Cardiff School of Computer Science & Informatics

One semester individual project
Final report
40 Credits

Machine translation of guitar audio

Author:
James Cohen

Supervisor:
Professor. A D Marshall

May 6, 2014

Abstract

Extracting relevant semantic features from guitar audio has long been a goal of musicians and digital signal processing (DSP) researchers. This research focuses on classifying the type of guitar, plucking intensity and extraction fret and string positions for single note guitar samples. Using the sum of the total content for an Inharmonic Comb Filter (ICF) feature vector of a single note sample an 87% accuracy can be achieved for predicting the exact string and fret position for the given note, this accuracy is even higher on the lower frequency strings of the guitar. Use of machine learning and classification has been successful for determining if a sample is an acoustic or electric guitar with an 80% accuracy. Machine learning has been less successful for both the plucked intensities and a multi-class guitar type classification which achieved an accuracy of 38% and 39% respectively.

Acknowledgements

Many thanks to Professor A D Marshall for supervising this project and providing wonderful advice in both the computational and musical domains of this project.

Contents

Introduction

The aim of this project is to develop computational tools that can discern different features of a piece of guitar audio and provide further semantic content. An understanding of the structure of guitar audio data has been achieved by studying and comparing the spectral content of guitar samples. The knowledge gained from this analysis is used in the extraction of information that may be useful to musicians or listeners from labeled guitar audio.

The primary assumption that is made during this research is that all of the features and differences between guitar audio content that are possible to be classified are audible, therefore it should be possible to find intelligent ways to measure these differences using the spectral and temporal domains (although the methods mentioned in this report focus solely on the spectral methods).

The project can be broken down into these steps:

- Appropriate guitar audio data was gathered and sorted.
- Programs were then written to visualise the data for different problems to attempt to see if differences in spectrograms could be seen.
- Programs to convert and filter data in to the accepted classifier format were written.
- The results of the classifier were analysed and appropriate adjustments were made where necessary.

Outcomes

The main outcomes of this project were:

- A tool that can predict the exact string and fret position of a given sample for a given guitar.
- An SVM classifier for guitar type.
- An SVM classifier for plucking intensity.
- Development of a reusable Matlab toolset that could be used in the future in similar projects.

Background

Initial research

Machine learning

A reasonable understanding of machine learning techniques has been essential for trying to make the most informed decisions possible about how to set up and design the classifiers for this project. As I came into this project with no real practical knowledge of machine learning I decided that the best course of action would be to read a textbook or to follow a course as such Stanford University's open course on Machine Learning taught by Andrew Ng [1]. This provided and excellent and approachable introduction to a subject I knew little about. One big draw of this course was that it focused on LibSVM [2] which is used in this project. This meant I had excellent tutorials to test that LibSVM worked as expected before actually diving into classification of my own. A knowledge of machine learning has been important in understanding different types of classification that could be used and what they identifiers meant. This allowed me to make informed decisions about what I was using and also allowed better understanding of problems that might have occurred such as over fitting and under fitting. Most importantly though, this series explains the basic concepts of different types of clustering, such as linear regression and it also explains concepts such as confusion matrices.

Extracting feature vectors

A paper that has been crucial in the development of the project is "Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano" by Galembo and Askenfelt. This paper talks about a method of creating a measure of inharmonicity which is *"A convenient representation of how much a spectrum deviates from the strictly harmonic case"* [3]. and extracting feature vectors from individual piano note audio files. This is pivotal to the research I have been involved in as using a simple histogram method produces a large amount of bins and the change in notes causes a shift in the fundamental frequency and the harmonics, meaning that vectors cannot be directly compared to each other due to the fact that the energy appears in a variety of different bins. The inharmonic comb filter method is essentially a normalisation technique but to normalise based on the frequency. An extension to this that would be preferable would also be to normalise the bin spacing. At the moment the the inharmonic comb filter bins are linearly spaced, this however is problematic due to the fact that the musical scale is exponential in nature, that is to say, the frequency of the octave of a note is double that of the octave below (for instance the frequency of E2 is 82.41 Hz and the frequency of E3 which is an octave above is 164.8 Hz). Therefore normalisation based purely on the fundamental frequency may prove not to be accurate enough. This means using a different method of mapping of the harmonic components may be necessary. Brown outlines a filter [1] used for re-spacing bins which is fundamentally an exponential to linear transform. This method could be used

[1] http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning

in conjunction with the note identification that the inharmonic comb filter can produce to provide the bin spacing.

Chord detection

There are various approaches to chord detection and the research into this topic is fairly extensive, possibly due to the fact that there are a wide range of applications of the technology that can be monetised. Chord prediction seems to be moderately accurate if the chord is played using it's most conventional shape, however it is my understanding the prediction is not wholly accurate. This means that the software is probably locating the largest harmonic components of the Fast Fourier Transform (FFT) of the signal and then matching these to a chord database that contains the principle components of the chord's signature. A technique similar to this is mentioned by Lee [6]. As this paper states there have been various approaches such as the template matching technique. Template methods attempt to produce a harmonic pitch class profile (HPCP) (also known as a chroma vector) such as [1,0,0,0,1,0,0,1,0,0,0,0] using the labeling [C,C#,D,D#,E,F,F#,G,G#,A,A#,B]. This is problematic, as the paper states due to the fact that real instruments produce varying amounts of overtones, which produces a noisy signal making it hard to match to the chroma vector. The paper improves on this method by introducing a concept called enhanced pitch class profiles (EPCP). This method improves on HPCP for chord recognition by suppressing non chord tones and helps to remove ambiguity.

