurate in its recognition, and be fast in its speech processing. The accuracy of a speech recognition system is principally measured using the Word Error Rate (WER), this is the ratio of word insertion, substitution, and deletion errors in a transcript to the total number of spoken words, and the speed is measured with real time factor (RTF) as in Park [3]. Other measures of the speech recognition system include; Single Word Error Rate (SWER) and Command Success Rate (CSR) [4].

In a computer based speech recognition system, the utterer speaks through a microphone, which then transmits the utterance into the recognition system for the actual processing. Thus, the utterer must speak as clearly and as close to the microphone as possible for efficient utterance recording and processing, otherwise the utterance might not be heard or for that matter be wrongfully heard; and that would yield unexpected results. Other factors include a crosstalk situation where there is overlapping speech by a device or as in a meeting where there are numerous people, and then speech recognition becomes difficult. In such a case, the processor has to do a lot of work in order to understand complicated words, phrases or long gasping sentences.

On the other hand, speech recognition systems must be able to handle and/or support valuable applications such as; dictation, command and control, embedded applications, telephone directory assistance, spoken database querying, medical applications, office dictation devices, and automatic voice translation into foreign languages, voice biometrics, Intelligent Call Steering (ICS), etc. This leads us to study techniques that are used in various stages of the speech recognition, particularly the speech feature extraction techniques and their effectiveness in speech recognition.

1.2 Problem Statement and Substantiation

This section describes the research problem statement and gives substantiation about what the actual study is looking for.

1.2.1 Problem Statement

An efficient speech recognition system must be accurate in its course of speech processing. This is a challenge since it is very difficult to build a speech recognition system that can closely recognise speech utterances or sound as human beings. Rather, improvements (by recent researchers, using different approaches) as in [1] are currently focused on building ASR systems that have minimal speech recognition errors, by improving different phases of the speech recognition system.

In order to achieve better accuracy, a lot of factors need to be taken into consideration, such as background noise, efficient speech feature extraction and the actual speech processing (either by hardware or software). Also, successful improvements on one phase of the speech recognition system should not compromise other aspects such as processing speed and quality of the expected output.

1.2.2 Substantiation

Speech technologies are used in many fields that are communication based, either in telecommunication systems or application-wise in embedded systems. These systems are used in areas such as those stated in the Introduction Section. Upon success of optimizing the proposed algorithm on a parallel architecture, an efficient speech recognition system will be achieved.

1.3 Research Questions

The research questions that this research work proposes to answer are:

RQ1: How can CUDA C++ code for the LPC algorithm be developed for efficient feature extraction?

RQ2: How can the LPC feature extraction algorithm be optimized for better performance in parallel architectures?

RQ3: What measures must be made as proof of concept that the then optimized LPC algorithm will be more efficient than the original LPC algorithm?

1.4 Research Goal and Objectives

The research goal and objectives of this research work are as follows.

1.4.1 Research Goal

The main goal of this research work is to optimize the LPC feature extraction algorithm for efficient automatic speech recognition on a GPU based platform using CUDA.

1.4.2 Research Objectives

The objectives of this research work are:

- To conduct a literature study on the GPU and CUDA based ASR technologies on different techniques.
- To optimize the LPC feature extraction technique for parallel processing by a GPU in an attempt to attain efficient extraction of relevant information features during the speech recognition process, on a CUDA based platform.
- To analyze and evaluate the performance of the proposed algorithm for efficient speech feature extraction towards better speech recognition in terms of accuracy.

1.5 Expected Results

The GPU based implementation is expected to perform better than the existing CPU based speech recognition systems.

1.6 Contribution of the Study

Achievements from our GPU based implementation positively influences the overall performance of the speech recognition system in terms of efficiency and accuracy thereby contributing to the fields of speech recognition and High Performance Computing

1.7 Summary of Dissertation

This section briefly outlines a description of what will be covered on each chapter in this research work.

Chapter 1: Introduction

This Chapter outlines the whole research, and also points out the main aim and objectives of the proposed work.

Chapter 2: Theoretical Background

This Chapter gives a theoretical background to the area of study, focusing particularly on the problem statement that leads to the proposed work. Previous achievements regarding these problems will be outlined.

Chapter 3: Literature Review

In this Chapter the literature with regard to findings and aspects that reflects in this research area will be reviewed. Various aspects and techniques for efficient automatic speech recognition in this research area will be analyzed and discussed, with a view towards motivation of the proposed work.

Chapter 4: Methodology and Experimental Setup

This Chapter describes the methodology and the actual experimental setup to be used in carrying out this research work. A list of tools to be used for the experiment will be specified.

Chapter 5: Results and Interpretations

This Chapter discusses the research results, wherein analytical interpretations of the achieved results with regard to the optimized algorithm for efficient speech recognition will be given.

Chapter 6: Summary and Concluding Remarks

This Chapter gives an overall summary of the achievements regarding the main focus of this research work, with comments and motivations on the deliverables of the proposed technique for speech recognition. Optimization issues under the proposed environment will also be discussed. Analytical conclusions on the particular study will be given.

CHAPTER 2: THEORETICAL BACKGROUND

This chapter gives a theoretical background around the area of speech recognition, with the focusing on the problem statement which leads to the proposed work, different speech recognition approaches and hardware improvements using these approaches. Previous achievements regarding these problems will be outlined.

2.1 Introduction: ASR Systems

There have been significant research advances in today's automatic speech recognition systems in achieving best speech recognition results. Since speech recognition systems work according to how they are parameterized, the act of developing accurate automatic speech recognition systems is still a challenge. These parameters explicitly define performance capabilities of these systems. Therefore careful considerations must be made regarding the actual design of these ASR systems. Some common parameters regarded for developing these systems (upon requirement specification) include isolated speech recognition and continuous speech recognition. These systems are also either speaker dependent (constructed to cater for one speaker) or speaker independent (can cater for any speaker). Again ASR systems differ on how much data they can handle.

2.2 GPU Systems and the CUDA Architecture in ASR

2.2.1. GPU Systems

GPU (Graphics Processing Unit) is NVIDIAs core for graphics processing developed by NVIDIA in 1999. GPUs were initially designed as specialized circuits with the effort to accelerate the image output to an output intended frame buffer for display. The first GPU developed was a GeForce 256, NVIDIA [5]. This GPU model could process 10 million polygons per second and had more than 22 million transistors, according to NVIDIA GeForce 256 release notes [5].

GPUs are very efficient at manipulating computer intensive tasks and are generally more effective than general-purpose Central Processing Units (CPUs) for algorithms where processing of large blocks of data is done in parallel. Today's GPUs are able to process

complex signal processes such as speech processing, artificial intelligence (such as in machine learning), computational linguistics and other core mathematical computations that require high processing capability machines.

GPUs are highly parallel programmable cores on NVIDIA developmental platforms, and offer great relevance for high performance computing. In this research work, the GPU hardware that will be used is the GeForce GTX 8800, which is based on the SLI (Scan Line Interleave) technology and comprises of 575 CUDA cores. These CUDA GPUs have parallel throughput architecture that emphasizes many concurrent threads. It is this distinction that makes them highly suitable for parallel programming [6]. Consequently parallel portions of applications called kernels are executed on the GPU.

GPUs are very efficient in the area of speech processing in that GPUs can be used to perform speech recognition using large or in some cases multiple speech models, so as to acquire high speech processing accuracy. This is due to the fact that, to obtain accurate speech recognition, larger speech models that cover various acoustic environments must be used, and broad vocabularies must be covered. Figure 2.1 illustrates the actual architecture of this GPU.

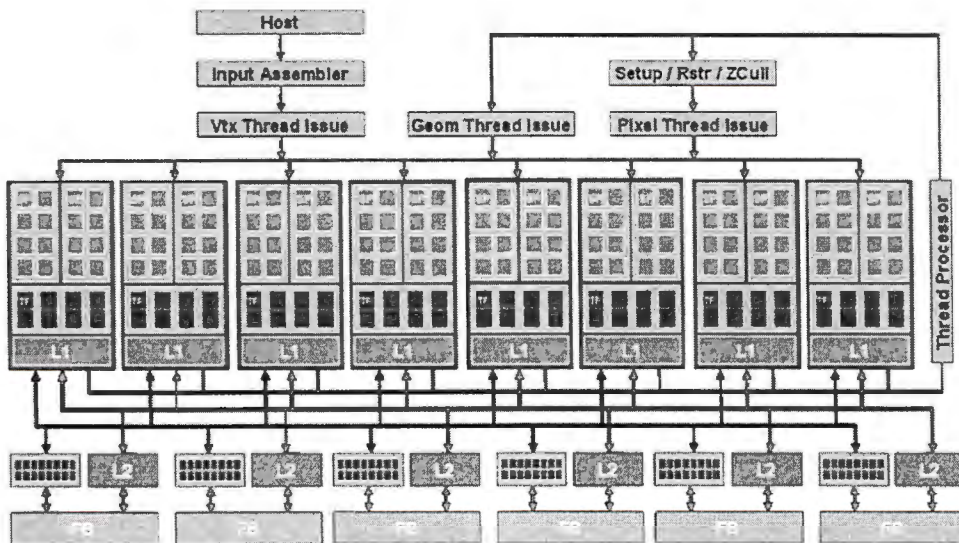


Figure 2.1: Architecture of the GeForce GTX 8800 [7].

2.2.2. The CUDA Architecture

CUDA developed by NVIDIA, is a parallel computing platform and programming model that enables efficient parallel computing performance by coupling with the processing power of the Graphics Processing Unit (GPU). The coupling effect of the GPU and CUDA module is computational and processing systematic since CUDA modules provides the GPU with effective programmable environment and computational support. In actual fact, CUDA is a scalable parallel programming model and a software environment for parallel computing.

The CUDA architecture provides developers with a way to efficiently program GPUs using minimal extensions to C/C++ environment and it is a heterogeneous serial-parallel programming model. CUDA programming is based on the data parallel processing model and exhibits great relevance for computing intensive tasks such as signal processing, computer visualization, scientific computing, etc. CUDA enables this outstanding performance through its standard APIs such as OpenCL and DirectX Compute, high level languages such as C/C++, FORTRAN, Java, Python and the Microsoft .NET framework.

CUDA exposes a fast memory region that can be shared amongst threads and allows threads in the same block to coordinate their activities using a barrier synchronization function. The CUDA shared memory architecture makes it possible for thread cooperation. Figure 2.2 illustrates the CUDA architecture with its components and brief descriptions of some components. PTX stands for *parallel thread execution*, ISA for *instruction set architecture*, and HLSL for *high level shader language*.

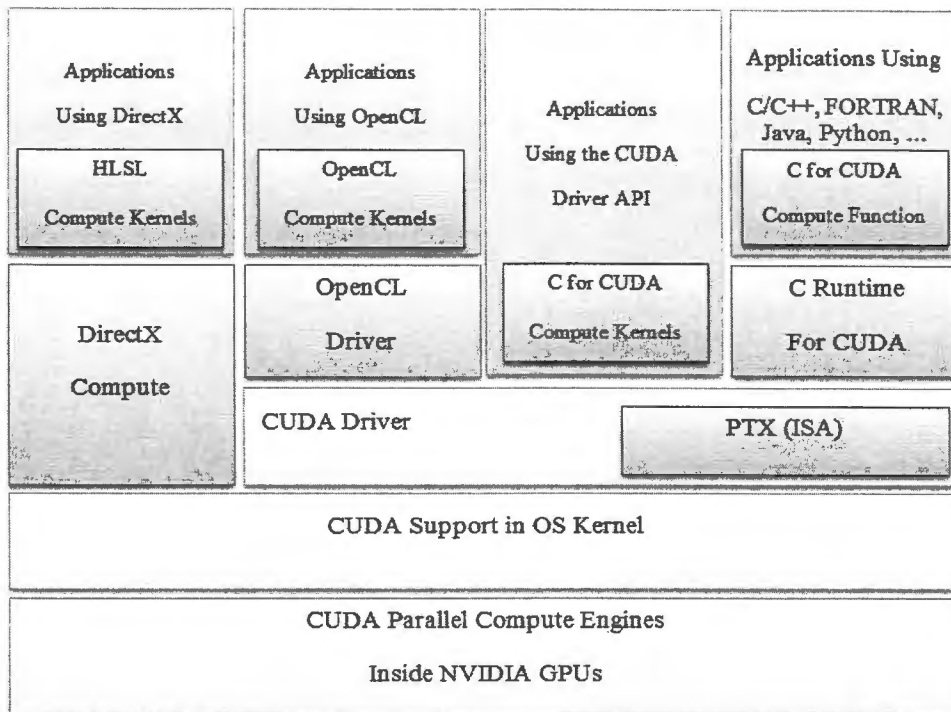


Fig. 2.2: CUDA Architecture.

2.3 ASR Systems and Approaches

Diverse approaches in the area of automatic speech recognition exist, applied also to different speech systems [8]. This is mainly because research around speech technology is aiming to achieve better results. The focus is not only on achieving these better results but also on developing speech recognition systems that are powerful enough to handle large data sets in this area. Thus striving to achieve good results in the process of speech recognition, goes with the effort of improving these systems for them to be able to do such work.

The area of automatic speech recognition has become of interest in today's research; from individuals to large companies that have some kind of either automated speech response or speech recognition systems [9]. Late in 1952 Davis et al. [10] developed a speech recognition system that could recognize only digits. This system was used to automatically recognize telephone digits when the articulated speech was from a single individual. This research by Davis et al. reported a speech recognition accuracy of about 97% and 99%. After some preliminary investigations the

system could be further improved to recognize the speech of that particular individual. However such systems needed a lot of work since they could only deal with very little data of particular digits. Conversely today's research around this area brings a lot of improvement to performance. Recent speech recognition systems are more sophisticated and complex. They can handle large amounts of data and can recognize voiced and unvoiced sound. Still the challenge of achieving high levels of speech accuracy persists since there are a lot of challenges facing this technology. The ultimate desire is to have a speech recognition system that could attain speech understanding as competently as human beings.

Recent research in speech analytic systems uses sophisticated machines or systems of approach that boost the performance of modern speech recognition systems [11], [12] , [13]. Systems of approaches around the ASR area such as those using Graphics Processing Units (GPUs) and Artificial Neural Networks (ANNs) brought a lot of improvement in acquiring better performance results. Artificial Neural Networks have been reported as being able to handle large amounts of data [14]. ANNs are also good in phoneme recognition and articulated digit recognition. GPUs on the other hand have improved the performance of speech recognition systems in terms of speech analysis and speech processing speed. The GPU architecture is mainly designed to work on a CUDA platform, which is a software abstraction GPU integrated machine. Subsections, 2.3.1 to 2.3.2 take a look at some of these different systems of approaches in the speech recognition area.

2.3.1 ASR Using ANNs

Artificial neural networks in the area of speech recognition were reported in the early 1960s, by Morgan [15]. However the study was not yet popular at that time. A research by Tebelskis [16] reported that the application approach of ANNs in automatic speech recognition can learn complex functions, generalize effectively, tolerate noise, and support parallelism. This work explored a hybrid speech recognition system called Neural Network – Hidden Markov Model (NN - HMM) wherein neural networks performed acoustic modeling, and HMMs performed temporal modeling.

Research advances in the area ASR have shown a lot of progress in terms of achieving good speech processing results. Another ANN model for automatic speech recognition known as Context-Dependent Deep Neural Network-HMM (CD-DNN-HMM) was presented by Dahl et

al. [17]. This model produced better results compared to the NN – HMM in terms of performance and processing large vocabulary speech recognition systems. This research has demonstrated that the DNN gave a better performance in terms of error reduction and processing time and is often used in the Gaussian Mixture Models (GMM) algorithm.

