Instrument and Note Identification

By:
Michael Lawrence

Instrument and Note Identification

By:
Michael Lawrence

Online:
< http://cnx.org/content/col10249/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: December 14, 2004
PDF generated: October 25, 2012
For copyright and attribution information for the modules contained in this collection, see p. 29.

Table of Contents

Chapter 1

Introduction and Background

1.1 Instrument and Note Identification[1]

1.1.1 Introduction

The final product utilizes several core signal processing concepts. Some are interesting and non-intuitive; others are straight-forward and common-sensical. Before we get into the nitty gritty details of the project's inner workings, we would be served well to familiarize ourselves with these core concepts.

With that in mind, please navigate (using the links on the left) to whichever page has a title unfamiliar to you. As a nod to our superiors and as an attempt to make use of this site in its full glory, we include many modules written by those more knowledgeable than us. The direct applications will be discussed when the project itself is specifically addressed.

1.2 Simple Music Theory as it relates to Signal Processing[2]

1.2.1 Simple Music Theory

For those of you unfamiliar with music, we offer a (very) brief introduction into the technical aspects of music.

The sounds you hear over the airwaves and in all manner of places may be grouped into 12 superficially disparate categories. Each category is labeled a "note" and given an alphasymbolic representation. That is, the letters A through G represent seven of the notes and the other five are represented by appending either a pound sign (#, or sharp) or something that looks remarkably similar to a lower-case b (also called a flat).

Although these notes were conjured in an age where the modern theory of waves and optics was not dreamt of even by the greatest of thinkers, they share some remarkable characteristics. Namely, every note that shares its name with another (notes occupying separate "octaves," with one sounding higher or lower than the other) has a frequency that is some rational multiple of the frequency of the notes with which it shares a name. More simply, an A in one octave has a frequency twice that of an A one octave below.

As it turns out, **every** note is related to every other note by a common multiplicative factor. To run the full gamut, one need only multiply a given note by the 12th root of two n times to find the nth note "above" it (i.e. going up in frequency). Mathematically:

(nth note above base frequency) = (base frequency)$2^{\frac{n}{12}}$

[1] This content is available online at <http://cnx.org/content/m12462/1.1/>.

[2] This content is available online at <http://cnx.org/content/m12461/1.5/>.

1.2.2 Harmonics

The "note" mentioned above is the pitch you most strongly hear. Interestingly, however, there **are** other notes extant in the signal your ear receives. Any non-electronic instrument actually produces many, many notes, all of which are overshadowed by the dominant tone. These extra notes are called **harmonics**. They are responsible for the various idiosyncracies of an instrument; they give each instrument its peculiar flavor. It is, effectively, with these that we identify the specific instrument playing.

1.2.3 Duration and Volume

We will also make a quick note (no pun intended) for the other two defining characteristics of a musical sound. **Duration** is fairly self-explanatory; notes last for a certain length of time. It is important to mention that in standard music practices most notes last for a length of time relative to the **tempo** of the music. The tempo is merely the rate as which the music is played. Thus, by arbitrarily defining a time span to be equal to one form of note duration we may derive other note durations from that.

More concretely: taking a unit of time, say one minute, and dividing it into intervals, we have **beats per minute**, or **bpm**. One beat corresponds, in common time, to a quarter note. This is one quarter of the longest commonly-used note, the whole note. The length of time is either halved or doubled to find the nearest note duration to the base duration (and so on from there). The "U.S. name" for the duration of the notes is based on their fraction of the longest note. Other, archaic, naming conventions include the English system replete with hemi demi semi quavers and crotchets (for more information, follow the supplemental link on the left of the page).

Volume, on the other hand, is based on the signal power and is not so easily quantifiable. The terms in music literature are always subjective (louder, softer) and volume-related styles from previous eras are heavily debated ("but certainly Mozart wanted it to be louder than **that**!"). For our project, we save the information representing the volume early on, then normalize it out of the computations to ease the comparisons.

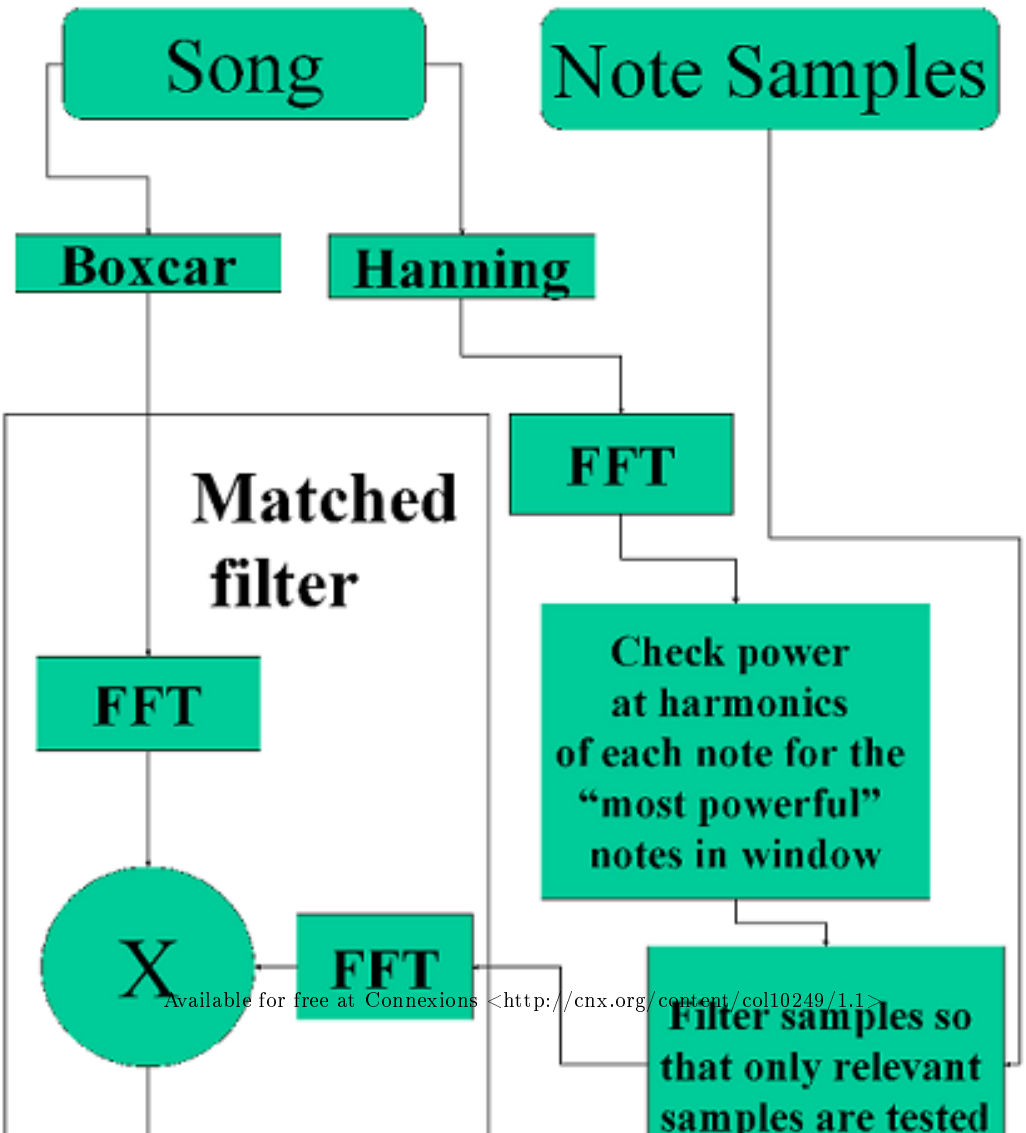
Chapter 2

Methodology

2.1 Note Recognition[1]

2.1.1 Macroscopic View

General-View Block Diagram



This project attempts (and, for the most part, succeeds) to identify a single instrument lost among a barrage of other instruments. More than that, it attempts to identify which sequence of notes the instrument is playing, the volume at which it plays them and the duration of time for which the instrument plays.