The research that has been completed during this project may help in trying to better distinguish different chord positions on the guitar due to the fact that exact fret positions can be obtained. By compounding individual samples to create chords and comparing them to recorded chords for the same guitar could be used as a metric for measuring how accurate these predictions are. This may be possible due to the fact that the prediction for the total normalised energy in the overtones is available by approximation (specifically by predicting the position based on energy of the note in the line

The specific deficiencies that the projects aims to address are mainly centered around increasing the amount of knowledge that can be gained from a guitar audio file simply by breaking looking at specific features in a more constrained set of samples.

Alternative approaches

The Music Information Retrieval Evaluation eXchange (MIREX) conference [2] and it's papers inspired some of the early ideas about how the audio data could be processed, whilst none of these methods were followed the papers provided an interesting perspective about what direction could be taken.

[2] `http://www.music-ir.org/mirex/wiki/MIREX_HOME`

Existing products and solutions

There are many programs that attempt to accurately transcribe guitar music. There does not seem to be any tool that can distinguish between guitar types, plucked intensity or position on the neck. An interesting summary [3] is given in the blog *Six string recess* and seems to include some of the main software products available. Capo 3 seems to be one of the more heavily recommended software solutions, however the reviews on the internet do not seem wholly positive [4] [5] [6]. The computer music market was worth approximately $350 million in the US in 2009 [7]. This is an impressive market size and one which could be capitalised on with software such as that examined in this project.

Stakeholders

There are variety of parties that interested in this type of research, the developers of the technologies come from both research and commercial backgrounds. The users of such technologies would primarily include musicians and people who listen to music (especially those who use music recommended by software).

Applications of this technology

There are wide and far reaching applications for using this technology within the multimedia computing sphere, these include:

A transcription tool for professional or amateur musicians

Musical transcription involves discerning information, primarily notes, their timings and dynamics from a piece of recorded music. It may also include features such as slides, bends, vibrato, muting and even strumming directions. It is often very crucial part of writing music for some musicians. Transcription is a laborious process for musicians and instrumentalists and normally requires a large amount of experience (although this will vary depending on the complexity of the piece of music). Precise guitar transcription is especially tricky due to the fact that a note may be played on several positions on the guitar (the number of positions varies depending on the exact note). Whilst this increases flexibility for the player, it adds an extra layer of complexity in the transcription phase that would not be an issue for a chromatic instrument such as a piano. The tools developed could perhaps be used as an extension to tools that already exist to help guitarists transcribe their music more quickly and efficiently.

[3] http://www.sixstringrecess.com/gear/guitar-software-shootout

[4] https://itunes.apple.com/gb/app/capo-3/id696977615?mt=12

[5] http://www.macworld.com/article/2052192/hands-on-with-capo-3-for-mac-easily-learn-your-favorite-songs.html

[6] http://bassmusicianmagazine.com/2014/03/learn-music-with-capo-3-a-review-by-john-kuhlman/

[7] http://www.nxtbook.com/nxtbooks/namm/2010globalreport/index.php?startid=28\#/30

Semantic labeling of content

Partially linked to transcription tools, the techniques outlined in this report could be used in music recommendation systems to help distinguish between different types of content.

Tuition

Private tuition may be considered prohibitively expensive for some students at a cost of mean cost of 24 [8] per hour it is certainly not affordable for everyone. The techniques developed in this project could be used to enhance current transcription tools in a way that makes them more usable for tuition system (more specifically the exact string and fret prediction tool).

Gaming

The system used in the video games *Guitar Hero* or *Rockband* matches a users guitar playing (using a to guitar) to that of a popular pre-recorded track and gives them a score based on their accuracy. A *Rockband* compatible guitar does exist [9] however this is a proprietary piece of hardware. A potential use for the system could be to use it to 'gameify' teaching of the guitar. Much like language tuition platforms such as *Duolingo* [10] and *Memrise* [11], that offer users points and rewards for competing levels of language (for instance this either comes in the form of vocabulary or grammar). This system has potential to be used to work out how accurately a user has played an already transcribed piece by comparing their transcribed playing with the systems representation.

Problems

The biggest problem with this research is obtaining good quality (high resolution with a high bit rate) and well annotated samples. This extremely important due to the fact that the audio needs to be as close to the original source. Having high quality audio Polyphony is also a problem in this domain and separating polyphonic audio is not, there are techniques such as Independent Components Analysis that could be used to separate the signal [9] [12].

[8] https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/222187/DCSF-RR081.pdf

[9] http://www.rockband.com/blog/fender-squier-pictures

[10] https://www.duolingo.com/

[11] http://www.memrise.com/

[12] https://www.cs.tau.ac.il/~nin/Courses/NC05/ICA.ppt

Design

The main project can be broken down into three experiments, although these experiments contains sub experiments that helped explore the space and validate or disprove the hypotheses.

Validating models

Validation is fundamental for proving a model or classifier's accuracy. If the accuracy has not been validated it cannot be said if the model is useful or not, or whether indeed it works properly.