The notion that ANNs support parallelism, is a huge advantage to today's parallel processing machines and hybrids of both sequential and parallel processing architectures. Thus an optimization of a proficient algorithm such as DNN on a parallel processing device like a GPU could boost the general performance of a speech recognition system. Figure 2.3 illustrates the graphical overview of the speech recognition process and the work of a neural network for the processing of a voiced word "two" to its text representation. A speech waveform is divided into frames wherein feature will then be computed for spectral representations onto the context window. A neural network then categorizes these features into their phonetic representations. A likelihood matrix is now formed after the categorization. This matrix together with the associated language and pronunciation model uses a Viterbi search to determine probable uttered word(s).

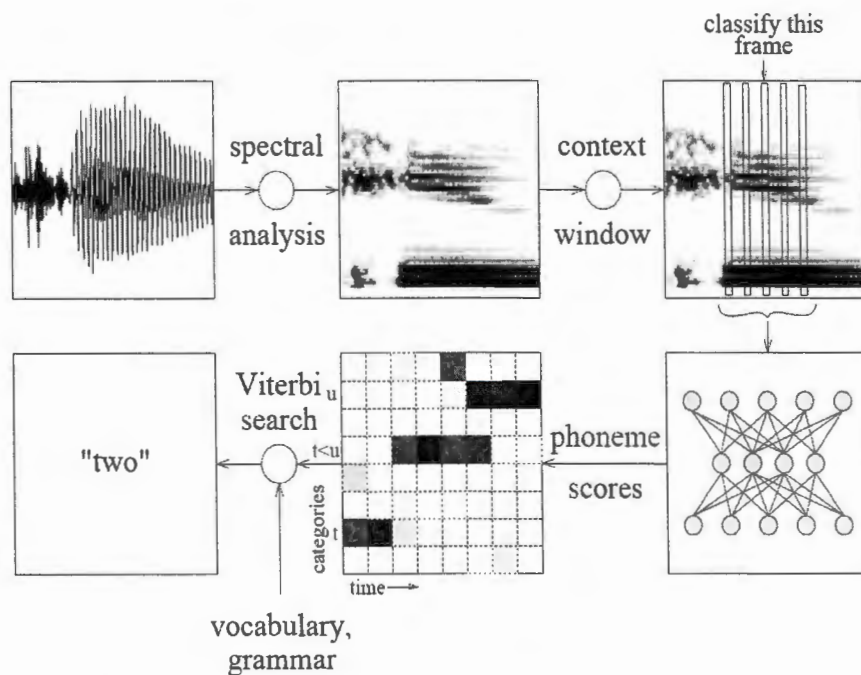


Fig. 2.3: Overview of ANN speech recognition process [18].

For best performance results, Neural Networks (NN) must be trained with a large data set so that the neural network itself has to be large. The technique of training NNs with large data sets allows them to have better speech recognition. However, it is a huge challenge for these NNs to be able to learn and identify meaningful speech recognized results due to the vast variability of speakers and different language dialects. This challenge needs high and powerful data processing systems for NNs to produce greater speech recognition results.

2.3.2 ASR Using CUDA GPUs

Another attractive technology in the area of automatic speech recognition is that of the use of Graphics Processing Units. Recent research advances of this technology in the area of speech have also shown a lot of improvement in the overall performance of speech recognition systems. Chong et al. [19] implemented a GPU optimized HMM speech recognition system. They used a GPU based system to parallelize a HMM based Viterbi search algorithm on Large-Vocabulary Continuous Speech Recognition (LVCSR) having a recognition model of 50,000 English words, with more than 500,000 word bigram transitions, and one million HMM hidden states. Their work has reported a real-time performance speedup for personal and mobile computing platforms. This is a clear indication of how much data these GPU cores can process without compromising the ultimate performance of the speech recognition system used.

GPUs are parallel processing cores that work on the CUDA (Compute Unified Device Architecture) platforms. The CUDA serves as a software abstraction to the GPU processing mechanism. Another interesting work around CUDA GPUs is that of speech indexing by the Nexiwave Company. Speech indexing is a computationally intensive task and also very expensive. However speech indexing can be efficiently processed in parallel, which makes parallel processing systems such as the GPU perfect to do this job. A talk by NVIDIA and Nexiwave's CEO Jiang [20] also indicated that the GPU will solve the cost issue of their system which is associated with indexing vast amounts of audio content quickly and accurately.

The approach of neural networks using GPUs is also a promising technology in automatic speech recognition. However it is very difficult to train neural networks to run speech processing software and produce accurate results, since these systems are required to train on

large amounts of data. As a benefit, neural networks take advantage of GPU parallel processing capabilities since they also support parallelism in data processing.

2.4 Feature Extraction and Feature Extraction Techniques

This section gives details about what feature extraction is in the area of automatic speech recognition. It also discusses some of the common techniques used at the feature extraction stage and performance analysis on each technique regarding the speech recognition process.

2.4.1 Feature Extraction

Feature extraction (in speech recognition) is the process of obtaining the most relevant information from the original (i.e., input speech) data and representing that information in a lower dimensionality space. This is the most crucial and important stage of speech recognition, as it is faced with the large variability of the speech signal. Thus, to reduce such variability; some feature extractions must be performed so as to eliminate various sources of information, such as whether the detected sound is voiced or unvoiced. In the case of speaker uttering; if the detected sound is voiced, the system eliminates the effect of the periodicity or pitch, amplitude of excitation signal and fundamental frequency etc. Otherwise the system treats any sound except the speaker's utterance, as noise.

Since the feature extraction of any speech recognition system is critical to the effect of achieving accurate speech recognition results, speech recognition systems must use efficient feature extraction techniques for even better performance. It is the overall task of the implemented chosen feature extraction technique to extract the best and relevant features from the input speech (or detected speech), towards accurate output results. If the input speech data is too large for the extraction technique used, and is supposed or found (by the feature extraction algorithm) to have redundant data, then the input speech data will be transformed into a reduced representation set of features. This process, of transforming the input data into the set of features is the actual **feature extraction**. Extracted features must contain information that can still produce the expected output results using this reduced representation set instead of the original or large speech data set.

Figure 2.4 indicates the general processes of feature extraction. This is where the first stage of the Automatic Speech Recognition, known as the Front-end analysis takes place; whereby the acoustic signal is converted into a sequence of acoustic feature vectors.

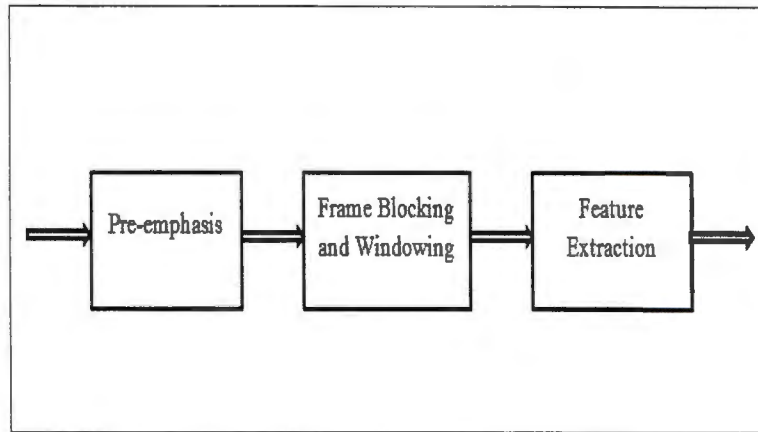


Fig. 2.4: General processes of feature extraction.

Before some speech features get extracted, the speech signal is pre-emphasised so as to flatten it and have it susceptible during the windowing stage, where speech frames will be sampled accordingly so that relevant speech features get extracted.

Section 2.4.2 gives a summary of some feature extraction techniques that have recently been used, especially in the area of speech recognition. Many of these techniques are also useful in other areas of speech processing.

2.4.2 Feature Extraction Techniques

Several feature extraction techniques exist, however they differ in their effort of extracting relevant information during the actual speech detection by the speech recognition system. Commonly used feature extraction techniques are:

- **Mel Frequency Cepstral Coefficients (MFCC)**
- **LFCC Speech Features (LFCC-FB40)**
- **Linear Predictive Coding (LPC)**

Details regarding these techniques are individually discussed in Subsection 2.4.2.1 to 2.4.2.3, and the analysis of which amongst these techniques is a better technique for efficient speech recognition will also be discussed in Section 2.5. However the focus of this work is on the LPC speech technique. Therefore this technique will be discussed in detail in Subsection 2.4.2.3.

2.4.2.1 Mel Frequency Cepstral Coefficients (MFCC)

The MFCC is considered to be a standard technique for feature extraction. It is also an efficient technique for speech recognition, except that it is very sensitive to noise due to its dependence on the spectral form, which is actually its disadvantageous aspect. The use of about 20 MFCC coefficients is common in automatic speech recognition, although 10-12 coefficients are often considered to be sufficient for speech coding as indicated by Elminir [21]. For a better understanding of the MFCC extraction technique, Figure 2.5 illustrates the stages of the MFCC.

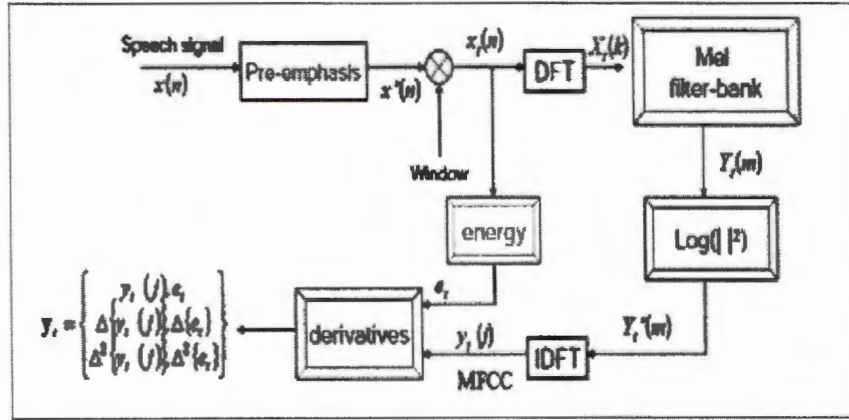


Fig. 2.5: Stages of the MFCC feature extraction technique [21].

2.4.2.2 LFCC Speech Features (LFCC-FB40)

The LFCC SPEECH FEATURES or the (Linear Frequency Cepstral Coefficients – Filter Bank 40), is also computed as the MFCC-FB40 with the only variance that the Mel-frequency warping step is avoided. Therefore, the desired frequency range is implemented by a filter-bank of 40 equal-width and equal-height linearly spaced filters. Each filter has the bandwidth of 164 Hz, of which the whole filter-bank covers a frequency range of [133, 6857] Hz. Clearly, the equal bandwidth of all filters renders unnecessary effort for normalization of the area under each filter [22].

2.4.2.3 Linear Predictive Coding (LPC)

This is the preferred technique for the speech recognition. It is one of the most powerful speech analysis techniques and is a useful method for encoding quality speech at a low bit rate. The Linear Prediction (LP) model is based on human speech production. It uses a conservative source-filter model, in which the glottal, vocal tract, and lip radiation transfer functions are integrated into one all-pole filter that simulates acoustics of the vocal tract. The LPC minimizes the sum of the squared differences between the original speech signal and the estimated speech signal over a finite duration. This can be used to give a unique set of predictor coefficients which are estimated on every frame in a period of 20 ms. [23].

Figure 2.6 illustrates the processes involved in the LPC extraction technique.

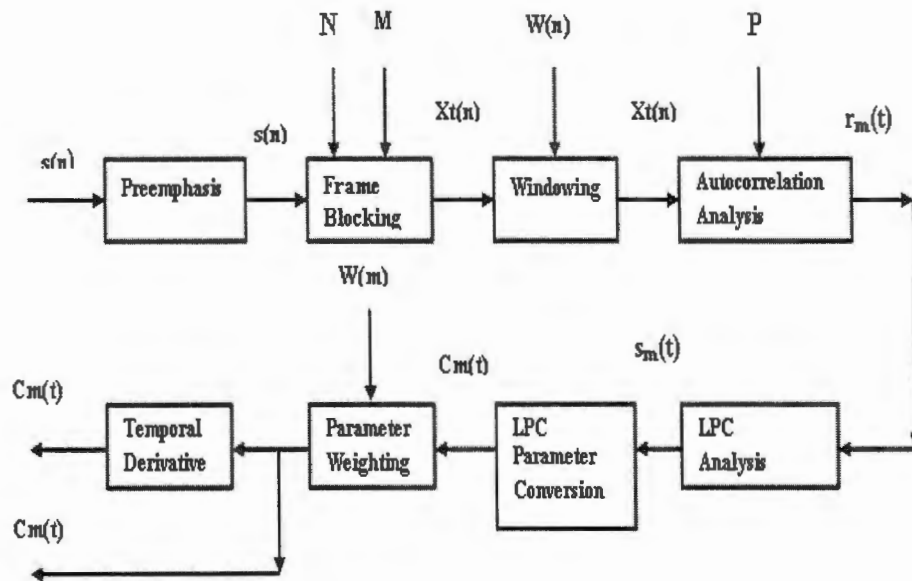


Fig. 2.6: The LPC processor [24].

2.4.2.3.1 The LPC Algorithm

Equations (1) and (2) were adapted from Rabiner and Schafer [25]. The principal idea behind the LPC is that; the current speech sample can be closely approximated as a linear combination of past samples i.e.:

$$\hat{s}[n] = \sum_{k=1}^p \alpha_k s[n-k] + e[n] \quad (1)$$

where $\hat{s}[n]$: is the linearly predicted estimate.

$\{\alpha_k\}$: are the p^{th} linear predictor coefficients.

$e[n]$: is the prediction error.

$$e[n] = s[n] - \hat{s}[n] = s[n] - \sum_{k=1}^p \alpha_k s[n-k] \quad (2)$$

Since my work is focused on optimizing the LPC feature extraction algorithm on the CUDA architecture; for optimization issues associated with the pre-emphasis stage and the error prediction equation (2), A CUDA optimization opportunity looks at:

1. Parallelizing the LPC coefficients for efficient feature matching, since LPC coefficients are run sequentially on non-CUDA systems. Thus a CUDA thread block must be applied to run the LPC coefficients parallel.
2. Minimizing the prediction error ($e[n]$) by optimizing linear predictor coefficients (α_k) to a minimum value. This will improve the speech recognition accuracy.

2.5 Feature Extraction Performance Analysis

The Linear Predictive model (LP Model) of the LPC is based on human speech production. It utilizes a conservative source-filter model, in which the glottal, vocal tract, and lip radiation transfer functions are integrated into one all-pole filter that simulates acoustics of the vocal tract. This makes the LPC an efficient technique for automatic speech recognition, in terms of its speech feature extraction capabilities.

On the other hand, both the MFCC and the LFCC-FB40 rely on the use of the Mel Cepstral Coefficients, which at times are affected by distorting signals during the recognition process of

speech. However, the LFCC-FB40 is also fairly efficient in its feature extraction process, since it does not experience the warping stage as the MFCC does.

Based on the literature study that was carried out, the LPC feature extraction technique was found to be a good technique that can yield better speech recognition performance results when applied on parallel architectures.

CHAPTER 3: LITERATURE REVIEW

This chapter reviews the literature with regard to findings and aspects of speech recognition. Various aspects and techniques for efficient automatic speech recognition is critically analyzed and discussed. Conclusions based on the insights gained from major studies towards motivation of the proposed work was also given.