The theory is relatively simple (indeed, we learned it in an introductory course). For the instrument recognition to work, we must first have a sample of that instrument playing. Ideally, we would need only one sample from which we could derive all the others using the one-dimensional application of a Mellin-Fourier transform. Considerations of time, however, caused us to forgo this option. We instead approached the collection of samples as a good communist would; with great emphasis on labor. For the purposes of this project, 33 samples (i.e. notes) of a clarinet playing were recorded.

Each of these samples was then matched against the inputted waveform to measure correlation. The algorithm for accomplishing this task is as follows:

Correlation Algorithm

1. If it is too large, "chop" the inputted waveform (henceforth referred to as "signal," an all-encompassing term) into smaller, easier-to-handle chunks.
2. Input each of those chunks into a program which takes the Fourier transform of both the signal and the samples, multiplies them, and then inverts them back into the time-domain (i.e. convolves the two signals).
3. Based on various thresholds and numerous considerations, choose the sample which most closely matches the signal (i.e. read off the highest peak and assign it a value; if that value is high enough, select it as the representative sample).
4. Output the data in a user-friendly fashion.

The implementation of the second step is called a "matched filter."

The remainder of this course will focus on the four steps of the correlation algorithm.

2.2 Deconstructing a Signal[2]

2.2.1 Deconstructing the Signal

The relevant portion of the block diagram

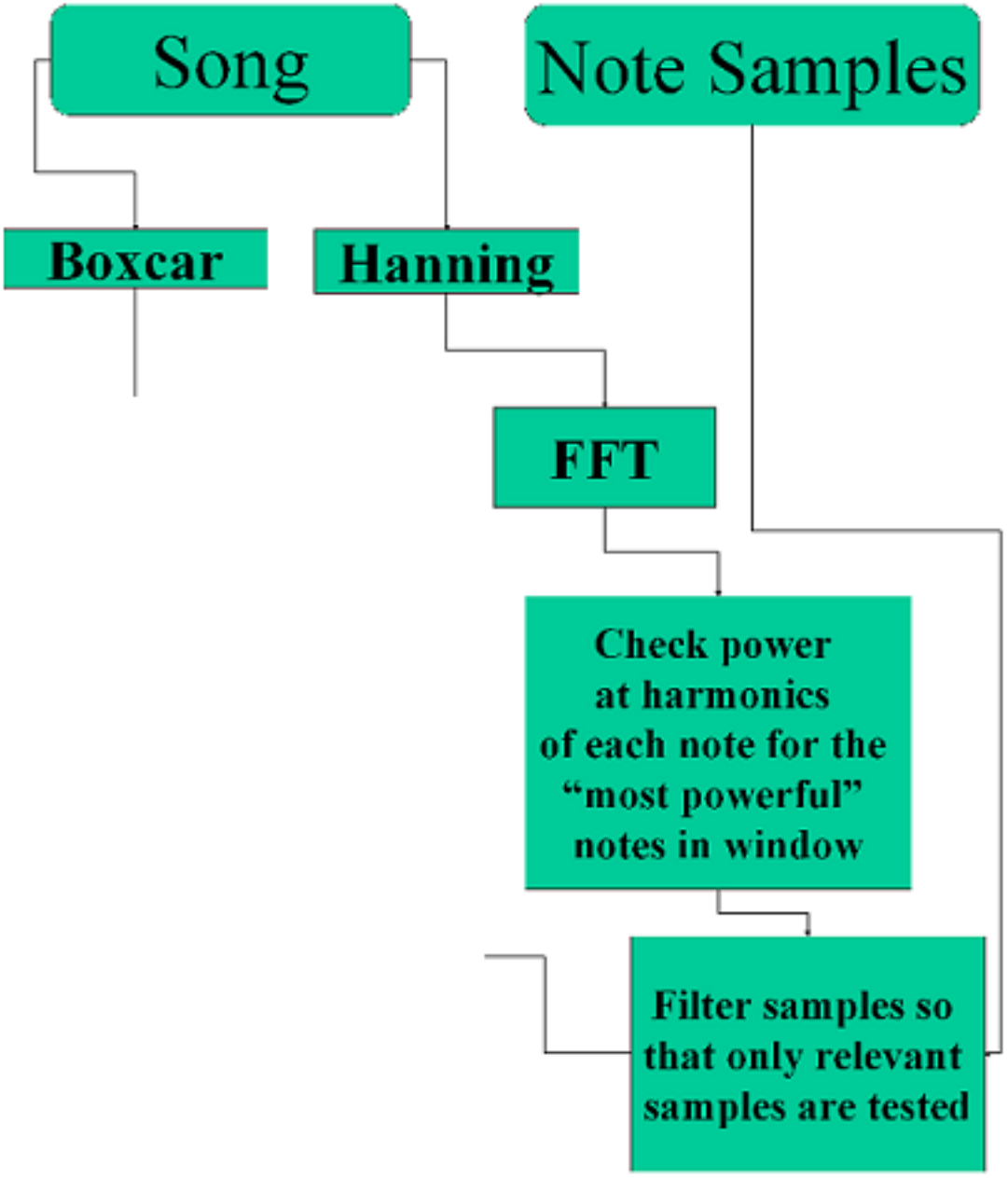


Figure 2.2: The signals must be tailored for the Matched Filter.

[2]This content is available online at <http://cnx.org/content/m12463/1.3/>.

The main program for our project requires six separate inputs. Two are the signal and the samples mentioned above. The other four are, in no particular order, the size of the pieces into which you want the signal broken, the size of the window looking at each piece, the rate at which the music is sampled from its continuous origins and the maximum number of notes the user expects to find playing at one time.

The first step of the correlation algorithm, deconstructing the signal, is concerned with the signal, the size of the pieces and the size of the window. The data from the signal is broken into several smaller chunks by boxcar filtering the original signal and storing the information in a new matrix. Each row of the new matrix corresponds to the data from each window. It is worth noting that each chunk does not overlap its neighbor and is left otherwise unaltered, save for the case of the final chunk to which zeros are appended if it is not of the correct size.

This data is then treated as a new signal; this helps speed the processing time. Each row of the new matrix representing the chunks is then divided into yet smaller windows for analysis. These windows may be either Hanning windows or boxcar windows, depending on the analysis to follow (two separate analyses occur to maximize note recognition and computational time). These windows **do** overlap.

At this point, the boxcar windows are ready to be sent to the next step. The Hanning windows, however, must first undergo a treatment and recognize their purpose. From the matrix extant after processing the original signal into Hanning windows, the power of each row is computed and weighed against all possible note values (that is, an entire octave; not the full range of the instrument). The most powerful result is placed in a new matrix and fed back to the parent function. This selection of a specific note means that the next stage of the correlation algorithm will need only check separate octaves of one note instead of running through the full range of an instrument (a very tedious and cycle-consuming proposition).

As a further clarification and justification, we shall rehash the difference between the Hanning and boxcar windows. All windows are initially filtered through a boxcar window in order to best preserve the data (we are not interested in reconstructing the signal). That information then goes two places: the matched filter and the note-recognizing algorithm. The latter set is put through a Hanning window for ease of analysis; low frequency signals are better preserved in a Hanning window. It is from that analysis that the one note (in a given octave) to test is chosen.

