Mel Frequency Cepstrum Feature Extraction Using Graphical Processing Units

José F. Martínez, José F. Rodríguez, Sylmarie Ortiz
Advisors: Dr. Nayda G. Santiago & Dr. Pedro Torres
ECE Department, University of Puerto Rico at Mayagüez

Abstract

A system used by MIT Lincoln Labs for Speaker Recognition is being optimized using parallel programming techniques. The ‘front end’ consists of various speech signal processing algorithms that extract features, while the ‘back end’ analyzes these features in order to identify a possible speaker in the speech signal. Among the many algorithms in the ‘front-end’, our task is to improve the execution time of the Mel Frequency Cepstrum Coefficients (MFCC) Feature Extraction algorithm. The experiment is based on comparing our C implementation where feature extraction of the frames execute sequentially to our CUDA C implementation where execution is concurrent. Early results show that many steps of algorithm can execute multiple times faster in CUDA C: framing & windowing of a speech signal executes 40 times faster, the FFT step executes 60.49 times faster, while computing the DCT can execute as much as 64.35 times faster.

Introduction

The speaker recognition system from MIT Lincoln Labs has had the back-end portion optimized while the front-end has become the bottleneck of the system.

![MIT Lincoln Lab Speaker Recognition System Pipeline](image1)

In order to improve the execution time of the front-end, each of the algorithms has to be optimized. In our case we needed to implement the Mel Frequency Cepstrum Coefficients (MFCC) Feature extraction.

Background

The speech signal is converted into a vector and copied to the device. Consequently the vector is divided into multiple frames, each one yielding its own vector of Mel Frequency Cepstrum Coefficient features. Thus, since each frame is independent of one another, the algorithm has a high degree of parallelization.

CUDA’s architecture and programming tools allows us to execute each frame in parallel. It is shown by previous research of the algorithm there has been reported 90-97x speed increase of the sequential algorithm when using a CUDA technology with a graphics processing unit [2].

Compared to our implementation we used Nvidia CUDA Compiler version 6.5 and the Fast Fourier Transform (FFT) library from the CUDA Toolkit, while the previous research mentioned used an implementation of the Discrete Fourier Transform from the Spiral Project.

Methodology

Research was done on how the algorithm works, basically the steps required to extract MFCCs from a speech signal are expressed in Figure 2.

![MFCC Algorithm Steps](image2)

Essentially to implement the algorithm we would follow the following steps:

- Implement the version in MATLAB
- C sequential implementation
- Parallel Implementation in CUDA C
- Fine tuning the CUDA C Implementation

Future Work

Finish the implementation of the Mel-based Frequency Filterbank step. Fine tune the CUDA C implementation to the GPUs used by MIT Lincoln Labs.

Experiments and Results

For the experiment, we compared the execution time of each step of the algorithm in both the C and CUDA C Implementation. Parameters for the MFCC algorithm can be seen in Table 2 and the comparison between the execution times and speed up in Table 3.

Conclusion

As seen from our experiments, in fact the MFCC feature extraction steps for “Framing & Windowing”, “FFT” and “DCT” display a considerable increase in execution time.

Table 2 – MFCC Feature Extraction Configuration

<table>
<thead>
<tr>
<th>Sampling Frequency</th>
<th>8000 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Filters</td>
<td>20</td>
</tr>
<tr>
<td>Frame Size</td>
<td>25 ms</td>
</tr>
<tr>
<td>Step Size</td>
<td>10 ms</td>
</tr>
<tr>
<td>Speech Duration</td>
<td>05:10 min</td>
</tr>
<tr>
<td>Data points</td>
<td>2,479,616 (floating data points)</td>
</tr>
<tr>
<td>Total Frames</td>
<td>30,993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>C Implementation (ms)</th>
<th>CUDA C (ms)</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framing &amp; Windowing</td>
<td>114.03</td>
<td>2.8721</td>
<td>X39.70</td>
</tr>
<tr>
<td>FFT</td>
<td>77.98</td>
<td>1.2890</td>
<td>X60.49</td>
</tr>
<tr>
<td>DCT</td>
<td>742.11</td>
<td>11.533</td>
<td>X64.35</td>
</tr>
</tbody>
</table>

Acknowledgments

Presentation of this work has been supported in part by NSF Grant CNS-1042341.

References