3.1 Introduction

Recent technologies in the field of Automatic Speech Recognition (ASR) are advancing toward the level of human interaction. Indications have been made in recent studies around this field by exploring efficient ways of achieving the best speech processing systems. New approaches are currently being developed and modified in an effort to achieve better speech recognition results. Even though different approaches to ASR exist, they are also conducted in different environments, such as on noisy or noise free environments, which directly points to challenges faced in the area.

Of late, automatic speech recognition has become an interest to a lot of companies that deal with automated speech technologies. Companies including financial institutions, utilities, consumer products, automotive, etc. are investing a lot on embedding powerful speech technologies on to their systems. The desire is to build a speech recognition system that can learn and adjust to human capabilities and variances. Thus intensive research in this field is focused on developing more intelligent speech systems.

Speech recognition systems differ in several capabilities such as how much data they can handle, linguistic measures, speaker variances, signal distortions (by machine), capabilities of handling external noise (based on the surrounding environment), irrespective of whether one uses an isolated or continuous speech recognition system. Similarly the work of extracting relevant features from a system recognized speech is a serious challenge. This can only be achieved through the efficiency of acoustic and language models of whichever technique is used as a basis for feature extraction, together with the speech understanding of the actual speech recognition system.

Common ASR systems are implemented on FPGAs and DSPs. Trebien & de Oliveira Neto [22] implemented a digital recursive linear filter using the GPU, wherein a comparison between the said approach and an equivalent CPU-based implementation demonstrated that, when used in a

real-time audio processing system, their implemented technique supported processing of two to four times more coefficients than it did on the CPU-based implementation. Their technique also eliminated the necessity of processing the filter on the CPU, by avoiding additional memory transfers between CPU and GPU, when one wishes to use the filtering in conjunction with other processes, such as sound synthesis.

Recently, automatic speech recognition using GPUs has produced good results with regard to the achieved improvements in speech recognition accuracy. Researchers such as in [19] explored the opportunities for parallelizing a more complex speech recognition algorithm. They implemented a Hidden Markov Model (HMM) based Viterbi search algorithm, which is typically suited for large vocabulary continuous speech recognition (LVCSR). Their GPU version proved to be 9 times faster than their CPU version.

Recent advances in the field of automatic speech recognition have shown that speech sound waves have complex distributions. Thus speech recognition systems use lower dimensionality speech encoders such as MFCC, LPC or PLP to avoid these complexities. Atal [26] talks about the fundamental concepts of Linear Predictive Coding, which greatly simplified the estimation of the vocal tract response from speech waveforms. Since then, basic ideas of applying fundamental pattern recognition technology to speech recognition, based on LPC methods, were proposed as in [27].

Modern speech recognition systems are efficient in their effort to reduce speech recognition errors, and to improve the accuracy of the recognized speech. Lately, deep neural networks are a common attraction to be used in ASR because of their capabilities of processing large volumes of data.

Researchers such as those in [28] used a Context-Dependent Deep-Neural-Network Hidden Markov Model (CD-DNN-HMM) which has shown great performance results for ASR, wherein large amounts of data were processed, and hence improved the ASR performance. Research has shown that this acoustic modeling technique outperforms the Gaussian-Mixture Model based HMMs (GMM-HMMs) on several automatic speech recognition tasks [17]. Even though these ASR systems are trained on large quantities of data, mismatches between the training and testing conditions are still experienced due to speaker and environment differences. As much as with

systems built upon statistical models, CD-DNN-HMM based ASR systems may fail to produce great performance when tested under mismatched conditions.

Poli *et al.* [29] applied the Dynamic Time Warping algorithm (DTW) to perform voice password identification, and they reported that it is possible to obtain an increase in performance by moving the computations onto a GPU. Cardinal *et al.* [30] used NVIDIA GeForce 8800 GTX to compute the acoustic likelihoods for their speech recognition system, which is based on a Finite-State Transducer (FST) framework. They gathered the average CPU and GPU times by computing the acoustic likelihoods 2000 times, and they reported a performance increase of 33.5% with the GPU implementation.

Also researchers such as Liu *et al.* in [31] used a GPU based system to accelerate computations of the speech acoustic probabilities. They used a Gaussian Mixture Model to accelerate these acoustic likelihoods computations and also applied the parallel reduction algorithm and matrix multiplication to boost the parallel processing of the GPU. This has significantly increased the actual speed of the speech recognition system. Their GPU acoustic likelihoods evaluation of the optimal method has shown an 11x speedup relating to the CPU. It is therefore the focus of this research work to improve the output speech recognition accuracy of our proposed implementation by putting more considerations into the feature extraction phase of our proposed system. Our idea is based on optimizing this phase in conjunction with the language and lexical models of our speech recognition system.

3.2 Components of an ASR System

As identified on most speech recognition systems, general components of automatic speech recognition systems include the following:

- Speech Signal acquisition
- Feature Extraction
- Acoustic Model
- Language Model
- Lexical Model
- Recognition

Each of these are now be discussed.

3.2.1. Speech Signal Acquisition

The speech signal acquisition refers to the process whereby audio signals (whether voiced or unvoiced) are recorded or attained by the speech recognition system for further processing by other components.

3.2.2. Feature Extraction Concept

The feature extraction component requires a lot of attention since the ultimate performance of the speech recognition system depends entirely on it. This part of the speech recognition system was fully explained in **Chapter 2** (Subsection 2.4.1). This part of the speech recognition system is faced with the challenge of speaker intonation, which is due to speaker variations. Thus when implementing a speech feature extraction algorithm, the fundamental frequency F_0 is the primary feature frequency that characterizes phrase intonation and word accent information. Thus it passes on speech material about the sound pitch, loudness, tempo and speech rhythm to convey information about the structure and meaning of an utterance as well as word boundaries.

3.2.3. Acoustic Model

This aspect of an ASR system accounts for most of the computational work and the general performance of the system since it works at the start of the system as the feature extraction. The model connects the recognized features of the speech signals with the expected phonetics of the assumed output phrase. The acoustic model is actually the part of the ASR system that detects uttered phonemes. When developing the acoustic model of an ASR system, audio recordings of speech and their text scripts are used and afterwards compiled into statistical representation of sounds which make up words. In this work, HMMs were used to do the acoustic modeling.

3.2.4. Language Model

The language model is the largest component of the automatic speech recognition system. This part of an ASR system is trained on large amounts of data, since it is possible that thousands of different words could be processed at times by the system. Language models help the speech recognizer to figure out probabilities of word sequences. This aspect of an ASR system works

independently from the acoustic model since the acoustic model is based on distinct sounds that forms a particular word whereas the language model is based on the language grammar associated with the system's corpus language. This model is directly associated with the speech engine's speech decoder. The language model contains a set of grammatical rules that enhances it for efficient estimation of likely word strings. ASR systems use n -gram language models to guide the search for correct word sequences by estimating the probability of the n th word on the basis of the $n-1$ previous words. The equation below by Ghai and Singh [32] gives the probability of occurrence of a word sequence W calculated as:

$$\begin{aligned}
 P(W) &= P(w_1, w_2, \dots, w_{m-1}, w_m) \\
 &= P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1w_2) \cdot \dots \cdot P(w_m|w_1w_2w_3 \dots w_{m-1})
 \end{aligned} \tag{3}$$

where P : is the word probability.

W : is the word sequence.

3.2.5. Lexical Model

The lexical model works hand in hand with the language model. It works exactly as a parser. If a word string is syntactically correct based on the adjacent language model, the lexicon parses it on. A lexicon is developed to provide the pronunciation of each word in a given language. Thus through a lexical model, various combinations of phones are defined to give valid words for the recognition phase.

3.2.6. Recognition Process

The recognition process or phase is the last stage of the ASR system. At this point a user can now see the recognized output results of the system. Figure 3.1 illustrates the working of the above mentioned models in a particular speech recognition system.

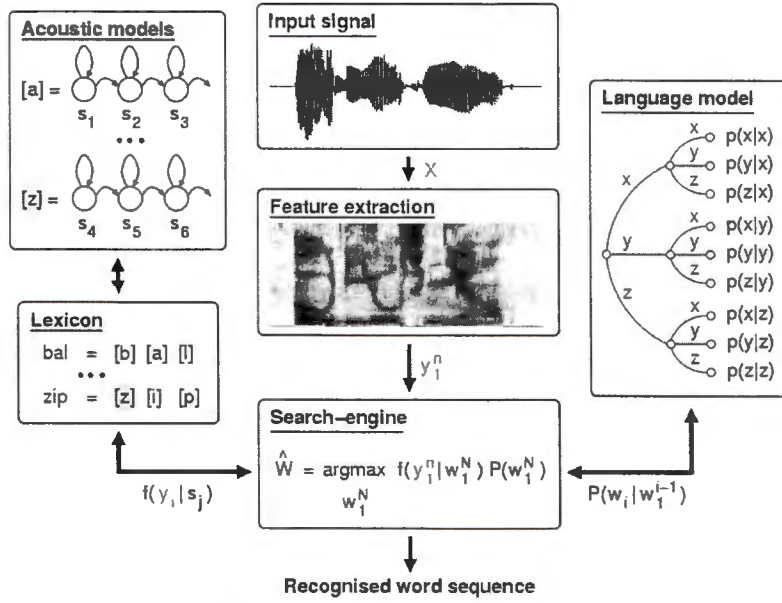


Fig. 3.1: Speech recognition models [33].

3.3 Hidden Markov Model (HMM) for ASR

An HMM is a generative probabilistic model, in which a sequence of observable variables are generated by a sequence of internal hidden states. However these hidden states are somewhat not directly observable. In the context of speech recognition, HMMs represent the transitions between these hidden states as a sequence of observation vectors derived from a probabilistic function of a first-order Markov chain. These HMMs are largely well suited to modeling time-varying patterns such as speech or audio signals.

Markov model states are identified with an output likelihood distribution that defines articulation variations, and hidden states are associated by probabilistic transitions that capture durational structures. Therefore an HMM can be used as a maximum likelihood classifier to compute the probability of a sequence of words given a sequence of acoustic observations. The advantage which adds to the efficient performance of HMMs is that these model states are highly parallel. Figures 3.2 and 3.3, illustrate the structure of an HMM for different speech recognition instances.

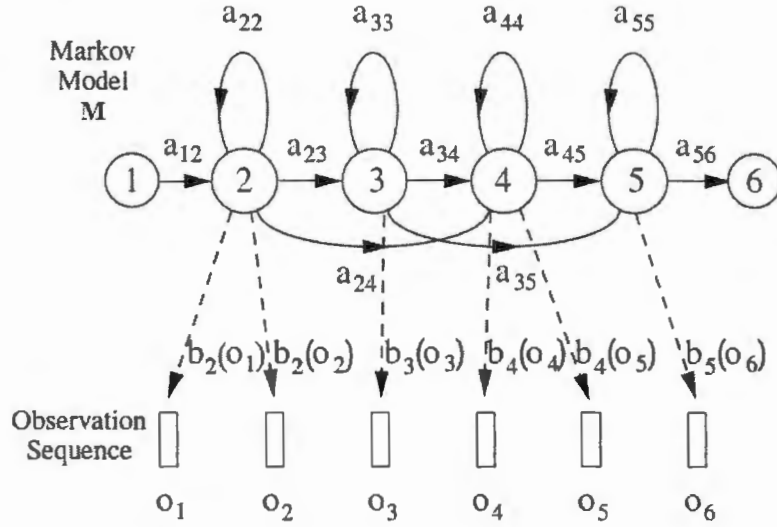


Fig. 3.2: HMM for an isolated speech recognition system [34].

In Figure 3.2, the observation sequence O is known and evidently the underlying state sequence X is unknown. With the condition that X is unknown, the required likelihood is computed using summation over all possible state sequences $X = x(1), x(2), x(3), \dots, x(T)$ such that:

$$P(O|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^T b_{x(t)}(o_t) a_{x(t)x(t+1)} \quad (4)$$

where P : is the word probability

O : is the observed sequence

M : represent the Model

X : are the underlying state sequences (i.e., hidden states)

$x(0)$: is constrained as the model entry state.

$x(T+1)$: is constrained as the model exit state.

b : is the probability density.

a : is the discrete probability.

Figure 3.3 illustrates an HMM for continuous speech recognition. To recognize these words the system has to parse relevant tokens at each state.

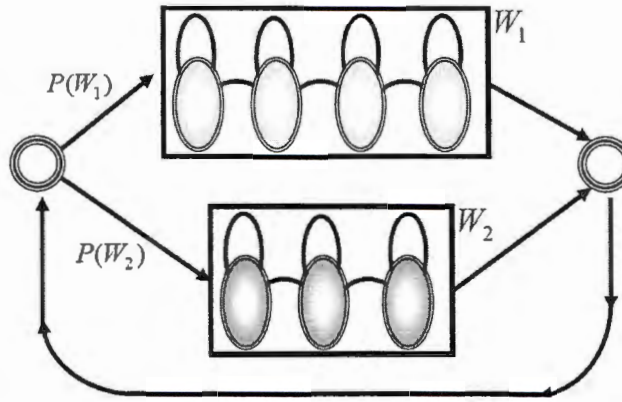


Fig. 3.3: Network of connected or continuous word recognition [35].

W_1 and W_2 represents word one and word two respectively. P represents the probability on which these words are estimated. In this instance, speech recognition systems allow users to speak almost naturally, while they (i.e., these systems) use dictation to determine the speech content. Systems with continuous speech recognition capabilities are most challenging to create since they use special methods to determine utterance limits.

3.4 Critical Analysis and Discussion

Pointing to what has been discussed as some of the challenges around speech recognition; a lot still needs to be done in terms of the efficiency (in speed) of speech processing and also the improvement of speech recognition accuracy. As noted in Section 3.1 and Subsection 3.2.3, the effort of computing acoustic likelihoods of a system recognized speech is a very compute intensive task. Therefore there is a need to improve not only the accuracy of speech recognition systems but the processing speed too. Powerful processing capability machines needs to be used together with optimum speech processing/ recognitions models to achieve better results.

However, recent research in GPU based systems has shown great achievements with regard to the improvement of general speech processing technologies. This is due to the fact that speech

recognition systems require compute intensive hardware to perform such demanding processing tasks. Therefore, the issue of hardware is critical in the area of speech recognition.

Approaches such as in [33] using artificial neural network, have shown possible best solutions to some of the serious challenges in speech recognition phases, such as when processing large volumes of data thereby increasing the ASR performance. Special capabilities of ANNs are that they can be trained on large volumes of data in any particular speech recognition system. This phenomenon about ANNs works as an advantage since the system is able to search for closely or accurately for possible estimates from these large pools of data. Enhancements on using these ANNs such as of Context-Dependent Deep-Neural-Network (CD DNN) have also shown great improvements on the accuracy of speech recognition as well as the performance speed.

Compared to CPUs, GPUs have a better standard in terms of efficient speech processing [36], provided all systems of operational implementations such as speech encoding algorithms, the system platform, external hardware (if any), etc. are compatible and are of optimum capabilities.

Recent advances in the field of ASR have developed very intelligent systems that can perform speech tasks similarly to humans. An example is the Apple's SIRI (Service Interface for Real-Time Information) which is currently embedded on Apple iPhones, iPads, and iPods. SIRI is a learning application; that means you have to teach it things that you want it to know. This application can respond to many utterances and commands that users can say to it. Users are able to use their voices to send messages, make calls, set reminders, etc. This application supports speech dictation fully. It is with such technologies that speech recognition is most interesting and thus there is a need to put effort into research on it.