2.3 Matched Filter[3]

2.3.1 Matched Filter

The relevant portion of the block diagram

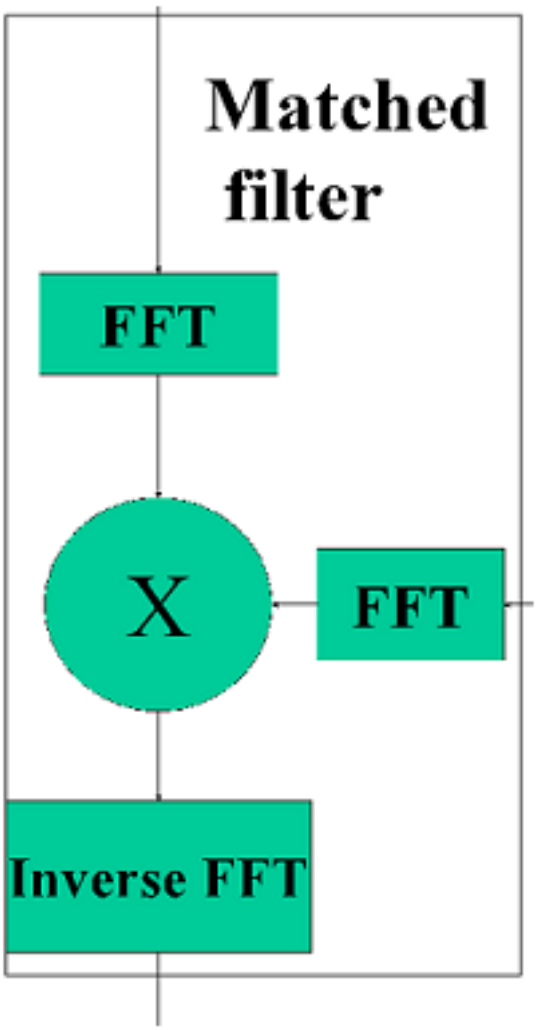


Figure 2.3: The Matched Filter.

The matched filter segment of our project is perhaps the easiest to code but the most difficult conceptually. An understanding of Fourier analysis, convolution and manipulations in frequency domain is required. Although we made available several supplementary links at the beginning of the course, a brief recapitulation of the basic theory is warranted.

[3]This content is available online at <http://cnx.org/content/m12464/1.3/>.

2.3.1.1 Fourier Analysis

Jean Baptiste Joseph Fourier[4] discovered, approximately 200 years ago, that any function could be represented by weighted averages of any periodic base function. The most common application of Fourier analysis is through the use of sinusoidal waves, but using a sinusoid as a basis is not necessary - any periodic function will do. This weighing was perceived as a series of coefficients, each coefficient corresponding to a specific frequency of the periodic base function. From consideration of these coefficients, it was realized there existed a **frequency domain**. Thus, there was a medium in which discussion of a non-periodic function's frequency made sense. This discovery has had many impressive results, not the least of which is the concept of a matched filter.

Carl Friedrich Gauss[5] later expounded on Fourier's realization and found what is the now-known fastest algorithm for computing Fourier coefficients. Unfortunately for 20th century scientists, no one remembered Gauss developed the algorithm until **after** it had been rederived.

2.3.1.2 Convolution

Convolution is at the heart of matched filters. One can easily visualize placing a transparancy of an image over another image to gauge the similarity of the two images. Convolution does essentially just that with two functions; it places one function over another function and outputs a single value suggesting a level of similarity, then it moves the first function an infinitesimally small distance and finds another value. The end result is a graph which peaks at the point where the two images are most similar.

Another method of attack may be seen through the Cauchy-Schwarz Inequality. This[6] page does an enviable job of approaching matched filtering from that angle.

The concept that binds these two seemingly disparate topics is almost startling in its beauty (for those of us who have ever had to perform convolution integrals). Convolution in the time domain (the domain in which we would want to compare two signals) just happens to be multiplication in the frequency domain. And, as it turns out, performing two Fast Fourier Transforms (FFTs), multiplying the results, and then performing two Inverse FFTs (IFFTs) is computationally faster than performing one convolution between the two signals and provides the same result.

[4] http://en.wikipedia.org/wiki/Jean_Baptiste_Joseph_Fourier
[5] http://en.wikipedia.org/wiki/Carl_Friedrich_Gauss
[6] "Cauchy-Schwarz Inequality" <http://cnx.org/content/m10757/latest/>

2.3.2 Summary

A matched filter compares two signals and outputs a function describing the places at which the two signals are most like one another. This is accomplished through FFTing two signals, multiplying their coefficients and IFFTing the result.

2.4 Matched Filter Output Analysis[7]

The relevant portion of the block diagram

Figure 2.4: The final analysis.

2.4.1 Processing

After having performed the actual matched filtering, we need to make sense of the data. Unfortunately, matched filters provide only an insight as to what **might** be the best answer; it cannot definitively say "yes, this is what you want." Without telling our program to ignore signals of little power, even the noise inherent in all recordings will match some specific value better than all other values. With that in mind we created two threshold constraints and validated our results from the matched filtering.

2.4.1.1 Validation Techniques

Perhaps the easiest to detect, the volume level affects results most significantly. If the sample's amplitude is substantially different than that of the signal, matching may not even occur. Therefore, our first technique is merely to normalize the volume between the sample and the signal. Important information is lost; we therefore record the volume for later reference.

The second technique involves determining a threshold value for noise. The question to be answered: "What is the level of power below which one may consider the signal silent?" Unfortunately, we found no sound method (pun intended) through which we could automatically find this power level. We therefore just listen to the signal and input the threshold accordingly. Fortunately, this threshold value affects only the outputting of the data (which will be discussed in the next section) and does not affect the integrity of our algorithm.

Also of concern is the degree to which a particular harmonic's strength weighs into the consideration of a note value. This becomes especially troublesome when one searches for multiple notes playing at a single moment in time. That is to say: based on what we know about a single note producing multiple harmonics,

[7] This content is available online at <http://cnx.org/content/m12466/1.4/>.

we must single out which harmonic is the true note being played. For one note playing at one time, this is simple: choose the largest peak and be done with it. For multiple notes, however, one must define yet another threshold; the threshold above which a possible secondary (or tertiary, or... well, you get the idea) harmonic transcends its lowly status and becomes the fundamental harmonic of a separate note. The value for this threshold is even more ambiguous in nature than the silence threshold. Lacking an automated routine to calculate this threshold is perhaps the most significant weakness in our program in terms of scaling the number of recognized notes to map the music from an entire orchestra.

Once one has completed the various methods of validation, one must put the information together in a coherent manner. Using the ever-versatile language of Matlab, we create a three-dimensional matrix to hold the relevant information before further condensing it into two dimensions. One dimension of the three-dimensional matrix is the number of samples against which the signal is checked; another is the number of the window against which the samples are checked; the third dimension is the number of data points being checked. So, for any given spot in the matrix, one can read off the data point tested, the window in which it was tested, and the score it received as a result of the testing. Because one checks only against a certain octave-set for matching (due to the Hanning window algorithm), most entries in this three-dimensional matrix are zero. The two-dimensional matrix simply integrates all of the windows into one coherent whole (remember, the windows overlap; thus this final form is not perfectly analogous to the original signal).

This information is the output of our primary program, ProjectD.m. All of the relevant data is stored in the outputted two-dimensional matrix. However, the formatting is such that it is difficult to understand from viewing only the entries. With this in mind, we created a secondary program, postProcessing.m, which is responsible for presenting the information in a user-friendly fashion.

