CANADA: Continuous Adaptive Non-Linear Analysis for Driving Applications

By:

Blake Brogdon
Thomas Deitch
Kyle Barnhart
Britt Antley

CANADA: Continuous Adaptive Non-Linear Analysis for Driving Applications

By:
Blake Brogdon
Thomas Deitch
Kyle Barnhart
Britt Antley

Online:
< http://cnx.org/content/col10499/1.2/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: January 23, 2008
PDF generated: February 4, 2011
For copyright and attribution information for the modules contained in this collection, see p. 16.

Table of Contents

Chapter 1

Introduction[1]

1.1 Introduction

As personal computer applications become increasingly complex and personal computers themselves become increasingly powerful multi-process machines, the issue of application control becomes an important issue. While the keyboard and mouse previously reigned as the de facto computer inputs, the capability of today's computers to do fast, adaptive signal analysis gives way to new and exciting control opportunities.

Most computers are equipped with basic microphones. Though the idea of spoken word or sound control is not a new idea, most voice control systems require complex algorithms to filter noise and account for the many differences between peoples' voices. Our project takes advantage of the simple frequency patterns in whistles in order to simplify this process.

To handle signal processing, we chose to utilize the features of the popular LabView development environment. Combined with the powerful Java programming platform, we are able to send control commands in order to drive an application, which in our case was iTunes. In the scope of our project, LabView instantiates a Java control program which then passes commands onto iTunes.

[1] This content is available online at <http://cnx.org/content/m15674/1.2/>.

Chapter 2

Algorithm Overview[1]

2.1 Algorithm Overview

The first step in detecting a signal is to input it into the system. Since we are using an audio signal, a microphone is the obvious choice. However, the desired control signal is not the only sound in the room. There is also the music that is being played through the speakers as well as outside noise. Unfortunately, there is not much that can be done about the random noise that is present in the room. However, there are tools available that allow us to minimize the interference caused by the music. Specifically, we can use an adaptive filter to mimic the room's effect on the output of the sound card, providing us with an estimate of the music's contribution to the signal received by the microphone. We can subtract the microphone's signal from this estimate in order to obtain—hopefully—only the whistle.

Figure 1 (Figure 2.1: Our System's Block Diagram) shows the block diagram of our system. All of these components fall into four main categories:

1. Signal acquisition
2. Whistle isolation
3. Whistle frequency analysis
4. iTunes interface

The acquisition phase is the top portion of the diagram, the whistle isolation system is represented by the $\hat{h}$ box and the band pass filter, the rest of the diagram (except for the Java controller) comprise the whistle analysis phase, and the Java controller is the iTunes interface.

[1] This content is available online at <http://cnx.org/content/m15673/1.2/>.