3.5 Review Conclusions

Many research studies have been done around the area of speech recognition and a lot still needs to be done. GPUs have shown better speech processing performance than CPUs. This provides evidence that if better speech processing techniques, models or algorithms become developed, GPUs will perform better.

Also different approaches used on the actual speech processing bring forth greater achievements. Better results for optimum speech recognition can be achieved by coupling some implementation techniques and enhancing them for greater performance.

Automatic speech recognition using artificial neural network have also registered good achievements especially for large vocabulary continuous speech recognition (LVCSR). Their recent advancements coupled with Hidden Markov Models on GPU integrated systems give hope of better future achievements in this field of speech recognition.

CHAPTER 4: METHODOLOGY AND EXPERIMENTAL SETUP

This chapter describes the methodology and the actual experimental setup used in carrying out the aim of this research work.

4.1 Introduction

Recent ARS systems are integrated in handheld devices, automotive and huge field machineries. Developing such systems is a very demanding task since there are many factors, not only developmental components and human errors but also environmental factors such as noisy and reverberant conditions wherein automatic speech recognition performance becomes very poor. A major challenge is with the first stage of speech recognition being the feature extraction, wherein undistorted relevant features need to be extracted by the recognition system for further processing. Since automatic speech recognition systems need to be trained in large variances of data to improve accuracy and performance, the pre-emphasis stage of the speech recognition system needs to be well considered. The main focus in this work, is to optimize the LPC feature extraction algorithm in a GPU based system as a means for efficient speech feature extraction. As an advantage, the LPC algorithm technique is well suited for parallel processing architectures since the LPC algorithm coefficients are of parallel nature. Thus its implementation is well catered for using highly parallel CUDA GPUs.

4.2 Outline of the Methodology and System Setup

As a methodology, this work followed an LPC system survey to gain recent speech processing improvements by this technique, then a systematic experimental setup of all the tools and environments to be used and lastly, validation of results as the research proof of concept.

4.2.1. LPC System Survey

A survey of research advances in the field of automatic speech recognition using the LPC feature extraction technique was carried out. Current research improvements in the technique were analyzed and discussed as per motivation of optimizing this speech feature extraction technique on CUDA GPUs.

4.2.2. Experimental Setup

An experimental setup is outlined below as the second research method to describe the environment and a list of tools that were used in implementing the proposed work. This research work was implemented on a GPU platform. CUDA C/C++ was used as a developmental language for actual coding. The system used the Hidden Markov Model Toolkit (HTK) for the automatic speech recognition. This implementation was run in a Linux environment, since the HTK works efficiently in this environment. A list of tools used is given in point form below.

Tools and environments to be used:

- Intel Core i7-37703
- GPU Core (GeForce GTX 8800)
- CUDA C++
- HTK

4.2.3. Validation of Results: Proof of Concept

As proof of concept, the then optimized LPC algorithm was evaluated and compared to the original (non-optimized) LPC algorithm, as a ruling of which version improves performance. This evaluation was done on both CUDA architecture and on a non-parallel architecture (i.e., a CPU based architecture with no parallel capabilities). This covers the third method of the described research methodology.

4.3 Details of Used Architectures and Developmental Tools

4.3.1. Architecture Used

4.3.1.1. Architecture Description

In this research paper, the author uses a heterogeneous system comprising of a CPU and a GPU. In a heterogeneous system the CPU is called the host and the GPU is called the device. The CPU takes control of the system. Instructions to the device are copied from the host for processing and copied back to the host after being processed (by the device)

for output or other serial workloads, using relevant device programmable model functions.

Figure 4.1 shows an example of a heterogeneous system. Note that the serial processing part resembles the CPU and the parallel processing part resembles the GPU.

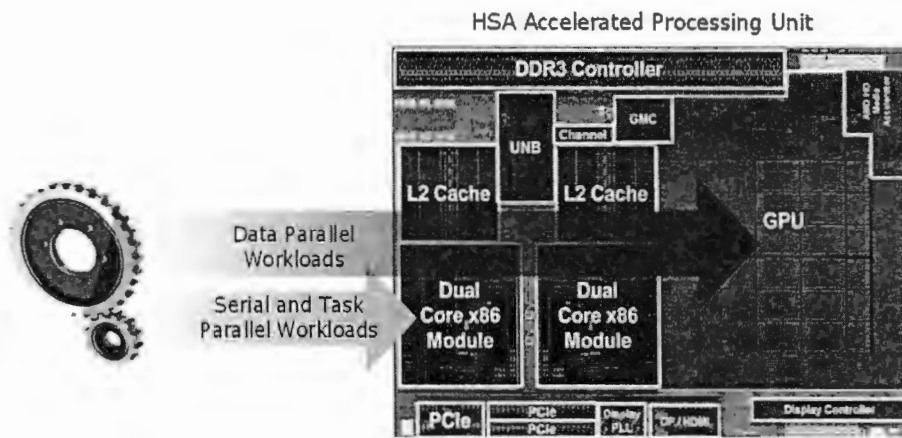


Fig 4.1: Heterogeneous accelerated processing system [37].

The architecture used in this research paper, runs on a 64 bit Linux (Ubuntu 12.04 LTS (Long-Term Support)) HP Compaq Pro Microtower with an Intel *i7* core as a host and a GeForce GTX 8800 GPU as a device.

4.3.1.2. The CPU Core

As indicated before, the CPU core used in this work is an Intel core *i7*-2130 with 4 GB of installed memory (RAM). Intel core *i7* is a member of the fourth generation core family. It is used in this research because it is approximately 20 times faster than the fastest Pentium 4 cores. Table 4.1 describes essential specifications of this core.

Table 4.1: Essential CPU Core Specifications.

Core Type	Intel 3rd Gen Core <i>i7-37703</i>
Number of Cores	Quad Core
Maximum Memory	4GB
Clock Speed	3.4 GHz
Intel® Smart Cache	8 MB
Graphics	Intel HD Graphics 4000
Chipset	Intel Q77 Express Chipset
Instruction Set	64-bit
etc.	

4.3.1.3. The GPU Core

The GPU core used in this work is the GeForce GTX 8800. This core is an 8 series GeForce family product member, it offers great thread processing capability and powerful computational speed and power in terms of data computation when compared to the GeForce 8800 GTS and the GeForce 8800 GT of its family product. Table 4.1 describes some of the essential specifications of the said core.

Table 4.2: Essential GeForce GTX 8800 Specifications.

CUDA Cores	575
Texture Fill Rate (billion/sec)	36.8
Memory Clock	900 MHz
Standard Memory Configuration	768 MB
Memory Interface Width	384 bit
Memory Bandwidth (GB/sec)	86.4
Supported Technologies	SLI
etc.	

4.3.2. Developmental Tools

4.3.2.1 CUDA Developmental Model

A CUDA programming model was used for implementing the objectives of this research paper (given in **Chapter 1**). The CUDA C++ programming language was used for all coding in this research. CUDA developmental tools namely: NVIDIA CUDA SDK (Version 5.5) and CUDA graphics drivers were installed on the system for software and hardware compatibility support and usage of the GPU.

CUDA also provides a natural extension of the C language that allows a transparent implementation of GPU accelerated algorithms. It is also easier to program NVIDIA GPUs using CUDA. These features of CUDA make it the most suitable platform to be used for signals and digital data processing.

4.3.2.2 Speech Corpus

A speech corpus is a database of speech based audio files and textual transcriptions of these speech audio files built in a format that can be used to create speech acoustic models. These acoustic models are used with a speech recognition system engine for speech processing.

In this work a Carnegie Mellon University (CMU) Sphinx 4 speech recognition toolkit was used for the recognition process. For the actual testing of the system (pointing to the experimental proposed algorithm) the author created a speech corpus file which was uploaded to the Sphinx 4 toolkit for the speech processing. This speech corpus was used due to limited resources since it is available for free.

4.3.2.3 The HTK

The Hidden Markov Model Toolkit was used in this work for automatic speech recognition. HTK maps well with parallel processing architecture since the toolkit itself supports parallelism. The HTK implements Hidden Markov Models as speech modeling techniques. For details regarding HMM see Section 3.3.

The HTK version used in this work is the: HTK version 3.4.1.

4.4 System Testing

The working system was then tested for efficient automatic speech recognition. Careful consideration regarding the overall performance of the system was taken as a measure for improvement. Performance comparison between the optimized LPC feature extraction technique and the non-optimized LPC one was done on two different architectures; a non CUDA GPU integrated system and a CUDA GPU integrated system.

A report of results regarding these systems are discussed and thoroughly analyzed next as to which one brings improvement in the overall process of automatic speech recognition in Section 5.8.

CHAPTER 5: RESULTS AND INTERPRETATIONS

This chapter discusses of the research results; wherein analytical interpretations of achieved results from both the CUDA GPU optimized LPC algorithm and the original LPC algorithm (i.e., CPU based machine) are given.

5.1 Introduction

In this research work, we used a CMU Sphinx 4 speech recognizer. To start with, we created a corpus file containing our desired language model as pointing to speech sentences which were then recognized by the Sphinx speech recognizer. This speech file was loaded onto the Sphinx recognizer library for speech recognition. A built-in LPC feature extraction technique was launched (here on a non-GPU based machine) for feature extraction performance analysis as it is the focus of this research. This technique implementation was also done on a GPU based system as emphasis for optimization and also to acquire performance results of that particular system.

As a means for testing the performance of the system, the performance survey of the original LPC feature extraction technique was taken as reference. The idea is to start by testing this system with minimum data (speech samples) whereby after some system review it can then be optimized for large data.

5.2 LPC Speech Feature Extraction Performance Survey

Research work such as in [36] has implemented the LPC using voice excitation; this approach was encoded using the Discrete Cosine Transform (DCT). Even though the use of DCT was computed at a lower bit rate (of about 2.4 Kbits) the resultant effect of their reconstructed speech signal was not intact compared to when the DCT was not used. The non-DCT encoding produced a better reconstruction of the original signal. This effect is due to the fact that without the DCT most of the signal's coefficients were processed rather than being compressed, which gives rise to a better reconstruction.

Figures 5.1, 5.2 and 5.3 illustrate the reconstruction of the original speech signals. Figure 5.1 shows the original speech signal, Figure 5.2 shows the reconstructions produced from a compression using the DCT and Figure 5.3 shows the reconstruction compressed without using the DCT.

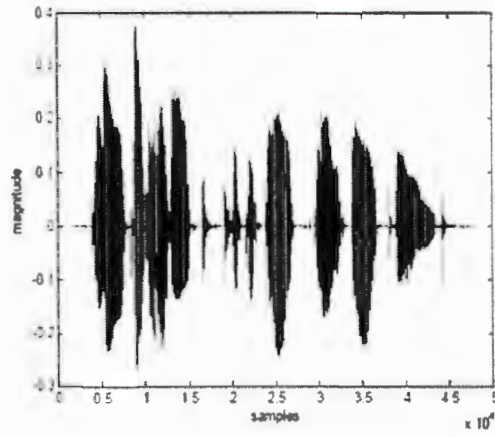


Fig. 5.1: Original LPC Speech Signal [36].

Figure 5.2 represent the voice excited LPC reconstructed speech signal compressed using the DCT.

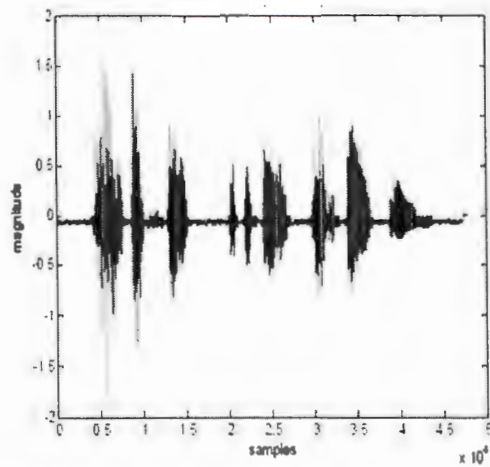


Fig. 5.2: Voice Excited LPC Speech Signal Using DCT [36].

Figure 5.3 represent the voice excited LPC reconstructed speech signal compressed without using the DCT.

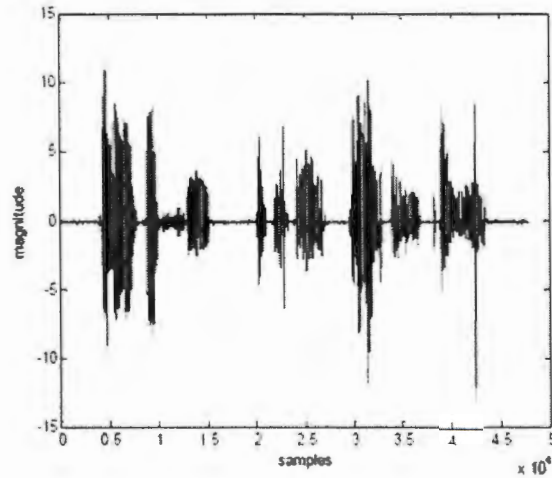


Fig. 5.3: Voice Excited LPC Speech Signal non-DCT [36].

5.3 Methods for Computing LPC Speech Feature Extraction

For efficiency measures, the following LPC speech feature extraction methods were used on both the CPU and GPU. Performance factors such as compute runtime, memory and power usage were considered since they contribute a lot to efficiency issues.

5.3.1. The Autocorrelation Method

The autocorrelation method involves setting minimal speech windows wherein required speech signals are sampled within the speech boundaries. This capability assists in refining the area on which the least square minimization is computed, thereby reducing prediction errors. This function works well since all samples outside the window equate to zero. The Autocorrelation pseudo code gives a clear compute description on how this method works. To achieve better approximations the autocorrelation method is computed over a large value of N .

Note that; x : is the original acoustic signal.

a : is the predictor coefficient.