2.5 Outputting The Results[8]

This module comprises three different graphical interpretations of the output data mentioned in the previous module. I will offer this brief introduction on the general layout of the subsections to follow so that you, the reader, may be better prepared to interpret the information:

At the beginning of each sub-section you will find the representative graph of the most simple "song" we could imagine: a chromatic scale. For those not well-versed in music, a chromatic scale is one in which the instrument 'outputs' a series of notes, each note directly above or below its predecessor in frequency. Note: "scale," in this sense, implies either a constant increase or decrease of tone; therefore if one note is directly above its predecessor, the following note must be directly above this one note. Likewise for the alternate direction.

Following this graph will be a description. At the end of the description will be placed another graph or three. The distinction between the original chromatic scale and these secondary graphs is an important one: the individual (Michael Lawrence) who played the original samples also played the chromatic scale; the secondary graphs are interpretations of recordings done by professionals. Thus, not only do we find we have an unbiased test-set, we see how the samples sampled at 22050 Hz compare with a recording sampled at 44100 Hz. Our upsampling algorithm created to deal with just such a discrepancy is covered in the following module.

NOTE: The samples which generated these results are available in the following module.

[8] This content is available online at <http://cnx.org/content/m12467/1.4/>.

2.5.1 Most Likely Note Graph

Chromatic Signal Most Likely Note

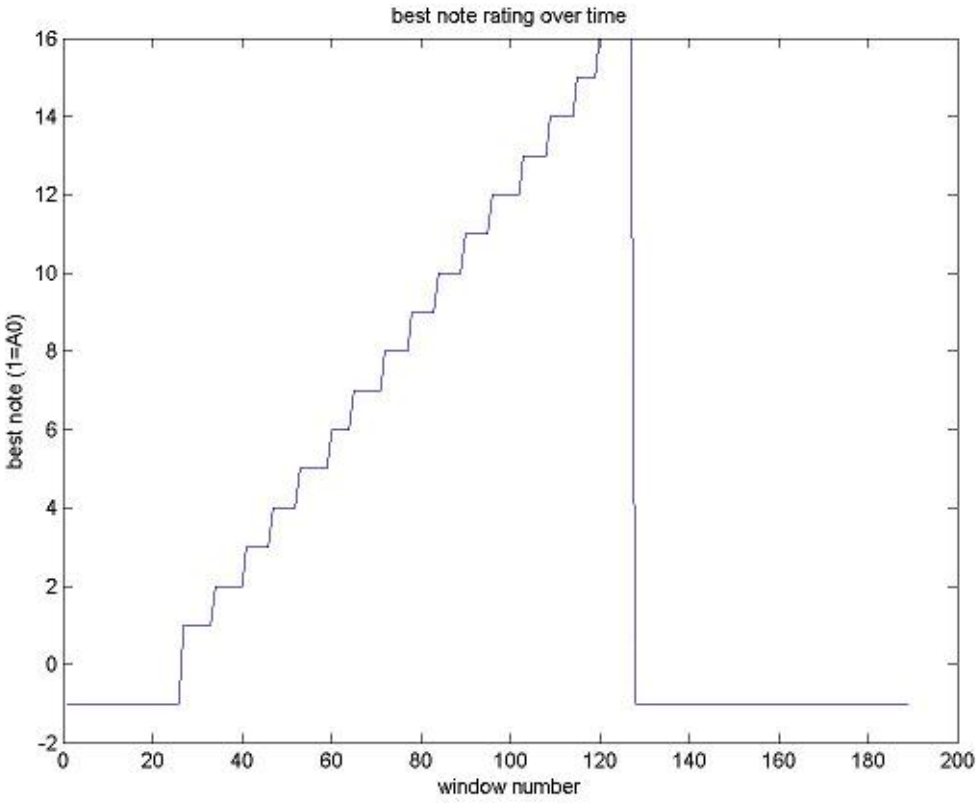


Figure 2.5: This graphically represents the most likely note played in each window for a signal in which a chromatic scale is played.

The above graphical output method is the result of the most straight-forward analysis of our data. Each window is assigned a single number which represents the note most likely to have been played within that window. This graph-type is the only one in which noise plays a considerable role; setting the threshold to zero results in "most likely notes" being chosen for each window in which there is only noise. Thus we have to **tell** the algorithm that only noise exists for those windows (i.e. it is silent). Our value for silence is -1. "1" corresponds to the lowest note on a Bb clarinet (an E in the chalameau register; in concert pitch, a D below middle C). Each incremental advance above that is one half-step (a **half-step** is the term used to describe two notes considered 'next' to one another in frequency).

The following graph is the output of our program when fed a professionally-recorded solo clarinet (playing the first 22.676 seconds (1,000,000 samples) of Stravinsky's **Three Pieces for Clarinet**). The chromatic waveform was created by the same individual who recorded the samples; thus the Stravinsky waveform represents an unbiased application of our algorithm against one instrument. This graph is meaningless for a song in which multiple notes occur; thus there is no output corresponding to a song in which multiple

instruments play.

Stravinsky Most Likely Note

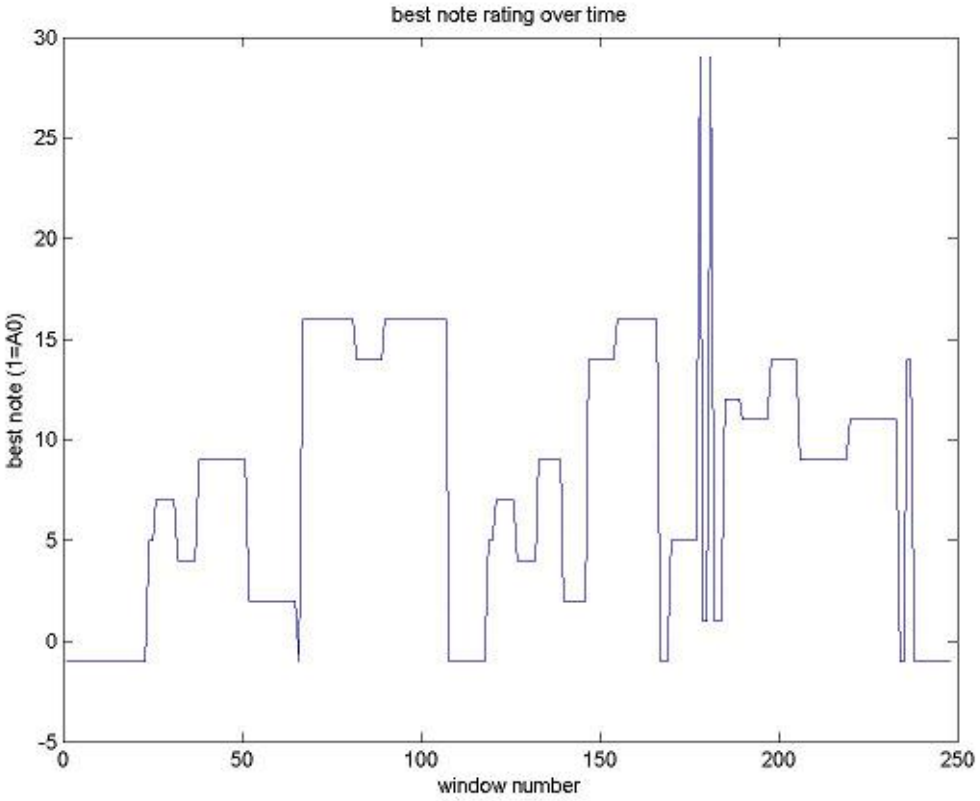


Figure 2.6: This graphically represents the most likely note played in each window for a professionally recorded signal (Stravinsky's **Three Pieces for Clarinet**). Note: This piece was chosen because it is a solo clarinet playing; no other instruments play. Also, note the one bad sample; the algorithm could not "pin down" its value. (Interestingly, the final peak is actually the performer's breath-intake; a testament to the value of thresholds). Lastly, the first note in the song is actually a grace note; our algorithm notices even this brief note as is made apparent in this graph.