Our System's Block Diagram

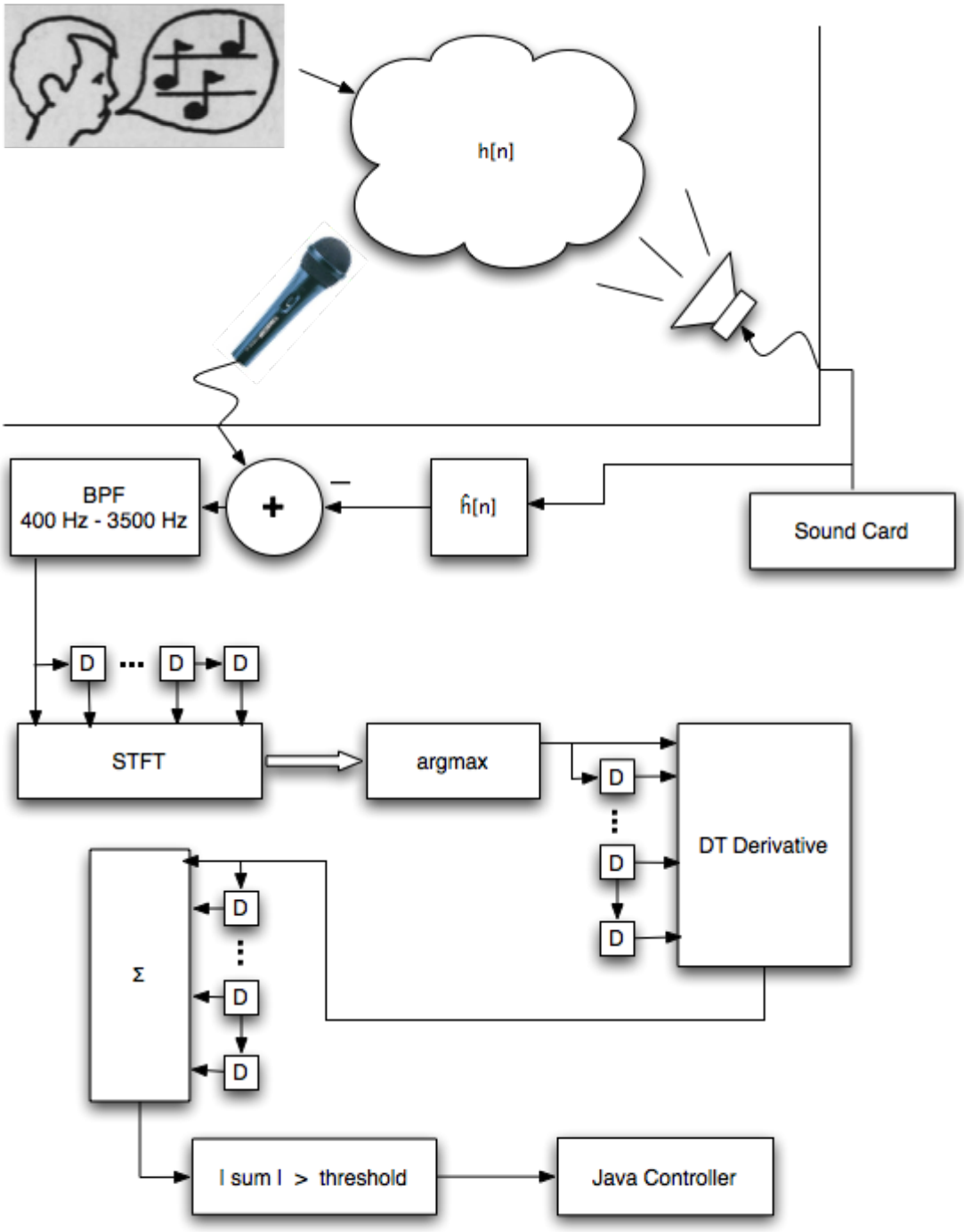

Figure 2.1: In our system, the sound is output through the speaker, and the microphone receives the music and whistles while the sound card receives the audio without the room affecting it. We then remove the music and process the whistle.

After isolating the whistle, we apply a band pass filter whose pass band corresponds to common whistle frequencies in order to remove extraneous noise outside of these whistle frequencies.

We then take the Short Time Fourier Transform in order see how the frequency components of the whistle change over time. If the frequency of the whistle is increasing iTunes should advance to the next track, and a decreasing frequency will skip to the previous track. To accomplish this, we examine the frequency with maximum power (the argmax in the figure below) and accumulate several readings of this frequency. In order to see if this function is increasing or decreasing we take the derivative and examine its average value. If the average value is positive, the function must have been increasing and the whistle must have been from high to low frequencies.

Chapter 3

Least Mean Squares Adaptive Filters[1]

3.1 Least Mean Squares Adaptive Filters

3.1.1 Why LMS?

In our system, the music that is being played by the computer is a known output that is being fed into the room, which has an unknown response, and picked up by the microphone. Since we have the signal both before and after the influence of the room, this is a candidate for an adaptive filter. The adaptive filter can be used to try to estimate the room's response based on the music being outputted and the input to the microphone. Once an estimate is available, we can use it to remove the interference of the music from the input to the microphone.