Cross validation is a group of methods for statistical model validation that attempts to measure if a models predictions are accurate. It does this by using part of the data set that could have been used for training.

The cross validation method chosen is repeated random sub-sampling. The reason for this is that it can give a large range of tests of any specifiable amount of the data set (although obviously at least a large proportion must be left aside for training data).

Repeated random sub-sampling works by removing a fixed percentage of the data (normally set at 20%) from the training set and using this as a testing set to validate the model using data of a verifiable level of accuracy. An advantage of this is that the data can be verified easily and simply whenever building a classifier and the data is from the same set so the data has a consistent labeling.

This methods lends itself to the sort of data sets such as the ones used within the classifiers of this project as a moderate amount of data can be stored separately for testing when the training data is compiled.

Repeated random sub-sampling unfortunately does come with trade-offs. The principle being that as the testing set is chosen randomly the classification rate returned will vary with each run, this is because repeated random sub-sampling exhibits Monte Carlo variation. The samples for both the training and testing data vary each time the model is created because these samples are selected pseudo-randomly from the complete dataset.

Three hypotheses were tested these were:

Multiple guitar classification

Different guitars produce different tonal characteristics therefore there may be a way to distinguish the type of guitar based on it's audible characteristics. An initial test was to put various single guitar samples into a multi class classifier using both a histogram method and the ICF method. These approaches were both compared to see if one produced a higher accuracy than the other.

Plucked intensity classification

An assumption that was to be tested was that the harder a string is plucked, the more energy that will be displayed in the subsequent overtones for that string. To test this a comparison between plucked intensity has been produced using periodograms to visualise the spectral density in Matlab. This was an initial verification as to whether it would be worth continuing to build a classifier.

Exact fret position

Another theory was that the the total energy of the harmonics would decrease for a normalised feature vectors when the fret position was increased. This would in theory mean that a note played in a lower or higher position on a different string would exhibit different characteristics in the harmonics. To test this requires using a data set that is marked appropriately, as such I only possess one data set currently that contains the right sort of data (string and fret positions for open and fretted strings up to fret twelve for every string on the guitar). The reason that this data is hard to obtain is that sample libraries on have one played position per note on the western scale as opposed to the multiple positions that are possible for the guitar. This means that vast swathes of the guitar's fretboard would have to be ignored in this analysis. Ideally this test would also be completed over a similar range of played notes for various plucked intensities to further establish the effect of plucking intensity and fret position.

The validation for this model is slightly different to the SVM classifiers due to the fact that it does not use machine learning principles to predict a value, but is based on a model, therefore, every sample used is actually a test so the whole data set is used for.

Tools

As changing languages or tool-sets partway through a project would often mean a complete re-write of all of the code up to that point it is crucial that tools are selected carefully.

Matlab

Matlab was chosen as the programming platform to be used for the all of the processing and as an interface for the machine learning in the project. There were a multiple reasons for this:

- Matlab provides a wide array of DSP functions that would often require importing from an external library in many other languages (or even writing from scratch). Having such a wide range of DSP functions available sped up the programming phases immeasurably. The FFT was of particular importance as all of the analysis dealt with audio in the spectral domain.
- Matlab's native plotting functions were especially advantageous whilst prototyping as they allowed quick and accurate visualisations of the data.

- It is a language I know well and as such was able to work quickly and efficiently in it.

LibSVM

It was clear from the beginning that a supervised learning approach would be taken for classification due to the fact that guitar audio data was widely available in both commercial sample packages and by home recording. LibSVM's range of models that can be chosen is also a boon. Support vector machines (SVMs) provide an obvious choice as they can facilitate multiple classes which is needed for this project. LibSVM [2] was chosen as the classifier for this project. An SVM works by separating each element of a feature vector out into a higher dimensional space and computing the clustering for each element of each class for the feature vector. Multiple clustering methods can be chosen, such as linear regression, radial and sigmoid.

Whilst there are many excellent classifiers available that use different models (such as Matlab's Neural Network Toolbox [13] and Matlab's SVM in its Machine Learning Toolbox [14]), LibSVM has a very easy to use Matlab interface that made combining it with a Matlab code base possible, removing the need to interact directly with the system from Matlab. LibSVM is also portable between languages and operating systems. This would make a potential move to a different language possible and is useful in that the tools to build a classifier can still be built and run in Matlab, but the classification itself could be run from a completely different interface such as Java.

Git

Git was used to track changes in files and was even used to track changes in the report. Git's use however was possibly slightly unneeded due to the fact that the project comprised of lots of small scripts and functions which were rarely changed (except for a couple of the larger scripts). One very clear advantage was that this made backing up the project a very simple process, due to the existence of websites such as Bitbucket and Github.

[13] http://www.mathworks.co.uk/products/neural-network/
[14] http://www.mathworks.co.uk/machine-learning/

Method

Rather than focusing on a complete transcription system, this project prioritised classifying various subsets important to transcription and automatic addition of semantic data with regards to guitar audio. The reason for this is that building a new transcription system from scratch would have been unlikely to produce any new insights to the topic, primarily because of the complexity of implementing a system such as this in the time constraints. Hence all of the research outlined from here on in focuses on gaining useful features of guitar audio that might enhance and improve current transcription methods.