2.5.2 Harmonic Likelihood Graph

Rating Acheived by Various Harmonics in the Chromatic Signal

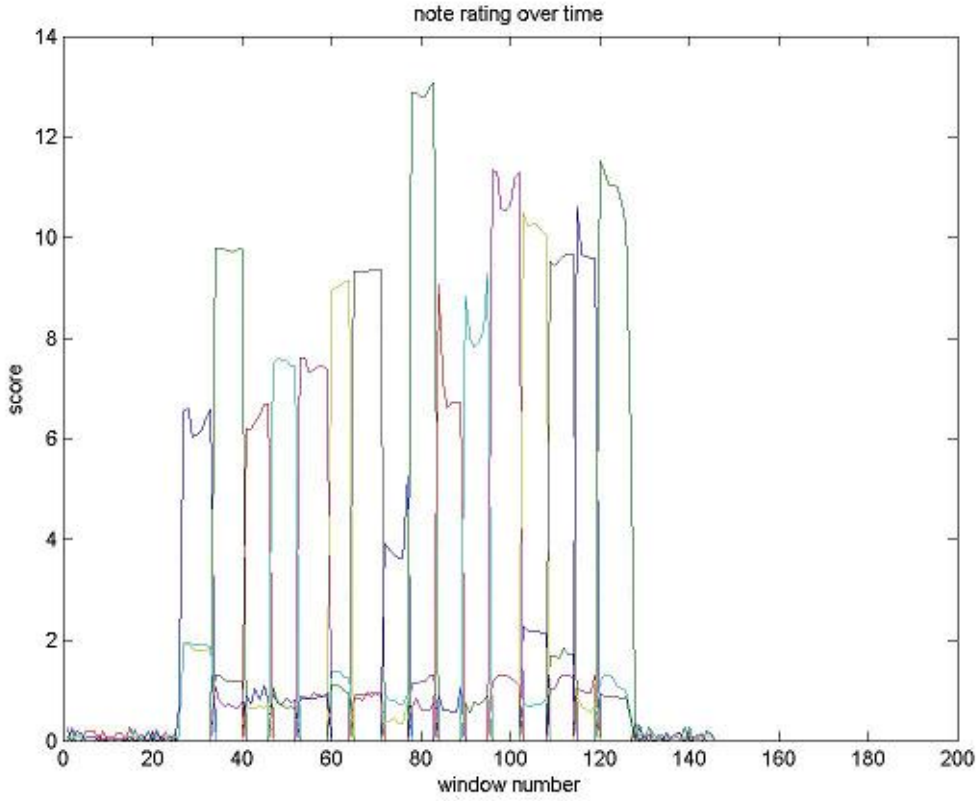


Figure 2.7: Each window has assigned to it several values indicating the strength of each harmonic within that window. For a window in which one note is played, this helps determine which is the note and which are merely the harmonics of that note. This grahpically depicts that rating for a chromatic scale performed on a Bb clarinet.

The above graphical output method is the result of a secondary, less straight-forward analysis of our data. To show the merit of tailoring the inputs for the matched filter, we graphically represented the rating assigned to each harmonic in a given window. Note how there is one over-arching, dominant waveform for nearly every window (except those in which there exists only noise) but see also the lesser, but still non-trivial, strengths of its harmonics. Without filtering for octaves of a signal note, our algorithm would more likely be tricked into thinking the harmonics of a note were the note itself (or perhaps other notes being played).

The noise is less of an issue when the data is perceived in this manner; thus no threshold value is required to determine that which is silence and that which is not. This method is still not useful in analyzing a song in which multiple notes are played during a single moment in time. For that, we turn to the third and last graphical method of representation.

Rating Acheived by Various Harmonics in the Stravinsky Signal

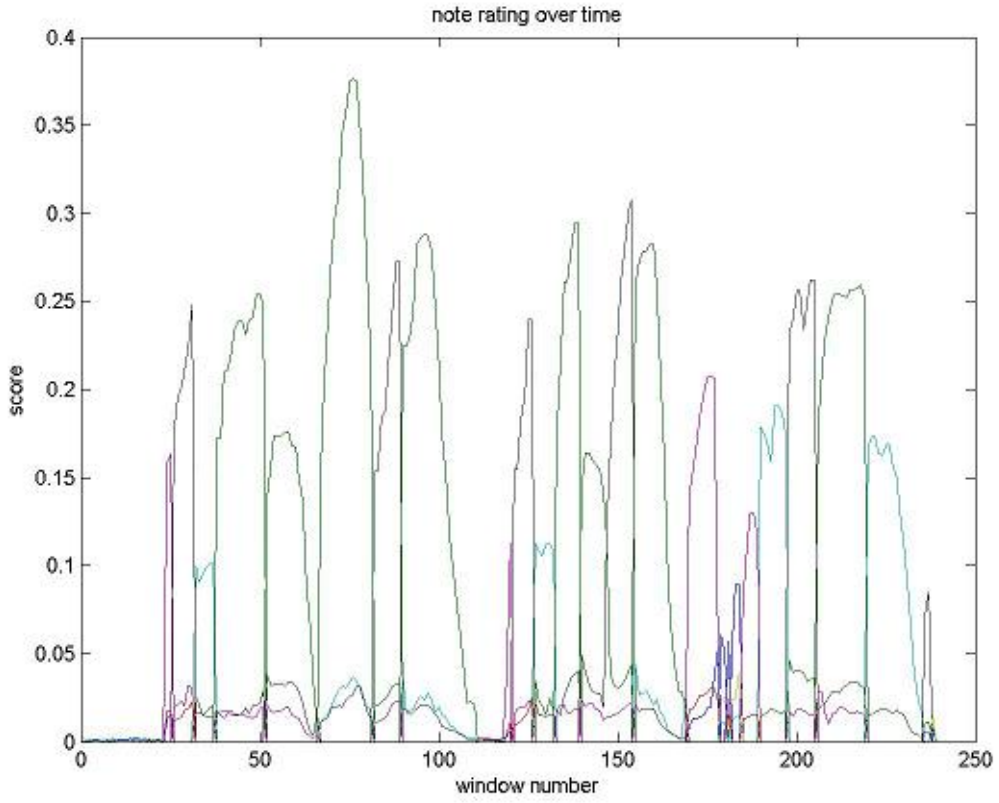


Figure 2.8: The monotony of color says nothing about the signal; unfortunately, Matlab assigns colors for waveforms on a rotation of 8. When graphing a series of functions in which it makes more sense to rotate by 12's, the coding becomes more difficult. Therefore, two signals having similar color says nothing about their relative values.