3.1.2 LMS Overview

One of the most popular adaptive algorithms used today is the Least Mean Squares (LMS) algorithm. Essentially, this algorithm attempts to minimize the error that occurs between the detected (or desired signal), d[n], and the estimated value of the signal, y[n]. This estimated signal is created by taking the original input, x[n] , and running it through an approximation of the unknown channel. Basically:

$y[n] = x[n] * \overset{\Theta}{h}[0] + x[n-1] * \overset{\Theta}{h}[1] + \cdots + x[n-N] * \overset{\Theta}{h}[N]$

or

$y[n] = x_N^T[n] * \overset{\Theta}{h}[n]$

where N is the order to which you are approximating the unknown system, x_N is a vector of the last N value of x, and each h[n] is a weight.

[1] This content is available online at <http://cnx.org/content/m15675/1.2/>.

LMS Algorithm Block Diagram

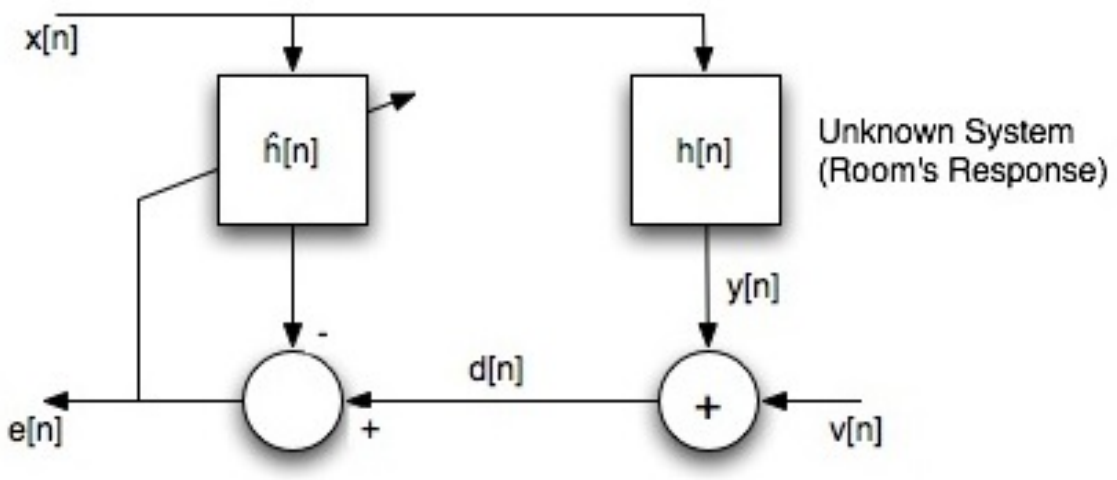

Figure 3.1: x[n] is the input to the filter, v[n] is the interference in the room, h[n] is the impulse response we're modeling, and e[n] is the error in this modeling.

Once the signal and its approximation are found, the error is just the difference between the two signals at the current point in time.

$e[n] = d[n] - y[n]$

Using the error, we can approximate the next set of weights as follows:

$\overset{\Theta}{h}[n+1] = h[n] + \mu * e[n] * x_N[n]$

where μ is a constant.

3.1.3 Choosing μ

The choice of μ is an important one as it greatly affects how the system will perform. Small values allow for greater precision in convergence, but slow down the response time of the system. If the value is too small, then the adaptive filter will not adapt fast enough. If it is too large, the system will not converge at all, and the value weights will actually diverge. Experimentation with different systems and environments is necessary to choose the ideal value for μ.

3.1.4 Isolating the Whistle

It can be shown that the error value will converge on a minimum point. This theoretical minimum will likely not be at zero error, since there is outside noise and interference, but it represents the original signal's effect on d[n] being completely removed. In the case of our signal detection system, the whistle that we are trying to detect will still be in the e[n] and therefore we will use e[n] as the input to the rest of our system. Once the adaptive filter has settled down, this signal will be the input to the microphone minus the estimated value of the music when it reaches the microphone.