Guitar audio sample data

Having large amounts of high quality samples is crucial for attempting machine learning and classification. As one set of samples may exhibit anomalous characteristics, the variance between sets also needs to be deduced. The methods being developed must be as flexible as possible and work with as many guitar types so having a range of samples from both electric and acoustic guitars is necessary.

The data was collected from two main sources, the internet [15] and by recording guitar samples myself. Recording allowed the greatest level of control over data was obtained, however it is an extremely time consuming process recording, trimming and labeling the audio files, as such this process was only completed once. Therefore pre-recorded samples were used for the bulk of the analysis.

Whilst around fourteen sets of samples were acquired, not every set could be used for every classifier and as such the abilities of each classifier are greatly limited. This problem became more apparent towards the end of the project where it proved most problematic (specifically when developing the classifier for string and fret position) as sometimes there was not enough data to test hypotheses rigorously as I would have liked.

Data preparation

Input files

All of the audio files used in this project used different naming conventions which proved to be a problem when building the classifiers. The reason that this proved to be so problematic was that all of the labeling that was used in the classification was stored in the filenames. To fix this I tried to start moving the directories towards a more standard naming convention.

Sanitising (or more specifically removing extraneous and unneeded files) and re-structuring the file directories was an iterative process in which file directories were modified when needed. The first stage of this was to separate out all of the files that could actually be read by Matlab. This was completed manually and only *.wav* guitar samples were used. These

[15] http://bedroomproducersblog.com/2011/12/03/free-sample-shootout-7-best-free-guitar-and-bass-samples/

samples were kept in their respective folders to preserve their links to each other (and to ensure that there were no clashes with different guitars that used the same naming convention). These folders were isolated into a folder of samples and such there was a clearer structure.

This approach was taken as it was decided from the beginning of the project that the multiple weeks it would have taken to design and implement a database were not warranted and would have slowed the research process down to a point where it would have not been possible to achieve anywhere near as much practically.

The Ruby programming language was used as it comes installed on Mac OSX (the platform I choose to program on) and it is very easy to run small scripts that change file names and positions. The main techniques used were regular expressions shown in Listing 1 and string substitution shown in Listing 2. These scripts were written to modify file names and move files to different directories. Using Ruby was also a good excuse to teach myself some of the basics of another programming language.

There were various instances where these scripts were useful:

- In removing certain characters such as brackets, which caused problems in Matlab's file reading.
- Substituting in different variations of characters increase homogeneity between different classes of guitar.
- For moving large batches of files to another directory when they were not appropriate for a classifier.

Listing 1: File modification in Ruby using regular expressions

```
filenames = Dir.glob("*.wav")

filenames.each do |filename|
     if /_GN_|_GU_/.match(filename)
        File.rename filename, "muted/" + filename
     end
end
```

Listing 2: File modification in Ruby using string substitution

```
filenames = Dir.glob("*.wav")

filenames.each do |filename|
    File.rename(filename, filename.sub( "_1.wav", "_L1.wav" ).sub( ...
       "_2.wav", "_L2.wav" ).sub( "_3.wav", "_L3.wav" ))
end
```

Histogram

A histogram was initially chosen as the method for reducing the size of the input data down to a size that it is more usable. It takes the arguments; the path of an audio file, the number of bins desired and a threshold. This functions in a very simple manner, by reading the given audio file, computing it's FFT, calculating the bin size (which is simply the floor of the FFT length divide by the number of bins) and then calculate the mean value for each bin. This means that some data can be lost from the end of the FFT due to having to floor the number. However, that end data is far beyond the useful frequencies for this analysis (the maximum fundamental frequency of a 24 fret guitar is 1319 Hz and the maximum frequency that humans can hear varies but for most is around 18-20 Khz depending on age [7]). The histogram whilst useful for comparisons of the same note in initial analyses such as plucking intensity isn't particularly useful for any machine learning or classifying due to the fact that the frequencies are not normalised.

Inharmonic comb filter feature extraction

A large portion of time towards the end of the project was spent refining the method originally set out by Galembo and Askenfelt [3] and was initially used to predict the frequency of bass piano strings. The reason for this is that the ICF method can provide an easy way to extract only the harmonics from an FFT and gets rid of a lot of extraneous data, reducing an array that can be hundreds of thousands of elements long to a length of thirty or less. The ICF Matlab program that has been modified for this project was originally produced by Pritchard in his Masters Thesis [8]. Several modifications have been made to facilitate feature extraction from guitars. The features that are extracted from an an audio sample using the ICF method are:

- A feature vector containing up to the first 30 harmonics.
- The fundamental frequency prediction.
- Inharmonicity estimation.

The basic way that the ICF works is by reading an audio file, computing its FFT, looping over the FFT and computing bands where the energies of the signal is stored. This loop is repeated three times to obtain an accurate fundamental frequency estimate, feature vector and inharmonicity estimate.

A crucial input to the ICF is the estimate inharmonicity (often referred to as BEst in the program). This is shown in Figure 1 originally from the paper 'Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano' [3]. An average value for the inharmonicity is used, the results to show why this is appropriate are outlined in greater depth in Table 1 (see results section), but to summarise those results, an average inharmonicity works equally as well in string and fret prediction as one that has been calculated for each individual note.

$$B = \frac{\pi^3 D^4 E}{64 T L^2}$$

Figure 1: Inharmonicity equation

An important modification was to allow a larger range of frequencies as an input. In the original program a range of 20 to 200 Hz was accepted as the range for the fundamental frequency. The maximum value was then increased to 1500 Hz to accommodate the greater range of notes that were necessary to classify.