2.5.3 Musical Score Interpretation Graph

An Intuitive View of the Chromatic Scale

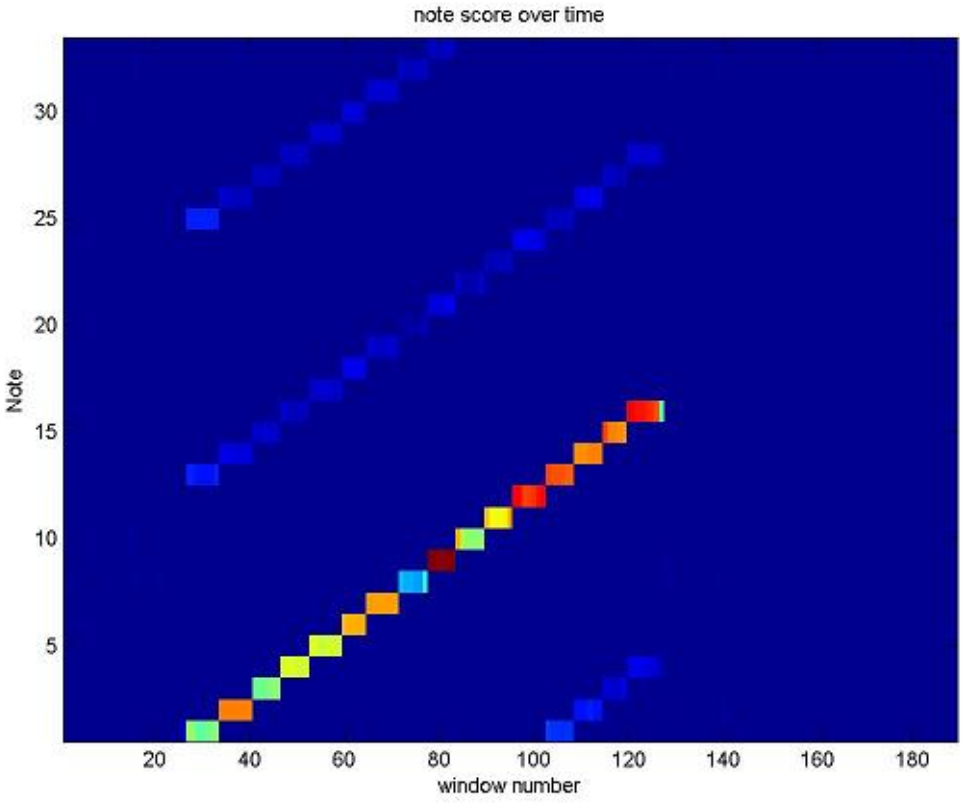


Figure 2.9: This graphical method is intended to represent the data in the most intuitive way (for a musician); the same way as seen on a musical score. Note that no harmonics are shown on this graph, only octaves of the given note. This is a result of the tailoring algorithm (using the Hanning window) preceding the matched filter which filters out excessive notes to decrease the time required for computation.

The above graphical output method is the result of our attempts to present an intuitive representation of our data. The goal is to produce an output which is most useful for someone completely unfamiliar with graphical methods yet intimately familiar with music. Thus, we graph each window as an image with colors assigned to the various values of the data inside the window. The result is something that looks surprisingly like a musical score. The more "intense" or colorful a particular region seems, the more likely it is to be a note played within that window. The chromatic scale serves to display the merits of this startling technique with distinction.

The first graph below this paragraph is the graphical interpretation of Stravinsky's **Three Pieces for Clarinet** using this method for processing our data described in this section. It serves the same purpose as the preceding Stravinsky graphs.

The most useful application of this graphical method, however, is that one may readily view several notes playing at once. This form is best embodied in the final two graphs. To first gather some sense of how to interpret the graph when multiple instruments are playing, the second graph below this paragraph shows a stripped-down version of the output from our program when it is fed the first 90.703 seconds (4,000,000 samples) of Barber's **Adagio for Strings** as played by a clarinet choir. A **choir** in its most general sense is merely the gathering of multiple like-familied instruments. That is, all sorts of vocal instrumentation (soprano, alto, tenor, bass) form the most standard interpretation of choir. Thus, a clarinet choir is one in which several members of the clarinet family (Eb, Bb, A, Alto/Eb, Bass, Contrabass, etc.) play in one ensemble. The final graph on this page displays the output for Barber's **Adagio For Strings** in all its glory.

The 'Musical Score' For The First 22 seconds of Stravinsky's "Three Pieces for Clarinet"

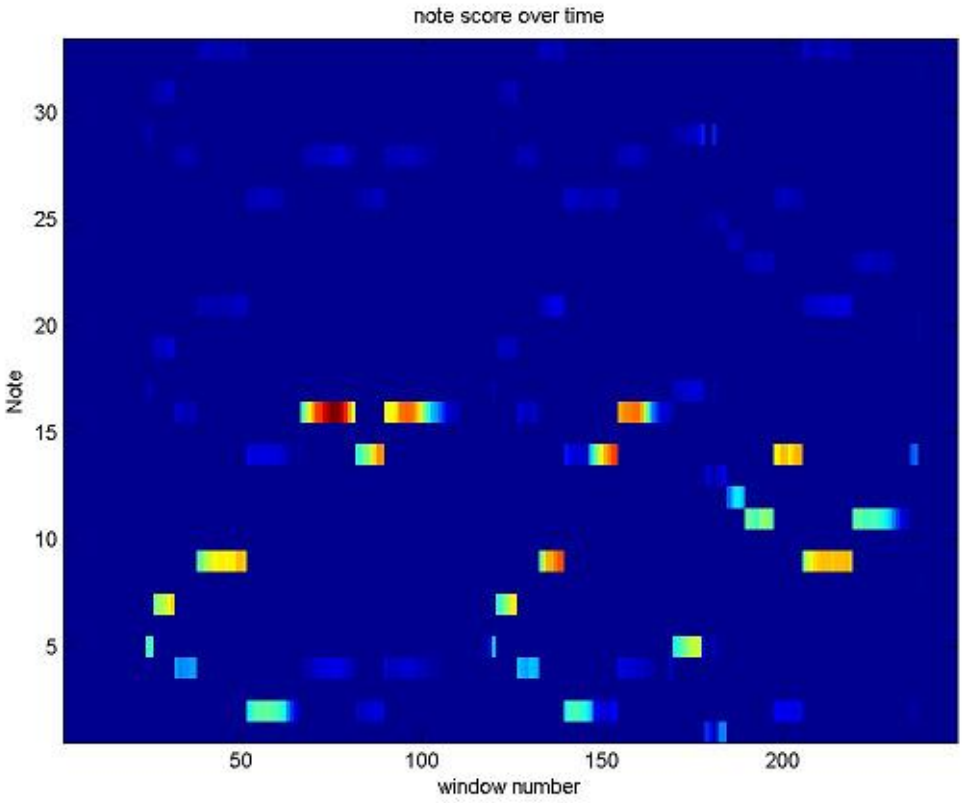


Figure 2.10: The musical score interpretation of the data output from an input of the first 22 seconds of Stravinsky's **Three Pieces for Clarinet**.