N : the maximum window predictor value

p : the initial predictor value

pseudo code: *Autocorrelation Method*

```
proc autoCorr(x, window, a, N, p)
float x, window, N, p; integer N, p;
array x[], window[], a[];
begin // create x window and compute autoCorr vector;
array Q[0:p]; integer i, m; double E;
for (i=0; step 1 until N-1; do x[i] = x[i]*window[i]);
for m: = 0 step 1 until p do
    begin integer n;
        Q[m] := 0;
        for n: = 0 step 1 until N-1-m do Q[m] := Q[m] + x[n]*x[n+m]
    end
    /* applying the Levinson Durbin method to solve the matrix equation */
    E := Q[0];
    a[i] := Q[1]/E;
    for i: = 2 step 1 until p do
        begin int j; array a1[1:i-1];
            E := (1-a[i-1]^2)*E;
            a[i] := Q[i];
            for j: = 1 step 1 until i-1 do a[i] := a[i] - Q[i-j]*a[j];
        a[i] := a[i]/E;
        for j: = 1 step 1 until i-1 do a1[j] := a[j]-a[i]*a[i-j];
        for j: = 1 step 1 until i-1 do a[j] := a1[j];
    end
end autoCorr;
```

5.3.2. The Covariance Method

The covariance method also makes an effort to solve LPC computations, wherein real values of speech signal terms are calculated. However this method only works well on frequency samples that are very small. A best description of this method is given in the following pseudo code.

pseudo code: *Covariance Method*

```
proc covaMethod(x, a, N, p)
// computing linear prediction coefficients in a[1:p]
float x, N, p; integer N, p; array x[], array a[];
begin /* calculating the upper-triangular covariance matrix in Q */
integer m; array Q[0:P, 0:P];
for m: = 0 step 1 until p do
    begin // calculating Q[j,k] for each j & k. [k-j]= m;
        double R,S,T; integer n,j;
```

```

R: = 0; S: = 0; T: = 0;
// calculate S and R value;
for n: = 0 step 1 until m-1 do R:= R + x[n]*x[n-m];
for n: = 0 step 1 until N-1-p do S:= S+ x[n]*x[n + m];
  for j: = 0 step 1 until p - m do
    begin // calculate R and S's workout to:
      Q[j, m + j];
      Q[j, m + j]: = R + S;
      If j < p - m then R: = R + x[-j-1]*x[-m-j-1];
    end;
    for j: = p - m step -1 until 0 do;
    begin // adding T's workout;
      Q[j, m + j]: + T;
      If j > 0 then T: = T + x[N-j-1]*x[N-m-j-1];
    end;
    end calculating Q;
begin // calculating the square root of Q;
integer k,i,j;
for k: = 2 step 1 until k - 1 do
  begin
    for i: = 2 step 1 until i - 1 do
      for j: = 1 step 1 until i - 1 do
        Q[i, r]: = - Q[j, i]*Q[j, k];
        begin actual X;
          X: = Q[j, k];
          Q[j, k]: = Q[j, k]/ Q[j, j];
          Q[k, k]: = Q[k, k] - Q[j, k]*X;
        end
      end
    end
  end calculating square root of Q;

begin // calculating a[1:p];
integer k, i;
  for k: = 2 step 1 until p do
    for i: 1 step 1 until r - 1 do
      Q[0:k]: = Q[0:k] - Q[i, k]*Q[0, i];
      for k: = 1 step 1 until p do Q[0,k]/Q[k,k];
      for k: = p-1 step -1 until 1 do
        for i: = k + 1 step 1 until p do
          Q[0,r]: = Q[0,r] - Q[i,k]*Q[0,i];
        for k:= 1 step 1 until p do a[k]:= Q[0,i];
        end calculating a
      end covariance method
    end
  end

```

5.3.3. The Levinson Durbin Algorithm Method

This algorithm (often called Levinson Durbin Recursion) uses the same set of system states as of the autocorrelation sequences to find the filter response of the signal. In its implementation this algorithm has a functionality that produces a minimum phase filter. The following pseudo code describes the work functionality of this algorithm. Note that the Levinson Durbin solution method builds on the autocorrelation application method.

pseudo code: *Levinson Durbin Method*

```
proc levinsonDub(x, window, a, N, p)
integer i, j; double E;
array Q[];
E = Q[0];
  a[i] = Q[1]/E;
  for i = 2 step 1 until p do
    begin int j; array a1[1:i-1];
      E = (1-a[i-1]^2)*E;
      a[i] = Q[i];
      // start applying recursion
      for j = 1 step 1 until i-1 do a[i] = a[i] - Q[i-j]*a[j];
    a[i] = a[i]/E;
    for j = 1 step 1 until i-1 do a1[j] = a[j]-a[i]*a[i-j];
    for j = 1 step 1 until i-1 do a[j] = a1[j];
  end
end levinsonDub;
```

Looking at all the methods described above, it can easily be noticed that there are a lot of opportunities for optimizing these methods using parallel architectures such as CUDA GPUs. All these algorithms are highly parallelizable; hence implementing them on CUDA GPUs was not difficult.

5.4 CPU and GPU Based LPC Implementation

System implementations for both the CPU (non-GPU integrated system) and GPU (CPU host system) are shown in the following subsections. All the implementations were done on a Linux (Ubuntu LTS ver. 12.04) system.

5.4.1. CPU Based LPC Implementation

The LPC speech feature extraction technique was also optimized on the CPU (non-GPU integrated) based machine. Created speech samples were read from the CMU Sphinx speech library. Figure 5.4 illustrates processes of loading speech samples from the dedicated library.

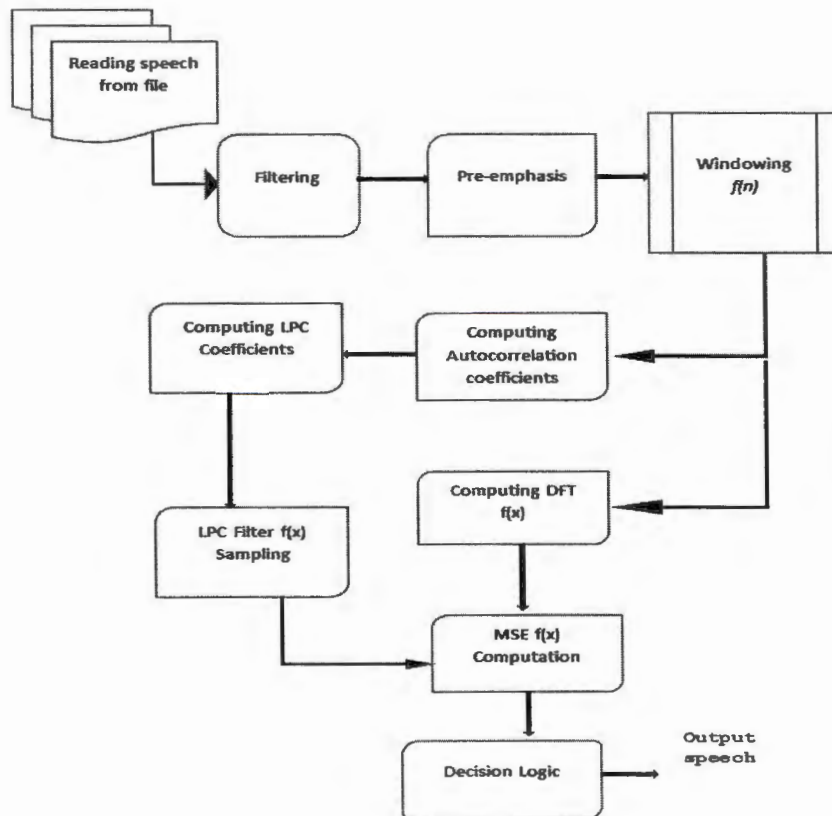


Fig. 5.4: CPU Optimized LPC Processes for reading and loading speech samples.

Note from Figure 5.4 above that speech samples are read sequentially into the LPC processing system.

5.4.1.1. CPU Processed LPC Speech Waveform and Power Spectrum

5.4.1.1.1. CPU Processed Speech Waveform

Figure 5.5 represents the LPC speech waveform implementation processed using the CPU. This waveform was generated at 100 per cent amplitude, 220 Hz frequency, zero degrees phase, 22 050 Hz wave sampling rate, and file length equal to 2 seconds under a sample size of 16 Bits.

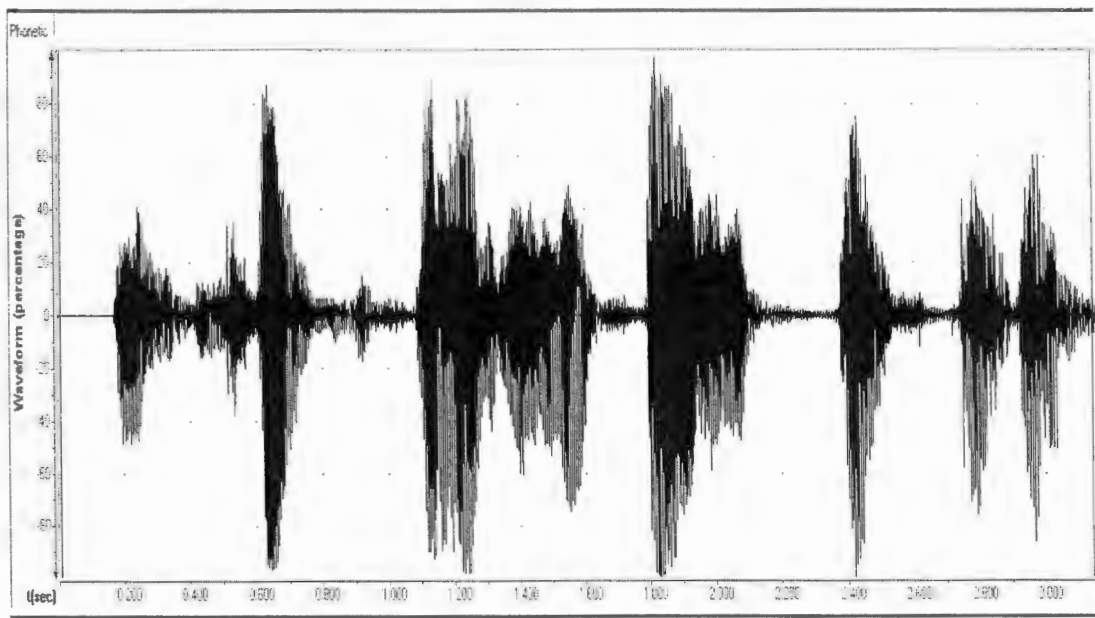


Fig. 5.5: CPU Processed LPC Speech Waveform.

5.4.1.1.2. CPU Processed Power Spectrum

The resulting power spectrum of the above speech waveform is illustrated in Figure 5.6. The y-axis represents power which is measured in Decibels (dB). The spectrum shows the power fall on each frequency. The x-axis represents frequency which is measured in Hertz (Hz). This spectrum gives a clear indication of the dissipated power at all the specified frequencies.

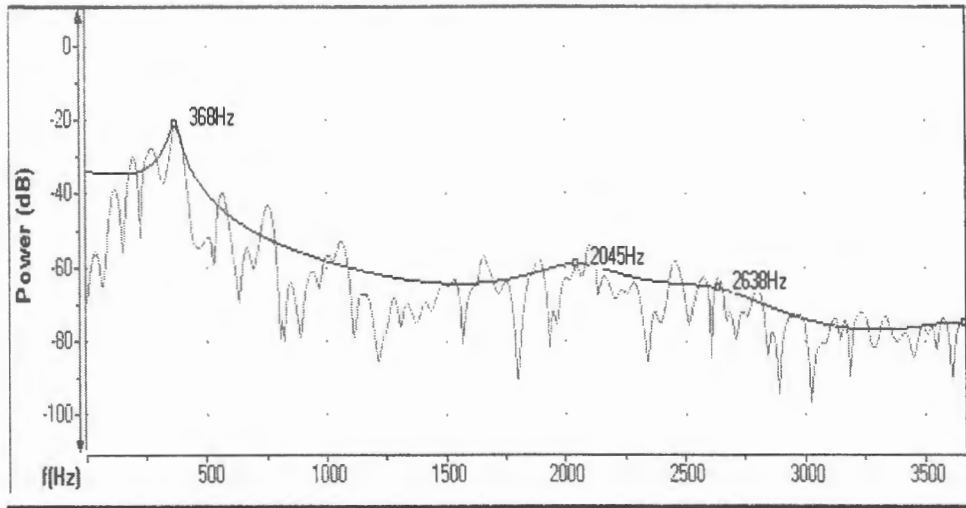


Fig. 5.6: CPU Power Spectrum at Specified Frequencies.

5.4.2. GPU Based LPC Implementation

The LPC technique was also optimized on the CUDA GPU machine. This implementation took advantage of the parallel nature of the LPC feature extraction technique; thus giving best speech processing performance results. Figure 5.7 illustrates the LPC processes for loading speech samples from the dedicated library.

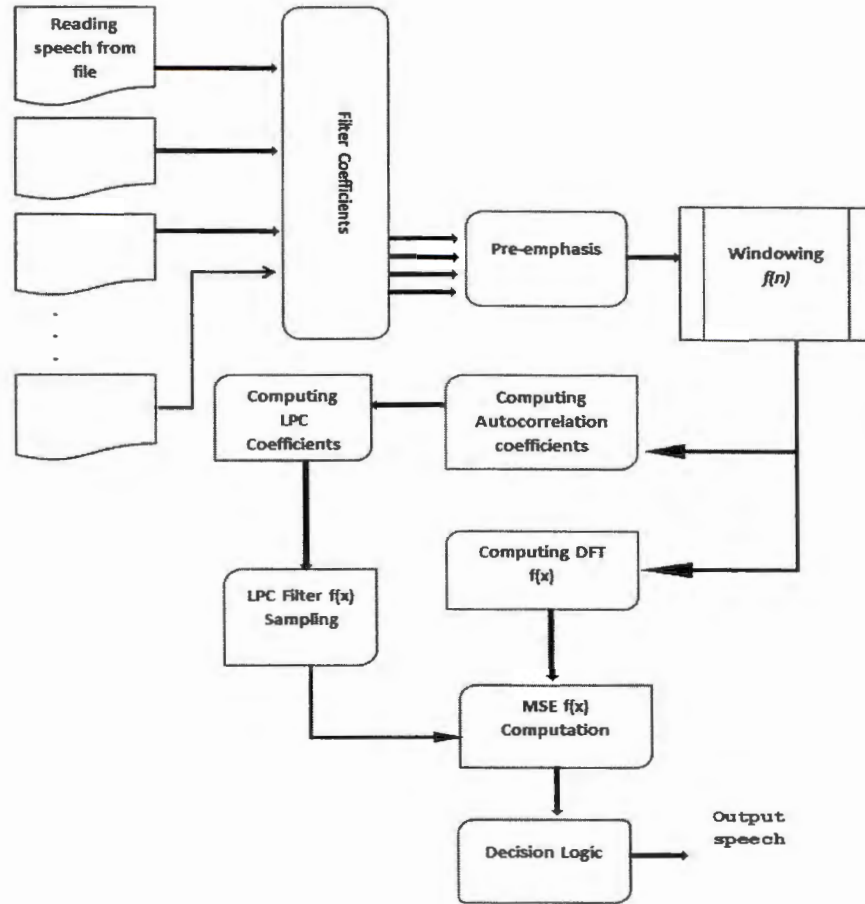


Fig. 5.7: GPU Optimized LPC Processes for reading and loading speech samples.

Since the LPC speech feature algorithm has parallel processing capabilities, the work of reading and loading speech samples into the system becomes an easy task. As an account to evaluate the above mentioned implementations, tabulated information regarding the system speedup and the performance accuracy is given in Table 5.1. As a measure of efficiency, several critical factors that are commonly associated with the optimal performance of a system were looked at. These factors include system memory usage, system runtime and the percentage recognition accuracy.

5.4.2.1. GPU Processed LPC Speech Waveform and Power Spectrum

5.4.2.1.1. GPU Processed Speech Waveform

Figure 5.8 represents the LPC speech waveform implementation processed using the GPU. This waveform was also generated at 100 per cent amplitude, 220 Hz frequency, zero degrees phase, 22 050 Hz wave sampling rate, and file length equal to 2 seconds under a sample size of 16 Bits.