Chapter 4

Joint Time-Frequency Analysis[1]

4.1 Joint Time-Frequency Analysis

4.1.1 Short Time Fourier Transform

After our system has isolated the whistle sound, it is input to a continuously running analysis algorithm based on the result of a Short-Time Fourier Transform. While the traditional Fourier transforms do not maintain a sense of time, the STFT allows for joint time-frequency analysis. The formula for the STFT is:

$\text{STFT}\{x[n]\} = X(m,w) = \sum_{n=-\infty}^{\infty} x[n]\, w[n-m]\, e^{-\text{jwn}}$

Individual sections of the signal are windowed and their FFT is taken. The result of this operation is a two dimensional function in terms of the offset from zero ("time") and the frequency content of the signal. According to the Heisenberg Uncertainty Principle, however, we cannot have an arbitrary amount of resolution in both the time and frequency domains. For our application, we are only interested in detecting trends in the frequency with highest power, so time resolution is less important.

4.1.2 Spectrogram

Squaring the magnitude of the STFT results in a 3-D function is known as a spectrogram. The spectrogram of a whistle whose pitch goes from low to high looks like this.

[1] This content is available online at <http://cnx.org/content/m15676/1.2/>.

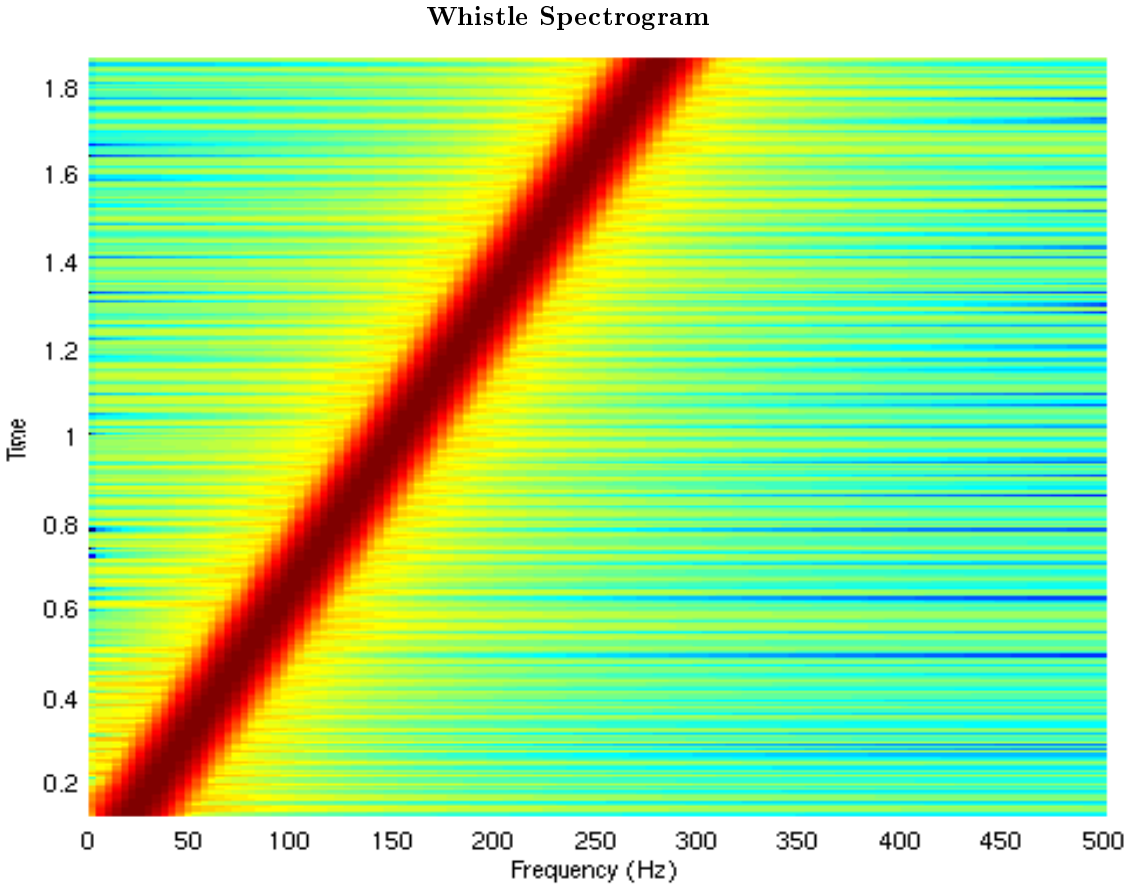

Figure 4.1: This is a whistle of increasing frequency. Each slice of the spectrogram is an FFT of a certain windowed chunk of the signal. This embeds a sense of time in the spectrogram.

Notice that there is a dominant power (dark red) that shows a very clear upward trend over time. Each slice of the frequency axis at a particular time corresponds to a single chunk of the signals' FFTs. The dark red corresponds to the maximum of one of these FFTs, and as time passes (in the positive vertical direction), we see the frequency of highest power increases. Below is a graph of three of these component FFTs, illustrating how the peak frequency increases across time. Since pitch and frequency are essentially synonymous, we can determine what kind of whistle is input by looking at trends in the frequency with the dominant power.

Power vs. Frequency for 3 Spectrogram Components

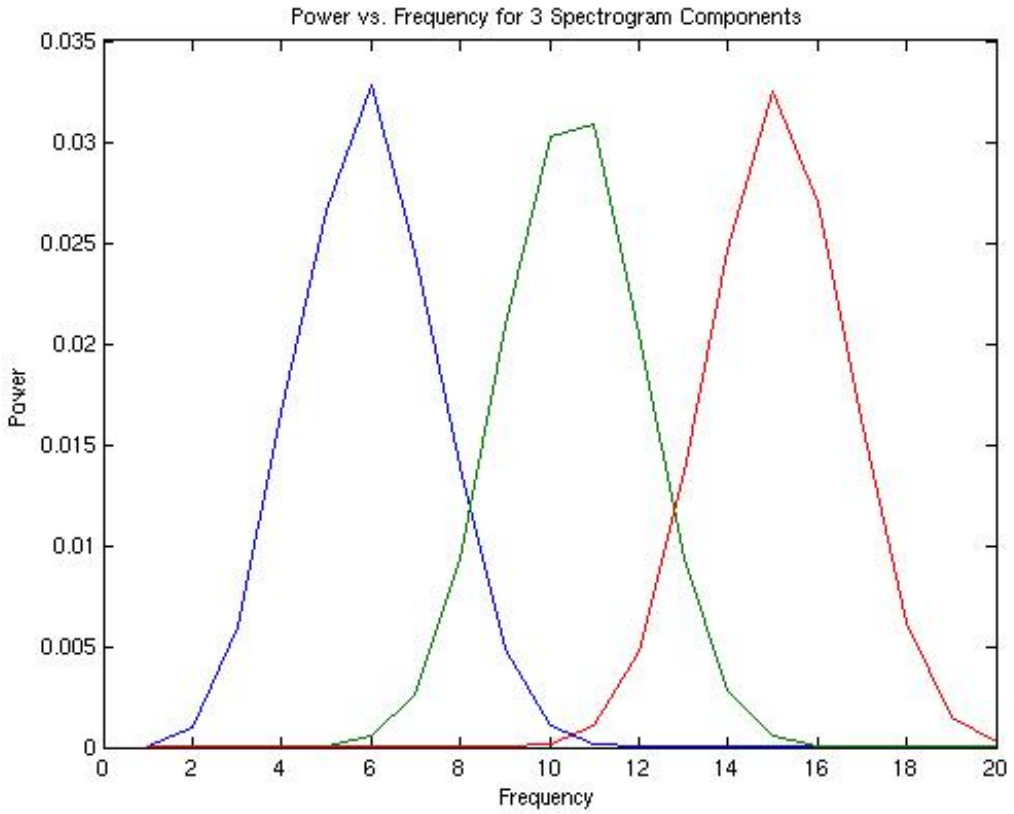

Figure 4.2: Here are 3 of the FFTs that comprise the spectrogram above. The frequency of maximum power is increasing.