The 'Musical Score' for the First 90 Seconds of Barber's "Adagio for Strings" as Played by a Clarinet Choir

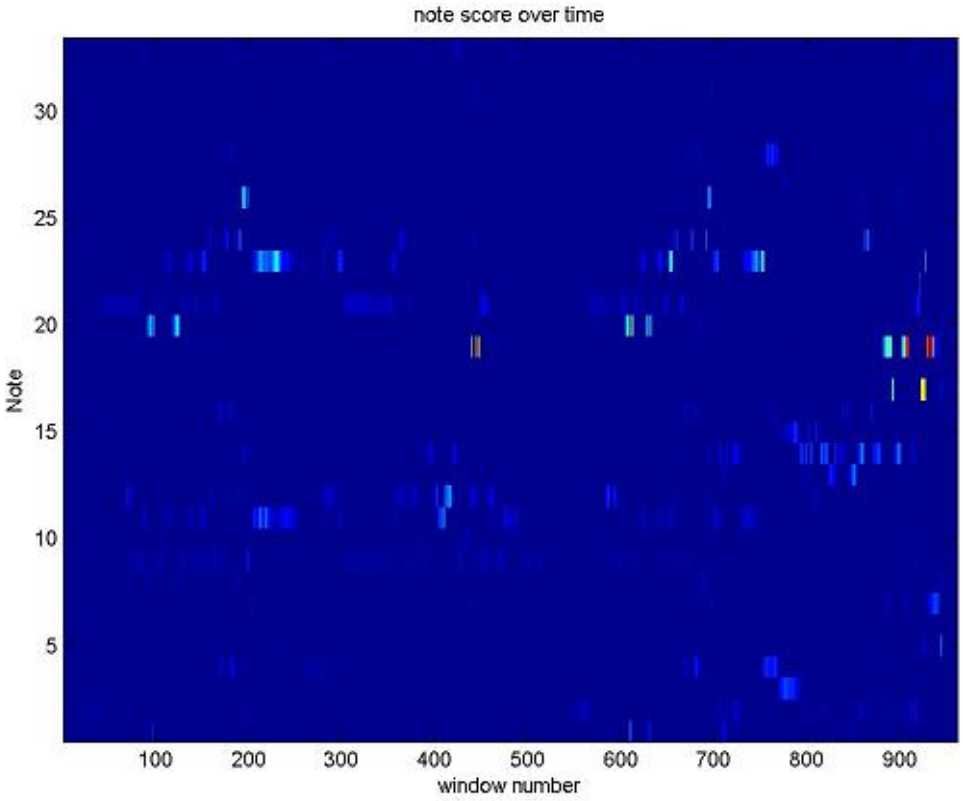


Figure 2.11: The musical score interpretation of the data output from an input of the first 90 seconds of Barber's **Adagio for Strings** as played by a clarinet choir. Note this is simplified for ease of interpretation.

The two graphs sandwiching this paragraph show the three voices of clarinet that play. The top line is the lead clarinet (there is only one). If one listens to the song, one may quickly note the 'v' in the middle appearing in the sound byte ("byte" used here in a general sense). The second line is the supporting clarinets and the lowest line is the bass line (or harmonics thereof). Our algorithm was told to search for three notes to produce these two graphs.

The 'Musical Score' for the First 90 Seconds of Barber's "Adagio for Strings" as Played by a Clarinet Choir

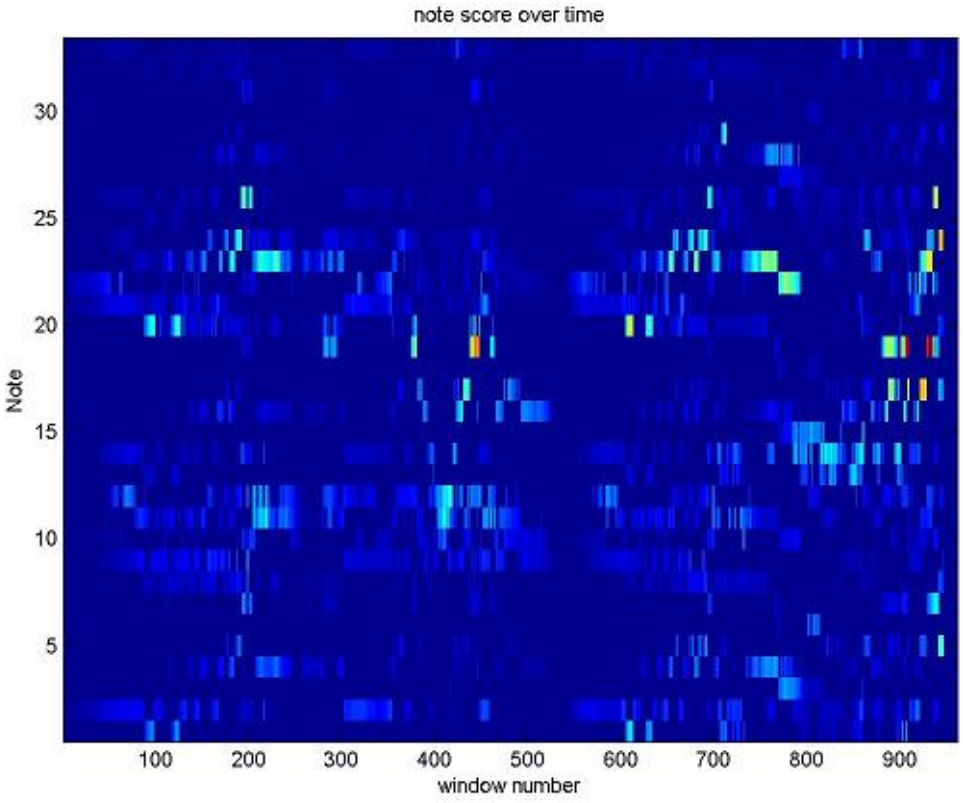


Figure 2.12: The musical score interpretation of the data output from an input of the first 90 seconds of Barber's **Adagio for Strings** as played by a clarinet choir.

Chapter 3

Completing The Task

3.1 Odds and Ends[1]

This module exists solely to tie up loose ends left over from the previous modules. More precisely, in this module may be found the Matlab code and workspace used for the purposes of this project and the sample waveforms against which we tested our program. First, though, a discussion on up-sampling:

3.1.1 Up-Sampling

Up-sampling (that is, representing something with few samples as something with many samples) is relatively straight forward when one deals with rational multiples. First, one converts the signal into the frequency domain using the Fast Fourier Transform discussed in a previous module. The samples are then "spread out" (zeros are added) based on the rational multiple by which one is up-sampling. Then a low-pass filter and IFFT later, you're back to an up-sampled version of the original signal.

3.1.2 Limitations

No project is complete without first recognizing the limitations inherent in whatever was accomplished. The most significant drawback in our program in terms of realizing our final goal is the lack of automating the threshold detection. Without an "intelligent" program, perhaps based on a neural network, we have little hope of filtering out a particular instrument in the data representing a full orchestra. We are quite capable, however, of detecting multiple instruments so long as the multiple is not too large and we are allowed to set the threshold ourselves.

The computational complexity increases with the number of instruments (samples) tested. We create no explicit infrastructure to break a song into component tracks (at least conceptually) so that we may analyze each one against a particular set of samples representing a single instrument. Also, we would be well-advised to input the frequency domain representation of the samples to decrease computational complexity.

A further limitation is the need to input several samples from the same instrument. Ideally, we would input merely the sound of that instrument playing and modulate that one sound to create as wide a range of tones as was required. The idea here being that a given instrument has a unique frequency fingerprint that remains intact over all frequencies. This is not perfectly true (each instrument has its own idiosyncracies relating to its real-world implementation), but might prove accurate enough for our proposed analysis.

3.1.3 Future Instrument and Note Recognition Endeavors

And no limitations section is complete without some mention of how to surpass those limitations. The goal of any project is to refine the product to the point beyond which refinement is no longer possible. Because

[1] This content is available online at <http://cnx.org/content/m12485/1.2/>.

this is in practice impossible to accomplish, we will list a set of future "next steps" we or others who follow may be encouraged to take.

To intelligently detect the relative volume of noise in a given sample, one might best be served to create a statistical filter which recognizes random noise. This statistical filter would, in theory, identify the windows which most resemble random noise. From knowledge of which windows cause noise, one might derive the volume-level (read: power-level) associated with said noise and set the threshold at some point beyond that. The upper-bound of the threshold could be found as the lowest power value for any other non-noise (as indicated by the statistical filter) window.