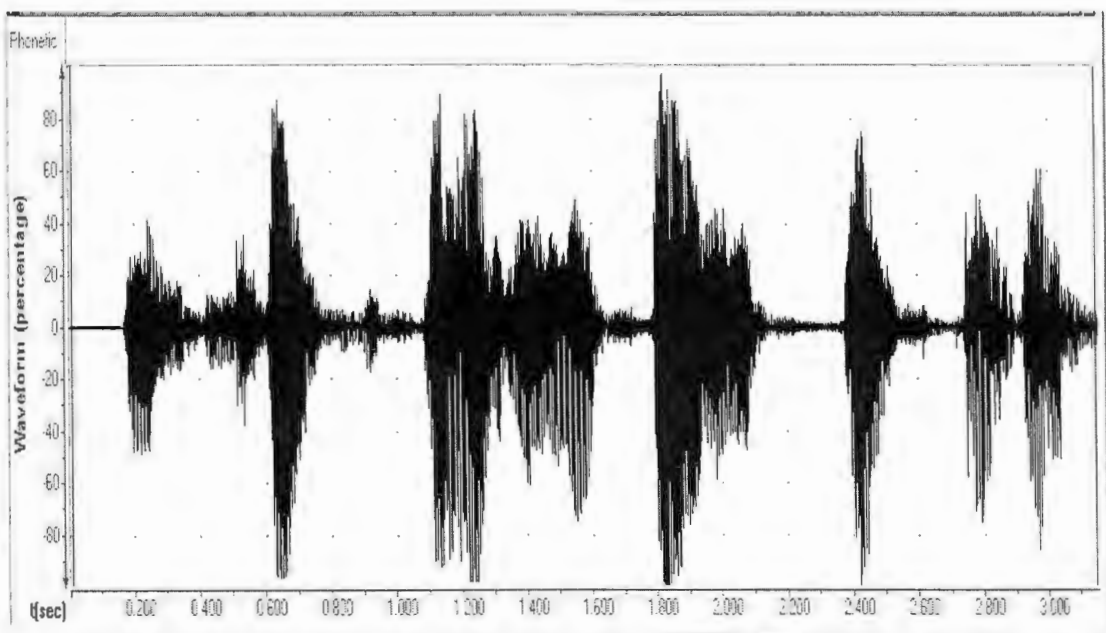


Fig. 5.8: GPU Processed LPC Speech Waveform.

5.4.2.1.2. GPU Processed Power Spectrum

The resulting power spectrum of the above speech waveform is illustrated in Figure 5.9 below.

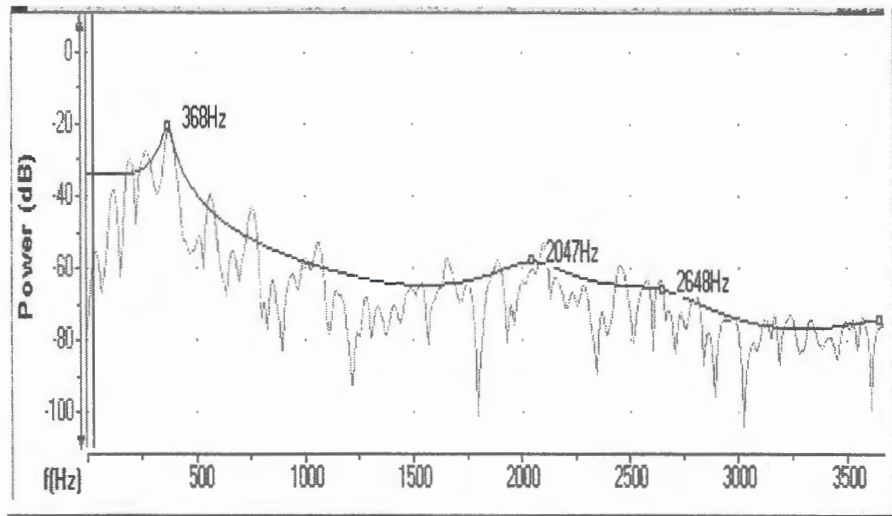


Fig. 5.9: GPU Power Spectrum at Specified Frequencies.

Table 5.1 gives a summary of speech processing performance achieved on the three LPC implementations. Speech processing metrics which are the Real Time Factor and Word Error Rate were used for testing the overall performance improvement of the recognition system. The speech recognition accuracy was also calculated to verify the best technique on the overall performance of the system.

Table 5.1: Performance Table for the Used LPC Implementations.

Algorithm Type	Speech Processing Metrics		Recognition Accuracy (%)
	RTF	WER	
Original LPC (non-Optimized)	13.64	24.76	68.56
CPU Optimized LPC	7.66	17.82	74.23
GPU Optimized LPC	4.26	12.86	82.32

From Table 5.1, it can be seen that the GPU implementation of the LPC brings good speech performance results compared to that of the original and optimized CPU implementations of the LPC algorithm. It was also realized that the GPU implementation uses more compute power and memory than the other LPC implementations. However the GPU implementation computes faster than the other implementations and hence gives the best recognition accuracy.

5.5 PSNR (Peak Signal to Noise Ratio) Signal Analysis

A PSNR is a tool used to compute a signal's maximum power ratio compared to its reconstructed power signal representation. This tool gives representation details such as signal noise or presentation intensities between the two signals. It is used to evaluate signals reconstruction quality. This quantity is expressed in logarithmic decibel (dB) balance. Since feature extraction involves a lot of signal processing, the PSNR signal analysis was also used as another method on both the CPU and GPU implementations of the LPC speech feature extraction to evaluate as to which implementation produced better signal presentation results. Again as proof of concept; both the CPU LPC (CPU optimized) version and the GPU LPC version (CUDA GPU optimized) speech feature extraction technique were compared with the original LPC (non-optimized) to determine which version improves performance. The PSNR is defined for two speech signals (Original and Recognized) O and R sampled on a window size $M \times N$ as:

$$PSNR_{(O,R)} = 20 \log_{10} \frac{MAX_O}{\sqrt{MSE_{(O,R)}}} \quad (5)$$

MAX is the maximum signal intensity value of the original signal O, and MSE is the mean square error between O and R. MSE is given as:

$$MSE_{(O,R)} = \frac{1}{M} \frac{1}{N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \|O(i,j) - R(i,j)\|^2 \quad (6)$$

PSNR value comparisons for both the CPU and GPU LPC implementations are given on the following subsection. System performance analyses regarding these values were given as motivations for which implementation between the two offers a better performance. The tabulated data in Tables 5.1 and 5.2 was achieved from two different speech samples. Each table reflects the same sampling size used.

Table 5.2: PSNR Values for Speech Sample A.

Speech Sample A	Sampling Size			
	264x316	324x416	424x512	516x564
Original LPC (non-optimized)	23.542411	28.451531	32.146412	35.462131
CPU Optimized LPC	22.562121	26.521132	29.452211	32.24511
GPU Optimized LPC	18.452641	22.4583251	23.651341	26.752131

Table 5.3: PSNR Values for Speech Sample B.

Speech Sample B	Sampling Size			
	264x316	324x416	424x512	516x564
Original LPC (non-optimized)	26.632412	33.325411	36.235142	41.452113
CPU Optimized LPC	24.442142	27.342216	33.263512	37.432512
GPU Optimized LPC	21.642131	24.457212	27.441323	32.542314

From Tables 5.2 and 5.3 above it can be seen that on each table, PSNR values for the GPU optimized LPC implementation offer less noise presentation.

Table 5.4 illustrates speedup times for the original CPU, optimized CPU and optimized GPU implementations of the LPC algorithm.

Table 5.4: Original CPU, CPU Optimized and GPU Optimized LPC Speedup.

Input Data Size ($M \times N$)	Original LPC CPU(ms)	CPU Optimized LPC (ms)	GPU Optimized LPC (ms)
3244x3204	32.42	26.44	16.72
2424x2284	28.26	24.62	14.48
1164x1024	26.41	22.44	13.34
1024x884	24.32	20.23	11.24
860x724	22.36	18.64	8.26
720x576	20.48	15.36	4.62
576x512	18.62	13.46	2.58

Figure 5.10 illustrates a runtime graph performed by the original CPU, optimized CPU and optimized GPU implementations of the LPC algorithm on different data sizes.

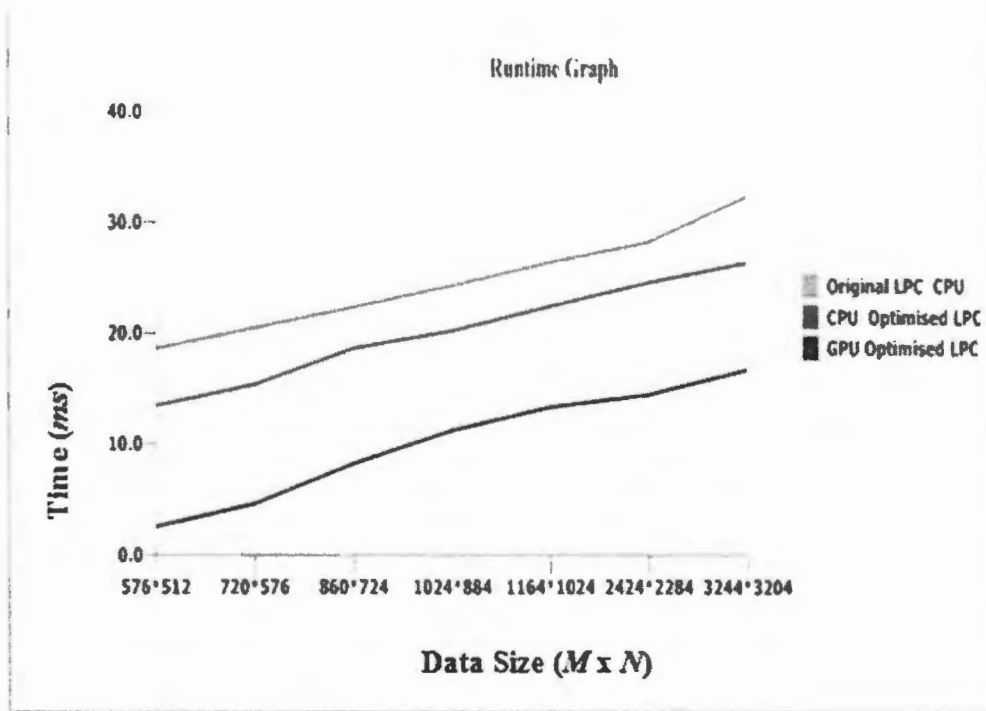


Fig. 5.10: Runtime Graph for Original CPU, Optimized CPU and Optimized GPU.

5.6 Analysis and Interpretation of Results

The purpose of this research was to optimize the LPC speech feature extraction technique on CUDA GPUs. Thus this section gives critical analysis and interpretation of the results obtained.

Different results were presented from Subsection 5.4.2.1.1 to Section 5.5 and are given in graphical form as on the Figures 5.5, 5.6, 5.7, 5.8 and Table 5.1.

Figure 5.5 presents an LPC speech waveform generated on a CPU core machine. Experimental results achieved on these implementations indicated that the CPU used in this work produced fair results in reference to the original LPC implementation. Speech processing accuracy issues were realized as this implementation did not exhibit sufficient improvements as compared to the original implementation. However this implementation has offered great memory usage which is also a direct concern to the overall performance of the system.

Figures 5.5 and 5.7 represent the LPC speech output waveforms by the CPU and GPU respectively. Based on the output waveform of these implementations, it was also realized that the CPU

implementation had filtering issues that impacted on the output speech signal whereas, on the contrary, the GPU produced a well filtered signal. Thus compared to the CPU implementation the GPU implementation produced a smoother and clearer sounding speech signal, which contributes greatly towards the improvement of speech accuracy.

Figures 5.6 and 5.8 indicate power spectrum functions for the CPU and GPU respectively. It can be observed from the two spectra that the GPU implementation consumes slightly greater power than the CPU implementation. This is due to processing procedures that manifest on this parallel architecture. However, the GPU reads and uploads data faster than the CPU. In this research work, we have achieved speech feature extraction improvements which evidently increased speech processing performance. When compared to that of the original LPC implementation (non-optimized), the GPU implementation brings great improvement. Table 5.1 gives the overall achieved speech processing performance as well as speech recognition improvements that each implementation brings.

It can also be observed in Tables 5.2 and 5.3 that the GPU optimized LPC offers less signal noise. This is due to the powerful filtering and smoothing capability of the GPU core. On each speech sample, the GPU implementation of the LPC speech feature extraction algorithm was also found to be faster on signal representation.

Table 5.4 shows speedup times for the original CPU, CPU Optimized and GPU Optimized LPC implementations on various data sizes. Even though the optimized CPU implementation shows some speedup improvements than the original CPU implementation, the GPU optimized LPC indicates good speedup performance rather than the other two implementations. This is also a positive factor of the overall speech processing improvement by the GPU.

Figure 5.10 illustrates a runtime graph for all the three LPC implementations. It can be clearly seen that the GPU implementation computes in a very lower time than the other two implementations. However the optimized CPU LPC version also indicates a better computation time than the original CPU implementation.

In this research the GPU based implementation of the LPC feature extraction algorithm performed better in terms of speed and recognition accuracy as compared to the CPU based implementation. In our implementation, an overall speech recognition accuracy of 82.32% was achieved which was

then recorded as the best performing implementation out of the three. Although in our case these results were recorded as the best achievement, the literature indicates that there are other researches such as Hachkar et al. [38] with 92% as well as Pour and Farokhi [39] with 98% that achieved better results.

CHAPTER 6: SUMMARY AND CONCLUDING REMARKS

6.1 Introduction

This chapter gives the overall summary of the achievements regarding the main focus of this research work, with comments and motivations on the deliverables of the proposed technique for speech recognition. Optimization issues under the proposed environment are discussed. Analytical conclusions on the particular study are given.

Recall from **Chapter 1** that the proposed goal and objectives of this research work were:

6.2 Proposed Goal and Objectives

6.2.1. Proposed Goal

The main goal of this research work was to optimize the LPC feature extraction algorithm for efficient automatic speech recognition on a GPU based platform using CUDA.

6.2.2. Proposed Objectives

The objectives of this research were:

- To conduct a literature study on the GPU and CUDA based ASR technologies on different techniques.
- To optimize the LPC feature extraction technique for parallel processing by a GPU in an attempt to attain efficient extraction of relevant information features during the speech recognition process, on a CUDA based platform.
- To analyze and evaluate the performance of the proposed algorithm for efficient speech feature extraction towards better speech recognition in terms of accuracy.

6.3 Analysis of Research Achievements

Analysis and Results of the Research Goal:

The goal of this research as indicated in Section 6.2.1 was achieved by optimizing an LPC speech feature extraction algorithm for efficient performance and implemented on both the CPU and GPU. An LPC speech feature extraction algorithm was optimized for efficient performance and implemented on both the CPU and GPU. This is illustrated in **Chapter 5** in Figures 5.5 and 5.7 where the CPU and GPU speech output waveforms are given.

Analysis and Results of the Objectives:

- A literature study on the ASR technology on CUDA GPUs was successfully conducted.
- An LPC speech feature extraction algorithm was optimized on CUDA architecture for efficient feature extraction and processing.
- Speech processing performance of the optimized LPC was analyzed and evaluated for efficiency as motivation for developing a better speech recognition system in terms of speech accuracy.

6.4 Research Challenges

Challenges encountered in this research project were that:

- Due to the unavailability of some dedicated research equipment, we had to remotely connect to a high performance cluster outside our institution to perform experiments. This forced the author to work under a lot of pressure due to the research runtime.
- Since the author was still new to the technology (CUDA and GPU programming) used in this research; therefore a lot of effort had to be made to ensure a thorough understanding of critical concepts of this technology.

6.5 Future Work

The LPC speech feature extraction algorithm has been used for some time, although its original implementation alone is not that effective. However, it can be further enhanced for example by parallel processing architecture towards greater feature extraction performance thereby contributing to better speech processing performance. It is also evident that the LPC speech feature extraction algorithm can be accelerated using the latest NVIDIA SLI (Scalable Link Interface) architectures.

As part of our focus, based on the results achieved, we are currently investigating LPC feature coefficients processing, and attempting to optimize the pre-emphasis stage of the feature extraction phase. This will improve the extraction process as well as the sound detection process of the recognition system.