One of the main modifications to the program provided by Pritchard is shown in Listing 3. This snippet of Matlab code decreases the bin size (called M in the program) of the ICF. This was necessary as the code that this was adapted from would access values from the FFT that were out of bounds for higher frequency guitar samples (this seemed to vary arbitrarily depending on the sample used). This ensures that when M reaches it's maximum value it will not try to access a value that is outside of the maximum length of the FFT array of the input guitar signal. *Point* is initialised to a large arbitrary value. The maximum theoretical value for M is decremented until it reaches a value that is as large as it can be without *point* causing a error when the FFT is accessed later in the program.

Listing 3: Bin size decrementing code

```
% Maximum value for n is 200
% sampRate is the sample rate (normally 44100Hz for the samples I have ...
    used)
f0 = exp( Testing( n ) );

M  = 30;
point = 10000000000;
FreqSum = 0;
Offset  = f0 * OffsetPercentage;

while point ≥ length(powerSpect)

    x = 1 + ( ( M^2 ) * bEst );
    fK = M * f0 * sqrt( x );

    J = floor( fK + Offset );
    point = FFT_length * ( ( J ) / sampRate );

    M = M - 1;
end
```

Another important change was to fix another out of bounds error for the linear interpolation that happens right at the end of the program. Initially these interpolations were to plus or minus two values either side of the final estimate value. These interpolations did not seem to be a problem with the original piano samples, however, it did prove a problem

for some of the guitar samples that were tested during this project due to the fact that the relevant feature vectors often sat at the edges of the matrix. Interpolation was not always possible and provided more out of bounds errors. The simplest fix was to modify the values so that no interpolation happened for the edge cases. Listing 4 shows how this was achieved.

Listing 4: Linear interpolation modification

```
    interpMinus2 = 0;
    interpMinus1 = 0;

    interpPlus1 = 0;
    interpPlus2 = 0;

    if MaxN ≥ 3
        interpMinus2 = 2;
        interpMinus1 = 1;
    end

    if MaxN ≤ 28
        interpPlus1 = 1;
        interpPlus2 = 2;
    end
```

Normalising feature vectors

Normalising feature vectors was crucial to get accurate results due to the fact that it was impossible to verify if the samples recorded had similar amplitudes. The simplest way to normalise the feature vectors was to find the maximum power value of an array and divide every value in the array by the maximum. This means for any given vector, the fundamental frequency is always the same (in all cases during this project the maximum value of one was chosen as the value to normalise to). The program for normalising an array is shown in Listing 5. Having this ability to normalise vectors cannot be understated and lead to easy and fair comparisons later in the project.

Listing 5: Normalise a feature vector to a given value

```
function output = normalise(inputData,scalingValue)

    maxValue = max(inputData);
    output   = inputData./maxValue*scalingValue;

end
```

Initial spectral analysis

The success of this project hinged on one very crucial aspect, that out of the many samples obtained, there were discernible differences that could be tested and then classified. We can hear if there there is an inherent difference between two audio samples by listening to both and comparing them. For instance it is possible to infer the intensity at which a guitar has been plucked by simply comparing the harmonic content of two similar samples. It may even possible to even further deduce what has been used to pluck the guitar eg. A finger, a thin nylon plectrum or a thick plastic plectrum with further research.

There were various assumptions that could be measured both audibly and then by measuring the difference on a periodogram (a plot of frequency against energy) with spectral analysis. The assumptions that could be tested for:

- The note attack (or how hard the string was plucked or strummed) and the effect in a note's harmonics with a signal with normalised amplitude.
- Exact position of a note on the fretboard.
- The type of stringed instrument, such as acoustic or electric guitar, or even other types such as violin or bass guitar.
- Other played features such as bends, slides, harmonics and palm muting.

The theory behind all of this analysis is that as there is an audible difference between two samples this must be measurable by analysing the signal.

An assumption that was made was that when a string is plucked vigorously a 'ringing' artifact is produced. The assumption made was this artefact would be visible in the harmonics of the signal.

The initial spectral analysis for the plucking intensity classifier consisted of listening to labeled audio files and determining which sounded as if they had been plucked harder and then comparing the signals. In this case a specific label that indicated how hard the string had been plucked already existed. The label was either *mf* (mezzo forte), *f* (forte) or *ff* (fortissimo). To compute the difference between the signals a program was written to read both audio files, calculate their histograms and normalise these histograms to a maximum value of one and then took the difference of the two vectors. This produced a relative difference between the two histograms and from this it was decided to continue with the plucking intensity classifier.

Machine learning and classification techniques

There are various methods for classifying sets of data, however as mentioned earlier in the report, LibSVM [2] has been chosen for use in this project. By producing different classifiers and running them separately it may be possible to build a more complete picture about the guitar signals to be analysed.