The threshold detection for specific instruments is more complicated: our suggestion is to develop some method of correlation or detection as-of-yet unknown to these authors (but likely known to those who research these concepts). This method would likely match frequency domain signals rather than time domain (that is, match filtering two frequency domain representations; sort of a meta-Matched Filter in terms of FFTs) using some statistical algorithm.

The computation complexity issue is trivial to solve. One must simply code the infrastructure to analyze a given signal in several channels, each acting as our entire program now acts. To convert the samples into the frequency domain, one need only FFT each sample.

The final observed limitation, too, is within our grasp. We briefly attempted a method which is promising: Mellin transformation. Essentially, when one takes a signal and transforms it into the Mellin domain (by multiplying by an exponential), one is in the position to merely phase-shift the frequency domain representation to acheive a modulation. Thus, converting back from the Mellin domain after phase-shifting the original transformed signal changes one note into another (musical modulation). This also has (**many**) more applications than simply for our particular program. Image recognition over dilation comes most immediately to mind.

3.1.4 Relevant Files

> NOTE: If you choose to use our files, we would like to be informed of their use. Not because we want to inhibit any potential use of our work but rather because we want to know our audience is more than a few trillion electrons searching the internet for googly content. Imitation is, after all, the sincerest form of flattery. We hope you find our work both enlightening and useful.

3.1.4.1 Matlab Code

Our primary program.[2]

Our output-processing program.[3]

Our Up-Sampling program. [4] (Expects a vector as input; outputs a vector).

Our Up-Sampling program.[5] (Expects a struct as created from Matlab's "Import Data" feature when importing a .wav file as input; outputs a similar struct).

3.1.4.2 Clarinet Samples

The samples used for analysis of the professional recordings (i.e. recordings sampled at 44100 Hz)[6]

The samples used for analysis of the unprofessional recordings (i.e. recordings sampled at 22050 Hz)[7]

One may convert these samples to any other sampling frequency by means of the up-sampling program. The samples cover from the lowest note on a Bb Clarinet (E in the chalameau register) to the highest C in the clarion register (right before reaching the altissimo register). The lowest three notes have questionable integrity (I choose to blame the microphone ;-)).

[2] http://cnx.org/content/m12485/latest/ProjectD.m
[3] http://cnx.org/content/m12485/latest/postProcessor.m
[4] http://cnx.org/content/m12485/latest/upSampleSample.m
[5] http://cnx.org/content/m12485/latest/upSampleSamples2.m
[6] http://cnx.org/content/m12485/latest/samples2.mat
[7] http://cnx.org/content/m12485/latest/samples.mat

3.1.4.3 Music Files (Signals)

A Chromatic Scale[8] , as performed on clarinet by the up-and-coming clarinetist, Michael Lawrence.

Stravinsky's **Three Pieces for Clarinet**[9] , unknown artist.

Barber's **Adagio for Strings**.[10] , Kalman Opperman Clarinet Choir.

For our program to work, .mp3 files must first be decompressed into .wav files. We used a free program found on http://www.cnet.download.com . We would post the decompressed files but, as one might imagine, they are too large to post on Connexions.

3.1.4.4 Poster

Our Poster.[11]

In the name of thoroughness, we include a copy of the poster created for an end-of-semester poster session show-casing our project. You should find a great deal of it familiar.

[8] http://cnx.org/content/m12485/latest/chromatic_slurred.wav
[9] http://cnx.org/content/m12485/latest/Stravinsky_Three_Pieces_for_Clarinet.mp3
[10] http://cnx.org/content/m12485/latest/Barber_Adagio_Clarinet_Choir.mp3
[11] http://cnx.org/content/m12485/latest/301_project.ppt

3.2 Auto-biographies[12]

3.2.1 Charles Tripp

Charles Tripp

Figure 3.1

[12] This content is available online at <http://cnx.org/content/m12486/1.1/>.

Charles Tripp is from Colorado, and is currently studying Electrical Engineering at Rice university.

I'm very interested in circuit design, computer engineering, and almost anything related to electronics or computers. I am an expert programmer and have written several widely distributed programs; I am comfortable with many programming languages, and am familiar with most computer systems. One of my major intrests right now is neural networks, and their application in various capacities.

My primary contribution to this project was the program. I am responsible for the majority of the projects code and the details of its implementation.

3.2.2 Michael Lawrence

Michael Lawrence

Figure 3.2

Hello world. My name is Michael Lawrence (as you might notice from the title of this section). I am currently an undergraduate Electrical Engineering major at Rice University.

My interests range from music to computers to sports to ... just about anything I find interesting, actually. I play clarinet and am competent with a trumpet, saxophone or piano under my fingers. I look forward to voice lessons next semester. I listen to Red Hot Chili Peppers, Guster, Radiohead, Linkin Park, and the Beatles substantially, but my favorite artist is probably Eminem. I also enjoy, on occasion, MC Hawking, Benedictine Monks, Pepe Romero (a classical guitarist) and Apocalyptica. A plethora of other artists might also make the list, but one must be concise.

Digital signal processing and physical electronics were my two favorite courses this semester in terms of content; therein may be found my electrical engineering interests. I am also in the process of building a robot capable of competing in the IEEE Region V competition; but it'll be the first thing to go when my workload gets too heavy next semester.

Michael Lawrence Playing Billiards

Figure 3.3

I play billiards. I won the Rice University intramural Billiards Singles competition last year and was on the winning team for College Intramural Billiards this year (singles and doubles competition have yet to take place). I enjoy a good game of ping pong. Basketball and tennis are two other athletic interests of mine.

My contribution to this project was primarily the coding/writing of the Connexions course you see before you. I provided a wall off of which Charles could bounce ideas for coding in Matlab but he generally dominated that section of our project.

3.2.3 Nate Shaw

Nate Shaw

Figure 3.4

Originally from Houston, I'm currently a Junior E.E. going into my second semester at Rice. When I'm not in the lab or doing a problem set you'll probably find me on the basketball court or listening to music in the 'privacy' of my spacious double. After Rice, I plan on getting an MBA...or becoming a professional gambler...whichever comes first :-)

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *Instrument and Note Identification*
Edited by: Michael Lawrence
URL: http://cnx.org/content/col10249/1.1/
License: http://creativecommons.org/licenses/by/1.0

Module: "Instrument and Note Identification"
By: Michael Lawrence
URL: http://cnx.org/content/m12462/1.1/
Page: 1
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Simple Music Theory as it relates to Signal Processing"
By: Michael Lawrence
URL: http://cnx.org/content/m12461/1.5/
Pages: 1-2
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Note Recognition"
By: Michael Lawrence
URL: http://cnx.org/content/m12465/1.3/
Pages: 4-5
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Deconstructing a Signal"
By: Michael Lawrence
URL: http://cnx.org/content/m12463/1.3/
Pages: 6-7
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Matched Filter"
By: Michael Lawrence
URL: http://cnx.org/content/m12464/1.3/
Pages: 8-10
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Matched Filter Output Analysis"
By: Michael Lawrence
URL: http://cnx.org/content/m12466/1.4/
Pages: 10-11
Copyright: Michael Lawrence
License: http://creativecommons.org/licenses/by/1.0

Module: "Outputting The Results"
By: Michael Lawrence
URL: http://cnx.org/content/m12467/1.4/
Pages: 11-19

Module: "Odds and Ends"
By: Michael Lawrence
URL: http://cnx.org/content/m12485/1.2/
Pages: 21-23

Module: "Auto-biographies"
By: Michael Lawrence
URL: http://cnx.org/content/m12486/1.4/
Pages: 24-27

Instrument and Note Identification
This contains the modules in which the Fall 2004 Elec 301 project of Charles Tripp, Michael Lawrence and Nate Shaw is explained.