6.6 Summary of the Research

In this research work, we have realized great feature extraction improvements using the CUDA optimized LPC algorithm, which has impacted positively on the speech recognition process. CUDA provides a good way of mapping parallel processing algorithms and procedures to the GPU, thereby boosting the performance of the powerful processing capability of this core. The GPU implementation of the LPC feature extraction algorithm used in this work has given hope for future studies and possible processing upgrades of this algorithm.

REFERENCES

- [1] A. G. Chițu, L. J. M. Rothkrantz, J. C. Wojdel and P. Wiggers, "Comparison Between Different Feature Extraction Techniques for Audio-Visual Speech Recognition," *Journal on Multimodal User Interfaces (JMUI)*, vol. 1, no. 1, pp. 7-20, March 2007.
- [2] Interactive Speech. (2013, March) Interactive-Speech. [Online]. Available: <http://interactive-speech.com/>. [Accessed: 22 March 2013].
- [3] Y.Park, S. Patwardhan, K. Visweswariah and S.C. Gates, "An Empirical Analysis of Word Error Rate and Keyword Error Rate," in *Proceedings of the International Conference on Spoken*, Brisbane, Australia, 2008.
- [4] G. Kennedy. (2009, March) Wodonga Tafe. [Online]. Available: <http://ndco.wodongatafe.edu.au/Data/Sites/1/universalaccess12usingspeechrecognitiongkict.pdf>. [Accessed: 18 April 2013].
- [5] NVIDIA Corporation, (2012, February), "NVIDIA CUDA Programming Guide v. 4.2". [Online]. Available: <http://www.nvidia.com>. [Accessed: 24 April 2013].
- [6] K. M. Modieginyane, Z. P. Ncube and N. Gasela, "CUDA Based Performance Evaluation of the Computational Efficiency of the DCT Image compression Technique on Both the CPU and GPU". *Advanced Computing: An International Journal (ACIJ)*, Vol.4, No.3, pp. 1-15, May 2013.
- [7] D. Triolet. (2012, February) NVIDIA GeForce GTX and GTS - BeHardware. [Online]. Available: <http://www.behardware.com/art/imprimer/644/>. [Accessed: 24 April 2013].
- [8] Furui S. (2003), "*Robust Methods in Automatic Speech Recognition and Understanding*", in EUROSpeech 2003. GENEVA, 2003.
- [9] X. Huang, J. Baker and R. Reddy, "A Historical Perspective of Speech Recognition". *COMMUNICATIONS OF THE ACM (CACM)*, Vol. 57, No. 1, pp. 94 – 103, January 2014.
- [10] K. H. Davis, R. Biddulph and S Balashek, "Automatic Recognition of Spoken Digits," *The Journal of The Acoustical Society of America*, vol. 24, no. 6, pp. 637-642, November 1952.
- [11] M. A. Anusuya and S. K. Katti, "Speech Recognition by Machine: A Review," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 6, no. 3, pp. 181-205, September 2009.

- [12] X. Chen, A. Eversole, G. Li, D. Yu, and F. Seide, "Pipelined back-propagation for context dependent deep neural networks." *In Proceedings of Interspeech, 2012*. September 2012.
- [13] Poli G., Levada A. L. M., Mari J. F. and Saito J. H. (2007), "*Voice Command Recognition with Dynamic Time Warping (DTW) using Graphics Processor Units (GPU) with Compute Unified Device Architecture (CUDA)*," 19th International Symposium on Computer Architecture and High Performance Computing, 2007.
- [14] N. Jaitly, P. Nguyen, A. Senior and V. Vanhoucke, (2012, March) Toronto University. [Online]. Available: <http://learning.cs.toronto.edu/> . [Accessed: 10 May 2013].
- [15] N. Morgan and H. Bourlard. (1995, May) International Computer Science Institute. [Online]. Available: <http://www1.icsi.berkeley.edu/ftp/global/pub/speech/papers/ieeespm95-hyb.pdf> . [Accessed: 20 May 2013].
- [16] J. Tebelskis. (1995, May) Carnegie Mellon University. [Online]. Available: [http://www.ru.lv/~peter/zinatne/ebooks/\(Brain%20Study\)_Speech%20Recognition%20using%20Neural%20Networks.pdf](http://www.ru.lv/~peter/zinatne/ebooks/(Brain%20Study)_Speech%20Recognition%20using%20Neural%20Networks.pdf) . [Accessed: 30 May 2013].
- [17] G. E. Dahl, D. Yu, L. Deng and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
- [18] J. Hosom, J. de Villiers, R. Cole, M. Fanty, J. Schalkwyk, Y. Yan and W. Wei. (2006, February) OGI School of Science & Engineering (OGI). [Online]. Available: http://speech.bme.ogi.edu/tutordemos/nnet_training/tutorial.html . [Accessed: 27 March 2014].
- [19] J. Chong, Y. Yi, A. Faria, N. R. Satish, and K. Keutzer, "Data-Parallel Large Vocabulary Continuous Speech Recognition on Graphics Processors," in *Proceedings of the 1st Annual Workshop on Emerging Applications and Many Core Architecture (EAMA)*, June 2008.
- [20] NVIDIA. (2010, June) NVIDIA Developer Zone. [Online]. Available: <http://www.nvidia.com/content/cuda/spotlights/future-voice-search-accelerated-cuda.html> . [Accessed: 02 April 2014].

- [21] H. K. Elminir, M. A. ElSoud and L. M. A. El-Maged, "Evaluation of Different Feature Extraction Techniques for Continuous Speech Recognition.," *International Journal of Science and Technology (IJST)*, vol. 2, no. 10, p. 201, October 2012.
- [22] F. Trebien, and M. M. de Oliveira Neto, "An Efficient GPU-based Implementation of Recursive Linear Filters and Its Application to Realistic Real-Time Re-Synthesis for Interactive Virtual Worlds," in *UFRGS*, Porto Alegre, August 2009.
- [23] A. R. Madane, Z. Shah, R. Shah, and S. Thakur, "Speech Compression Using Linear Predictive Coding," in *Proceedings of the International Workshop on Machine Intelligence Research*, Nagpur, 2009, pp. 119-122.
- [24] U. Shrawankar and V. Thakare, "Techniques for Feature Extraction in Speech Recognition System: A Comparative Study," SGB Amravati University, Amravati, 2013.
- [25] L. R. Rabiner and R. W. Scafer, *Digital Processing of Speech Signals*, 1st ed., A. V. Oppenheim, Ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1978.
- [26] B. S. Atal., "The history of linear prediction," *IEEE Signal Proc. Mag*, no. 23, pp. 154–161, 2006.
- [27] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan*, vol. 53-A, pp. 36–43, 1970.
- [28] K. Yao, D. Yu, F. Saide, H. Su, L. Deng, and Y. Gong, "Adaptation of Context-Dependent Deep Neural Networks for Automatic Speech Recognition," in *SLT 2012, IEEE*, Miami, 2012.
- [29] G. Poli, A. L. M Levada., J. F. Mari and J. H. Saito, "Voice Command Recognition with Dynamic Time Warping (DTW) using Graphics Processor Units (GPU) with Compute Unified Device Architecture (CUDA)," in *19th International Symposium on Computer Architecture and High Performance Computing*, 2007.
- [30] P. Cardinal, P. Damouchel , G. Boulianne. and M. Comeau, "GPU accelerated acoustic likelihood computations.," in *Proceedings of 9th International Interspeech Conference*, Bisbane, 2008, pp. 181-205.
- [31] Y. Liu, Z. Zhang, Y. Hi, Q. Zhao and Y. Yan, "Accelerate Acoustic Likelihood Computations on GPU for Speech Recognition," in *International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, Paris, 2013, pp. 1019-1022.

- [32] W. Ghai and N. Singh, "Literature Review on Automatic Speech Recognition," *International Journal of Computer Applications* (0975 - 8887), vol. 41, no. 8, pp. 42-50, March 2012.
- [33] Automatic Speech Recognition: Short Introduction : [Online].
Available: http://www.esat.kuleuven.be/psi/spraak/demo/Recog/asr_intro.html.
[Accessed: 20 March 2014].
- [34] Isolated Word Recognition: [Online].
Available: http://wwwlands2.let.kun.nl/members/software/HTKBook_2.0/node5.html .
[Accessed: 13 February 2014].
- [35] B. Pellom. (2004, September) University of Colorado. [Online]. Available:
http://www.cs.tut.fi/~puhtunn/lecture_model_search.pdf . [Accessed: 25 February 2014].
- [36] N. Sharma and N. Mehta, "Advanced Speech Compression VIA Voice Excited Linear Predictive Coding Using Discrete Cosine Transform (DCT)," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 2, no. 3, pp. 144-148, February 2013.
- [37] AMD, "AMD Developer Central," [Online]. Available:
<http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/>. [Accessed: 11 April 2013].
- [38] Z.Hachkar, A. Farchi , B.Mounir and J. EL Abbadi, "A Comparison of DHMM and DTW for Isolated Digits Recognition System of Arabic Language," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 3, no. 3, pp. 1002- 1008, March 2011.
- [39] M. M. Pour and F. Farokhi, "An Advanced Method for Speech Recognition ," *World Academy of Science, Engineering and Technology* , vol. 3, no. 1, pp. 995-1000, January 2009.

BIBLIOGRAPHY

- [1] K. M. Indrebo, R. J. Povinelli and M. T. Johnson, "Minimum Mean-Squared Error Estimation of Mel-Frequency Cepstral Coefficients Using a Novel Distortion Model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1654-1661, November 2008.
- [2] Aida-Zade K.R., Ardil C. and Rustamov S. S. (2006), "Investigation of Combined use of MFCC and LPC Features in Speech Recognition Systems", *World Academy of Science, Engineering and Technology*, 2006.
- [3] Hinton G., Deng L., Yu D., Dahl G., Mohamed A. R., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T. and Kingsbury B. (2012), "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, 2012.
- [4] Hong X., Yao H., Wan Y. and Chen R. (2006), "A PCA based visual DCT feature extraction method for lip-reading", *IIHMSP*, Vol. 0, pp. 321–326, 2006.
- [5] Itakura F. & Saito S. (1971), "Digital filtering techniques for speech analysis and synthesis", *Proc. 7th Int. Conf. Acoust.* 25(C-1), pp. 261–264 (1971).
- [6] Kelly F. and Harte N, (2010), "A Comparison of Auditory Features for Robust Speech Recognition", 18th European Signal Processing Conference, August 23-27, 2010. EURASIP 2010, ISSN 2076-1465. 2010.
- [7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath and B. Kingsbury, "Deep neural networks for acoustic modelling in speech recognition," *IEEE Signal Processing Magazine*, 2012.
- [8] X. Hong, H. Yao, Y. Wan & R. Chen, "A PCA based visual DCT feature extraction method for lip-reading ," *IIHMSP*, pp. 321–326, 2006.
- [9] L. Yong, Z. Zhen, S. Yujing, Z. Qingwei and Y. Yonghong, "Accelerate Acoustic Likelihood Computations on GPU for Speech Recognition," in *International Conference on Computer Science and Electronics Engineering (ICCSEE)*, Beijing, China, 2013, pp. 1019 - 1022.
- [10] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky, "Feature extraction using non-linear transformation for robust speech recognition on the Aurora database," in *Proc. Int. Conf. Audio, Speech, Signal Process.*, 2000, pp. 1117–1120.

- [11] T. Virtanen, J. F. Gemmeke, and A. Hurmalainen, "State-based labeling for a sparse representation of speech and its application to robust speech recognition," in Proc. Interspeech, 2010, pp. 893–896.
- [12] H. Van Hamme, "Robust speech recognition using cepstral domain missing data techniques and noisy masks," in Proc. Int. Conf. Audio, Speech, Signal Process., 2004, vol. 1, pp. 213–216.
- [13] M. D. Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools, and D. Van Compernelle, "Template based continuous speech recognition," IEEE Trans. Audio, Speech, Lang. Process., vol. 15, no. 4, pp. 1377–1390, May 2007.
- [14] H. Van Hamme, "Handling time-derivative features in a missing data framework for robust automatic speech recognition," in Proc. Int. Conf. Audio, Speech, Signal Process., 2006, pp. 293–296.
- [15] M. N. Schmidt and R. K. Olsson, "Linear regression on sparse features for single-channel speech separation," in Proc. IEEE Workshop Applicat. Signal Process. Audio Acoust. (WASPAA), 2007, pp. 26–29.
- [16] T. N. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran, "Bayesian compressive sensing for phonetic classification," in Proc. Int. Conf. Audio, Speech, Signal Process., 2010, pp. 4370–4373.
- [17] F. J. Damerau N. Indurkha, "An Overview of Modern Speech Recognition," in *Handbook of natural language processing*. London: CRC Press, 2009, ch. 15, pp. 339–366.
- [18] O.P. Prabhakar and N. Kumar Sahu, "A Survey On: Voice Command Recognition Technique," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, pp. 576–585, May 2013.
- [19] A. Lipeika, J. Lipeikiene and L. Telksnys, "Development of Isolated Word Speech Recognition System," *INFORMATICA*, vol. 13, no. 1, pp. 37–46, 2002.
- [20] S. Furui, "50 years of Progress in speech and Speaker Recognition Research," *ECTI Transactions on Computer and Information Technology*, vol. 1, no. 2, pp. 64–74, November 2005.
- [21] M. De Wachter, M. Matton, K. Demuynck, P. Wambacq, R. Cools and D. Van Compernelle, "Template-Based Continuous Speech Recognition," *Ieee Transactions On Audio, Speech, And Language Processing*, vol. 15, no. 4, pp. 1377–1390, may 2007.
- [22] J. Eisner, "An Interactive Spreadsheet for Teaching the Forward-Backward Algorithm," in *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Philadelphia, 2002, pp. 10–18.
- [23] K. Vesely, L. Burget and F. Grezl, "Parallel Training of Neural Networks for Speech Recognition," in *INTERSPEECH*, Chiba, 2010, pp. 2934–2938.
- [24] P.R. Dixon, T. Oonishi and S. Furui, "Harnessing graphics processors for the fast computation of acoustic likelihoods in speech recognition.," *Computer Speech and Language*, pp. 510–526, March 2009.