With the dominant frequency for any given windowed input now known, our system then takes the discrete-time derivative of that frequency over the window. Calculus tells us that the derivative of a function at a point is positive for increasing functions and negative for decreasing functions. By continuously taking derivatives of these windows our system can track the basic shape of the spectrogram.

4.1.3 Analysis using the STFT

Our analysis algorithm keeps a running buffer of the signs of these discrete-time derivatives. It then takes the magnitude signs inside the buffer. If the buffer encounters a number of continuous signs above or below a certain threshold, either positive or negative, it concludes that the input is a whistle with increasing or decreasing pitch, respectively. Finding the optimal threshold value is mostly trial and error; we looked at recordings of sine waves with constant frequencies and white noise in order to determine a reasonable upper bound for the area under the derivative curve.

There is a tradeoff between the size of the buffer and the quality of the analysis. Given that a whistle may last up to three seconds, it's clear that the buffer need not contain that many samples in order to find

and characterize the whistle. But on the same token, too few samples will result in an unusual number of false positives and change the song without any user interaction. And while this is not necessarily complex math, doing it quickly and continuously requires the window be as small as possible.

Chapter 5

iTunes Control Interface[1]

5.1 iTunes Control Interface

To accomplish anything useful with our signal processing, we need to drive an application. We chose iTunes, a popular multiplatform program for playing digital music files, as our example application for several reasons. First, the ability to control iTunes through external means is well documented and fairly simple to achieve. More important, however, is the aspect of music or audio playback in a room while simultaneously listening for whistles.

5.1.1 Java-COM

iTunes is both the system to control as well as the dominant source of noise to combat. Based on the profile of the whistle detected, whether it be increasing or decreasing in frequency, we will skip to the next or previous track in iTunes. To accomplish this, LabVIEW makes a call to a Java program that interacts with iTunes by accessing the Component Object Model (COM) interface of Windows. The JACOB[2] library serves as an intermediary to complete this task. JACOB[3] is a native Java-COM bridge, allowing Java programs to simultaneously run and interact with system applications. Once the COM interface receives instructions from LabVIEW, it then interacts with iTunes and changes tracks appropriately.

[1] This content is available online at <http://cnx.org/content/m15677/1.2/>.
[2] http://danadler.com/jacob/
[3] http://danadler.com/jacob/

Chapter 6

Conclusion and Possible Improvements[1]

6.1 Conclusions and Possible Improvements

While we did not create a fully working prototype, we believe that our algorithm is a solid approach for implementing a solution to this problem. While trying to implement the whistle detection in LabVIEW, we were faced with obtaining synchronized samples from two audio input sources while outputting through another. In order to assure that all samples are read in at the same time, we would need to use real time operating system or at least the real time LabVIEW plugin. We believe the best solution to this problem would be to implement the system in hardware where we would have complete control over sample acquisition.

We were able to implement the LMS filter in MATLAB, but its running time was too long to be a feasible solution. Its running time appeared to be O(n^2) and running the filter on a 10 second clip from a song took 4 hours to process. Again, since the LMS filter is such a popular adaptive filter, more efficient implementations must exist; and implementing it in hardware would surely provide a significant performance improvement.

Finally, one of the most important advantages of our design is that it's modular. While we chose to implement a rising/falling frequency detector, any type of aural recognition could be implemented with this system. For example, we could replace the STFT and increasing/decreasing detector with a voice recognition system.

[1] This content is available online at <http://cnx.org/content/m15678/1.2/>.

Attributions

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.