Two classifiers were built:

- To classify the different types of guitar which this may potentially be useful in the labeling of audio to provide richer semantic content for users. This could be interesting to someone that would like to know the type of guitar used on a particular recording where there is no more information other that e.g. if a user is listening to a song and wonders what type of guitar the musician used to record that song.

- To classify plucking intensity, this would be useful as it could add information about the articulation of notes to an automatic transcription of a piece of music.

There are two methods that have been used to generate feature vectors for the classifier, shown in Listings 6 and 7. These feature vectors were then added as a new row to an empty matrix (that had zeros pre-allocated to increase run time speed). The labels for the feature vector were written into a column vector. The classifier label for the guitar type was an integer value that got incremented every time a new directory was accessed. This meant that each folder of files was assigned to a new class in the classifier. The classifier label for the plucking intensity classifier was taken from the file name each file used in this classification was labeled L1, L2, or L3 (L1 being a light pluck and L3 being a heavy pluck). Regular expressions were used to filter out any files that did not conform to this label in the data sets that were used. The classifier label was then taken from the file label by using regular expressions to separate the number from the end of the match.

Either the histogram or the ICF can be calculated for a given audio sample. For both the histogram and the ICF the power is normalised, which is particularly important due to the fact that the audio files range greatly in amplitude.

The feature vectors must then be converted into sparse notation for libsvmwrite to accept it. Libsvmwrite writes feature vectors to a format that can be read by LibSVM. This is done by using the Matlab function sparse on the matrix of the feature vectors. Listing 8 shows how the files for LibSVM are created.

Listing 6: Writing histogram vectors

```
processedSample = normalise(flattenMirror(sample, histogramBinNumber, ...
    histogramThreshold)),1);
```

Listing 7: Writing inharmonic comb filter vectors

```
[FreqEst, BEst, featureVector, M] = icf(bestIn, sample,icfBinOffset);
```

Listing 8: Create output text files for LibSVM

```
libsvmwrite('outputTrain.txt', trainingLabels', sparse(trainingMatrix));
libsvmwrite('outputTest.txt', testingLabels', sparse(testingMatrix));
```

A classifier followed the structure shown in Listing 9. Using *grid.py* the parameters for the function *svmtrain* were optimised. The parameters shown in this listing are not those used in classification as these changed depending on what *grid.py* suggested.

Listing 9: Classification program

```
[y, x] = libsvmread('Training_data.txt');
model = svmtrain(y, x, '-t 2 -c 100 -q');

[m, r] = libsvmread('Testing_data.txt');
[predict_label, accuracy, dec_values] = svmpredict(m, r, model);
```

Fret position prediction

Acquiring possible fret positions

The *findPossibleNotePositions.m* function takes a frequency as an argument and outputs a matrix containing all of the possible positions for that note below fret twelve (the octave of the open string) on the guitar. The function will accept a two percent error to allow for inaccuracy in tuning and more importantly, variance in frequency due to change in intonation in different fret positions. The two percent value was chosen experimentally and returns the correct note with one hundred percent accuracy for all of the samples I have tested.

Predicting the exact fret position

Initially, predicting the fret position using inharmonicity was considered as a method for deducing the position of a note on the neck, the measurements for the inharmonicity are shown in Figure 2. As inharmonicity is considered a measured of the amount of harmonic content in a signal it was thought that there would be a trend with fret position, however as this graph shows there is not a clear trend for the inharmonicity of each string and fret on a guitar. As such this cannot be used as a predictor for the position of a note on the neck of a guitar.

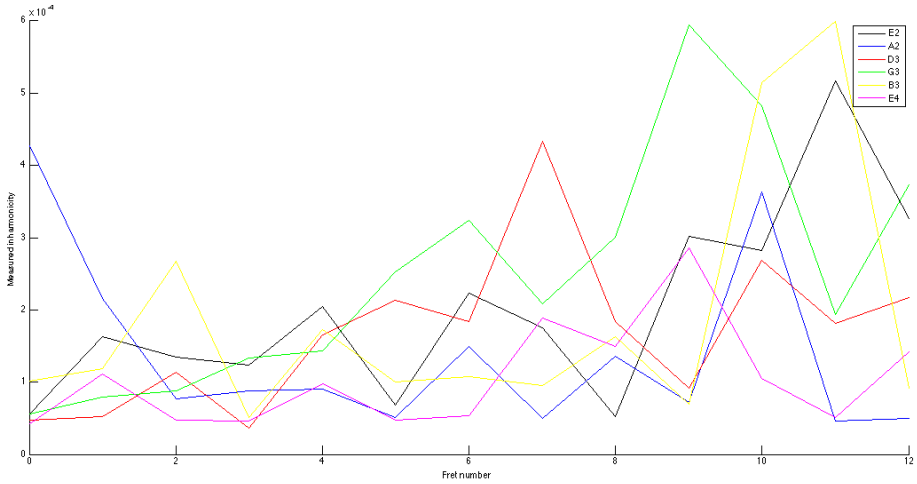

Figure 2: Inharmonicity for each string of a guitar in standard tuning for frets 0 to 12

By using a constant value in the inharmonicity prediction, there was no change in accuracy of the string and fret prediction show in Table 1. This is important as it means that inharmonicity prediction can be a fixed value. This is essential for building a classifier for this data as when classifying audio to extract a specific fret position, an accurate inharmonicity cannot be used because values such as length and string diameter are variables in the inharmonicity estimation equation (Figure 1).