- [25] M. K. Rehman, M. U. Sarwar, M. R. Talib, M. S. Mansoor and M.B. Sarwar, "Parallel Implementation of Dynamic Programming Algorithm Using Graphics Processing Unit," *International Journal of Computer Science and Management Research*, vol. 2, no. 4, pp. 2097-2107, April 2013.
- [26] K. Gupta and J. D. Owens. (2010) Towards High-Quality Speech Recognition on Low-End GPUs.
- [27] K. You, J. Chong, Y. Yi, E. Gonina, C. J. Hughes, Y. Chen, W. Sung, and K. Keutzer, "Parallel Scalability in Speech Recognition," *Ieee Signal Processing Magazine*, pp. 124-135, November 2009.
- [28] A. Hannani and T. Hain, "Automatic Optimization of Speech Decoder Parameters ," *Ieee Signal Processing Letters*, vol. 17, no. 1, pp. 95-98, January 2010.
- [29] E. Stoimenov and T. Schultz, "A Multiplatform Speech Recognition Decoder Based on Weighted Finite-State Transducers.," in *ASRU 2009*, Merano, 2009, pp. 293-298.
- [30] O. Kalinli and S. Narayanan, "Continuous Speech Recognition Using Attention Shift Decoding with Soft Decision. ," in *interspeech*, Brighton, 2009, pp. 1927-1930.
- [31] M. Guezouri, L. Mesbahi and A. Benyettou, "Speech Recognition Oriented Vowel Classification Using Temporal Radial Basis Functions.," *Journal Of Computing*, vol. 1, no. 1, pp. 162-167, DECEMBER 2009.
- [32] J. Li and C. Lee, "Soft Margin Feature Extraction for Automatic Speech Recognition," in *INTERSPEECH 2007*, Antwerp, 2007, pp. 30-33.
- [33] B. Lecouteux, G. Linares, Y. Esteve and G. Gravier, "Dynamic Combination of Automatic Speech Recognition Systems by Driven Decoding.," *JOURNAL OF IEEE TRANSACTIONS ON AUDIO, SPEECH AND LANGUAGE PROCESSING*, vol. 1, pp. 1-10, November 2012.
- [34] Y. Park, S. Patwardhan, K. Visweswariah and S. C. Gates, "An Empirical Analysis of Word Error Rate and Keyword Error Rate," in *Proceedings of the International Conference on Spoken Language Processing*, Brisbane, 2008.
- [35] Speech corpus. The Free Dictionary. [Online]. Available: <http://www.thefreedictionary.com/corpus> . [Accessed: 01 March 2014].
- [36] M.C. Raja, "High Performance System in GPU and CUDA Media Processing System.," *International Journal of Advanced Research in Computer Science*, vol. 3, no. 3, pp. 45-50, May-June 2012.
- [37] R.Singh and B. Raj. ((2011), January) Design and Implementation of Speech Recognition Systems. document. [Online]. Available: <http://asr.cs.cmu.edu/spring2011/class1.19jan/Intro.class1.pdf> . [Accessed: 12 March 2014].
- [38] S.C. Harish, D. Balaji, M. Vignesh, P. Deepan Kumar, and V. Adinarayanan, , "Scope for performance enhancement of CMU Sphinx by parallelising with OpenCL," *International Journal of Wisdom Based Computing*, vol. 1, no. 2, pp. 43-46, August 2011.

- [39] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 1, pp. 4–37, 2000.
- [40] G. Potamianos, J. Luetin, and C. Neti, "Hierarchical discriminant features for audio-visual LVCSR," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Salt Lake City, UT, 2001, pp. 165–168.

APPENDICES

Appendix A

LPC_Autocorr: Test Sample Reading CPU LPC Implementation.

The following code list reflects the test function computed on the CPU.

```
// Part of CPU Based Implementation of the LPC (non-optimized)
const string *filename="/users/mark/trial";
const string *spSampleN="ss";
const string *aiPart="aip";
const string *fnlabel="FL";

// load speech sample from test file
speechSigform ReadSample()
{
    // read in annotation
    annotationList anlist;
    if (anlist.load(filename, spSampleN)!=0) {
        cerr << "Could not find annotations\n";
        exit(0);
    }
    // find annotation
    Annotation an = anlist.find(fnlabel);
    if (!an.name) {
        cerr << "Could not find annotation\n";
        exit(0);
    }
}
```

```

// load appropriate section
Signal isig(1,1.0);
isig.load(filename, spSampleN, aip.spsn, aip.cur_size);
// convert to waveform
speechSigform samplewv = makeCont(isig, 0.001);
return samplewv;
}

int main()
{
// graph formation
Graph gr(3,1,"LPC Spectrum");
// load sample waveform
speechSigform infile = ReadSample();
// windowing
isp.in = procHamming(infile_curr);
isg.rep(gr,1,"Speech Sample");
// check and apply string alignment
while[( i > 1) && (j > 1)] do
  if D[i,j] - sim_mat[bToInt(s[j - 1]), bToInt(t[I - 1])] = D[I - 1, j - 1] {
    t_aln := t[i-1].t_aln:
    s_aln := s[j-1].s_aln:
    i := i-1: j := j-1:}
  else if D[i,j] - gap_score = D[i,j-1] {
    s_aln := s[j-1].s_aln:
    t_aln := ' '.t_aln:
    j := j-1:}
  else if D[i,j] - gap_score = D[i-1,j] {
    s_aln := ' '.s_aln:
    t_aln := t[i-1].t_aln:
    i := i-1:}
  else
    cout<<"strings not of the same score\n";
  else if {D[t[i,j] = D[s[i,n]]}

// perform FFT on windowed signal
Complex speechSigform infile_curr = speechSigform +
                                     toComplex (speechSigform(infile));
LPCSpec isg.sps = ComplexFFT(infile);

// plot FFT spectrum
(sps.half()).repLogMag(gr, 2, "DFT Spectrum");

```

Appendix B

Part of CUDA Kernel Computations.

The following code list reflects the kernel function computed on the CUDA GPU.

```
// CUDA kernel function
void __global__ cudaMalloc ( float buffer , unsigned long speechSamples , float retVal ) {
const unsigned int i = blockIdx.x * blockDim.x + threadIdx.x ;
buffer [ i ] = retVal ;

_device_ float filter[ssamleSpect];
_global_ void convolve (float *ssampleSpect) {
_shared_ float sampleBlock[sectSize]; sectSize[threadIdx = ssampleSpect[i];
_syncthread(); ssampleSize[j] = resSpect; }

}
int  LPCspeechSol( speechWaveform x, int sampleOrder, LTISystem& ltis)
{
    // computing method autocorrelation
    double *r = new double[sampleOrder + 1];
    {
        int i = blockIdx.x*blockDim.x + threadIdx.x;
        if (i < sectSize)
            resSpect[i] = ssampleSize*x[i] + resSpect[i];
            double sampleAccum = 0;
            {
                for (int k = 1; k <= x.count() -i; k++)
                    sum += x[k] * x[k + i];
                rsample[i] = sampleAccum;
            }
    }

    // building a set of linear equations
    funEquset *mat = new resEquation[sampleOrder+1];
    if(i < sampleOrder)
        if (j = 1; j <= sampleOrder; j++)
            mat[i][j] = r[abs(i - j)];
            if(i=1; i <= sampleOrder; i++)
                spmat[i][sampleOrder+1] = -rsample[i];

    // free autocorrelation array
    cudaFree [] rsample;
```

```

// solving linear equation
if (MatrixSolve(spmat, sampleOrder) == 0) {
    cudaFree [] spmat;
    return -1;
}

// copy coefficients into LTI System
ltis = LTISystem(0, sampleOrder);
ltis.a[0] = 1.0;
if(I = 1; i <= sampleOrder; i++)
    ltis.b[i] = mat[i][order + 1];

cudaFree [] spmat;
return repSignal;
}

```

Appendix C

Additional Spectrums and Spectrograms for both CPU and GPU LPC Implementations.

The following are additional spectrums and spectrograms for the included (under the results) and other generated speech samples on both the CPU and GPU optimized LPC Implementations.

A GPU optimized LPC generated spectrum for speech sample C which was not included as part of the results.

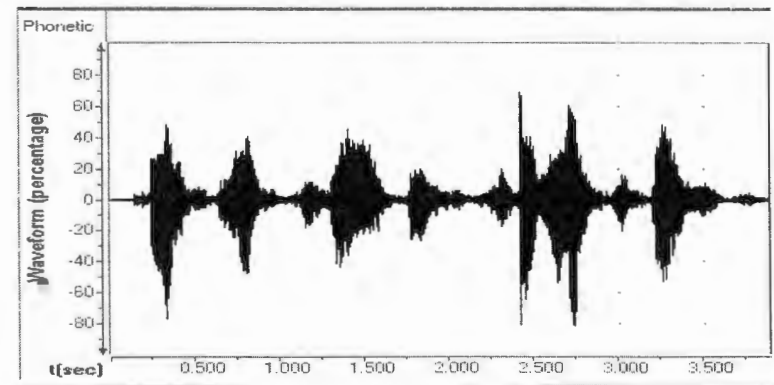


Fig. C0: GPU optimized LPC Spectrum for Speech Sample C.

Figures C 0.1 and C 0.2 represent spectrum and spectrogram generated from speech sample C indicated in figure C above.

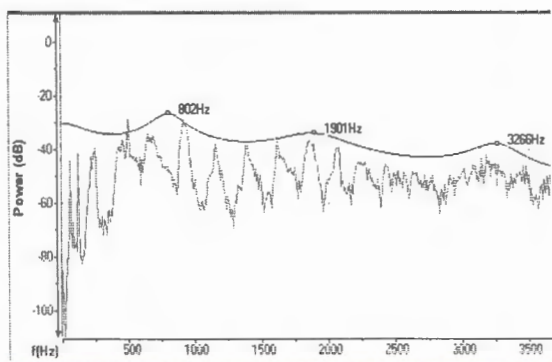


Fig. C 0.1: GPU Optimized Power Spectrum.

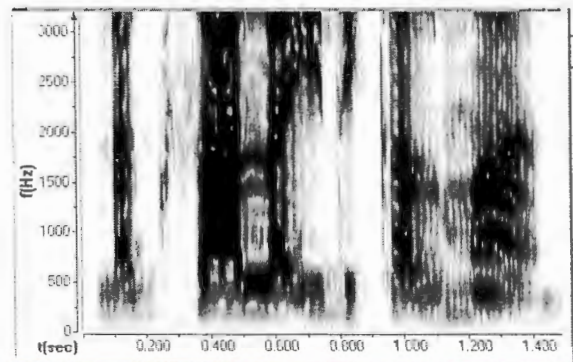


Fig. C 0.2: GPU Optimized Spectrogram.

A CPU optimized LPC generated spectrum for speech sample D which was not included as part of the results.

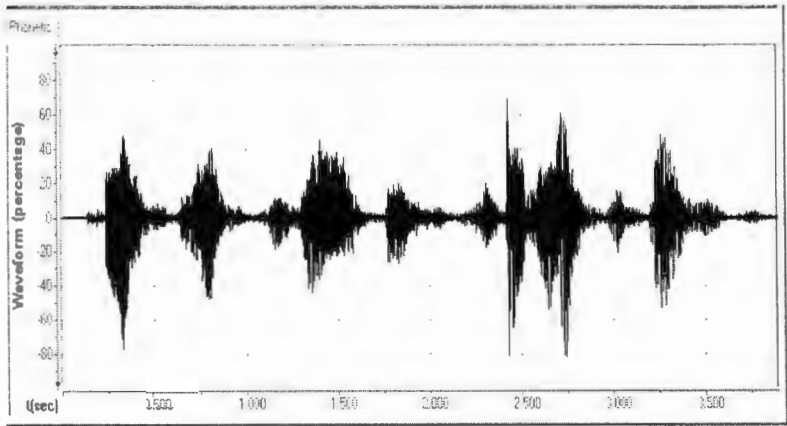


Fig. C1: CPU Optimized LPC Spectrum for Speech Sample D.

Figures C 1.1 and C 1.2 represent spectrum and spectrogram generated from speech sample D indicated in figure C1 above.

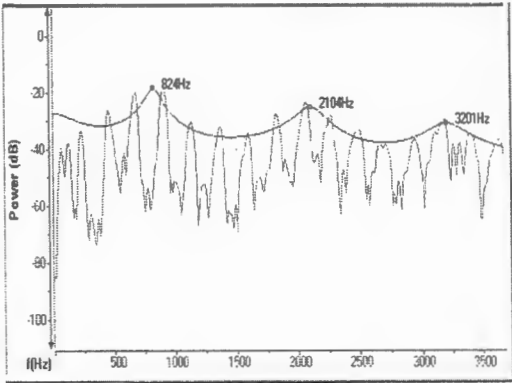


Fig. C 1.1: CPU Optimized Power Spectrum.

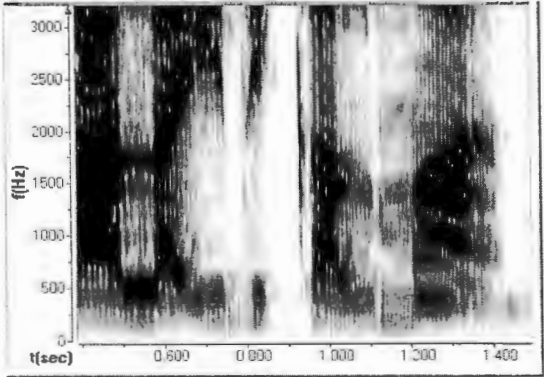


Fig. C 1.2: CPU Optimized Spectrogram.

Figures C2 and C3 below were generated from speech sample A which was included as part of the experimental results (please refer to Table 5.2 in Chapter 5) of this work.

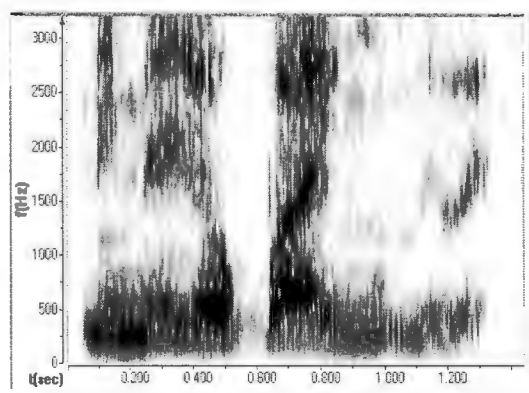


Fig. C2: GPU Optimized Spectrogram.

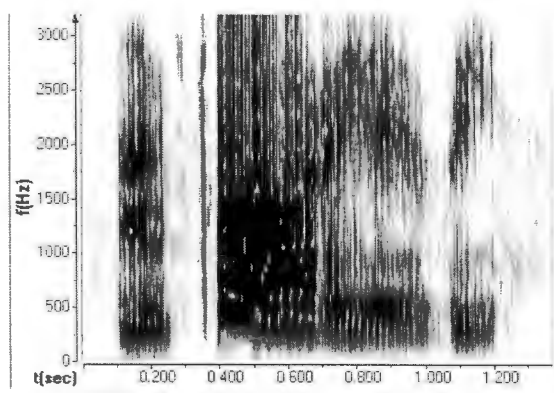


Fig. C3: CPU Optimized Spectrogram.

Figures C4 and C5 were generated from speech sample B which was included as part of the experimental results (Table 5.3 in Chapter 5) of this work.

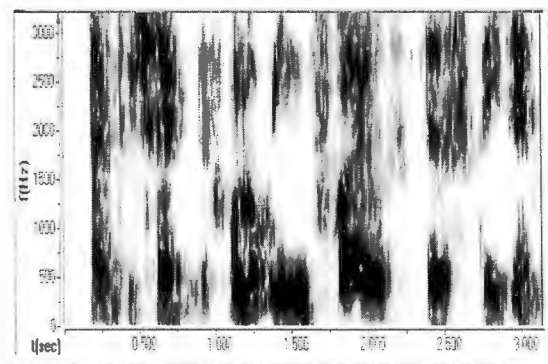


Fig. C4: GPU Optimized Spectrogram.

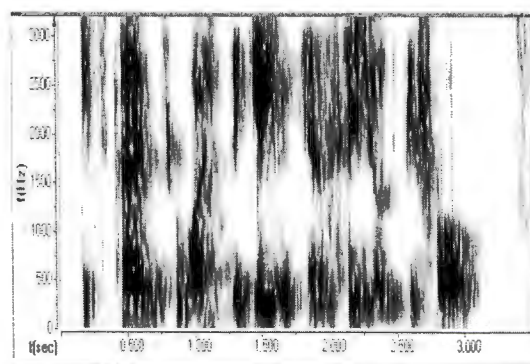


Fig. C5: CPU Optimized Spectrogram.