Instead of using the inharmonicity measured by the ICF, the approach of summing the total energy of the feature vector array was taken to produce a comparison between the energy produced at each fret position. The three dimensional bar graphing facilities of Matlab helped immensely in the development of this method. These graphs made it possible to visualise all of the feature vectors for each fret position of a given string. This made spotting trends a much easier process. Figure 3 shows a typical set of feature vectors for the D string of the test guitar. This directly led to the discovery of the approximately linear trend of the decrease in energy as the fret position is increased.

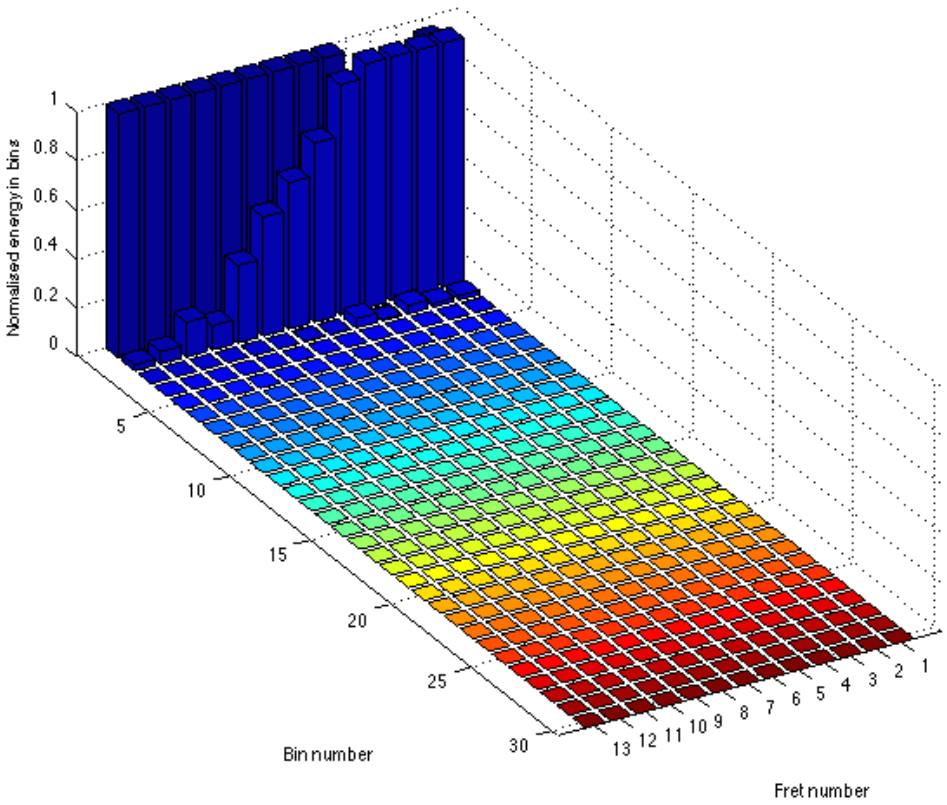

Figure 3: Feature vector for frets 0 to 12 (marked as 1 to 13) on a guitar D string

By measuring the relationship between fret position and total energy of all of the bins of a feature vector for a given series of notes the graph shown in Figure 4 was produced. This shows the linear trend in a much clearer way than with the bar chart and proves that (at least for this dataset) it is possible to predict the fret position on a guitar by measuring the total energy of the feature vector produced by the ICF.

The input values to calculate the inharmonicity coefficient are shown in Listing 10. These are approximation of the setup that was used on the guitar that the samples were recorded on

Listing 10: Inharmonicity input values for the estimation

```
1      stringDiameters = [0.25 0.33 0.43 0.76 1.066 1.3208];
2      stringTensions  = [7.36 6.98 7.52 11.22 11.61 9.59];
3
4      d    = mean(stringDiameters);
5      t    = mean(stringTensions);
6      ym   = 200;
7      l    = calcFrettedStringLength(610,0);
8      BEst = inharmonicity(d,ym,l,t);
```

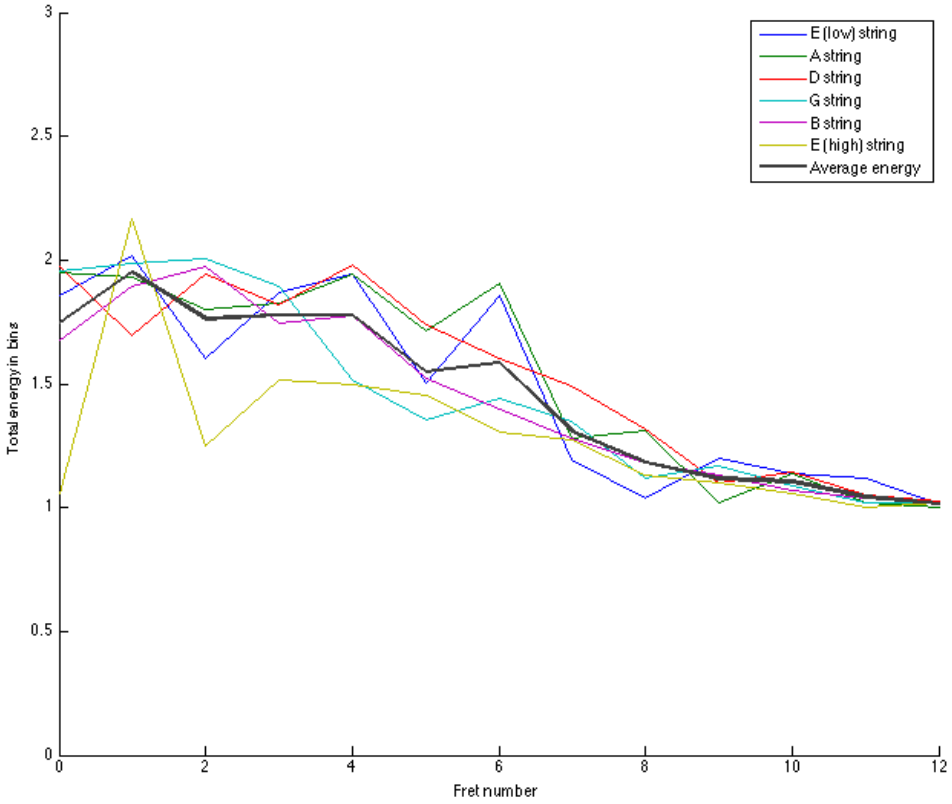

Figure 4: Total harmonic content of a guitar across all of its strings

A linear relationship is assumed for simplicity, whilst this is not totally correct, by studying the graphs of the total energy against fret number it was decided that the high fretted notes had a lower total energy (on a normalised scale) and low fret and open strings had higher total energy. This relationship was approximately linear, although with more available time, more care could be taken to deduce a general curve that could potentially give better matches.

Currently this test has only been conducted with one guitar (due to lack of adequate testing data). Therefore with more testing the profile of this line may be optimised to work with a wider range of guitars.

By calculating this linear relationship for normalised feature vectors over a range of frets and strings for a given guitar, it is then possible to predict the rough fret position for a normalised feature vector (of a given plucking intensity).

The program shown in Listing 11 is the function that is used predict the string and fret position. This function uses the ICF method to obtain both a fundamental frequency prediction and the feature vector, the feature vector is normalised and the total energy in it is summed. The possible fret and string values are calculated using the predicted fundamental

frequency. A linearly spaced array is created with the value max, min and range values given in the input. The closest value to the summed feature vector's value is found in the linearly spaced array, the array index of this value is returned. This index is the approximate fret value of the sample. The closest value to index is located in the possible fret positions for the given note. The index is then returned, this index is then used to retrieve the closest matching string and fret value from the matrix of possible predictions.

Listing 11: String and fret position prediction

```
function output = predictPosition(sample, BEst, binOffset, max, min, range)

    [FinalF0Est, FinalBEst, featureVector, M] = icf(BEst, sample, ...
       binOffset);
    totalEnergy           = sum(normalise(featureVector,1));
    possibleNeckPositions = findPossibleNotePositions(FinalF0Est);

    estimateFretPossibilities = linspace(max, min, range);

    tmp = abs(estimateFretPossibilities - totalEnergy);
    [idx idx] = min(tmp);
    predictedFretPosition = idx; %closest value in to match on neck

    tmp = abs(possibleNeckPositions(:,2)' - predictedFretPosition);
    [idx idx] = min(tmp);

    output      = zeros(1,2);
    output(1,1) = possibleNeckPositions(idx,1);
    output(1,2) = possibleNeckPositions(idx,2);

end
```

Problems encountered

There were a couple of problems that slowed the project down when a critical bug appeared, development would grind to halt. This was because the nature of most of the work in the project work done earlier in the project. A specific example of this was the ICF which would seemingly break for no particular reason when used with certain data sets. This meant that the program had to be re-worked to an extent that was not planned for in the initial time frame, which delayed its usage in further research.

Lack of data was one of the most frustrating general problems and it's significance didn't really become apparent until the research portion of the assignment neared it's conclusion. This was of particular significance when testing the program that retrieved the exact fret and string position of a sample. This was due to the fact that only one of the available sample sets (one that had been specially recorded for this project) has enough samples per string to test the theory that higher fretted notes have a lower total harmonic content than low fretted strings. All of the sample sets gained from other sources firstly had too few samples

per string to do a proper gradient prediction and secondly due to the lack of higher fret position samples there was not enough data to produce reliable testing for these guitars.

A mistake was realised very close to the deadline of the project which caused a big disruption to the results. Initially it was thought that an approximately 90% classification could be achieved for the different guitar type classifier. This however was increased to 100% by increasing the cost. This seemed suspiciously accurate and as such the program that wrote this data was re-examined to check for any errors. It was then found that this program wrote training data into the testing data file instead of the testing data. This is clearly unacceptable and was rectified immediately as such the classification rate dropped down to around 30% in an initial test. By simplifying this test to acoustic and electric guitars a much better result (around 80%) was achieved. These testing accuracies are expanded upon in the results section.

Classification of plucking intensity is still not successful, at a rate of less than 40% classification, the model is obviously not working as it needs to, hypotheses for why this is the cases have been outlined in the results.

Organisation of audio sample data was also an issue, although nowhere near as significant as the lack of data. It would have made the programs simpler, easier to understand and easier to write if all of the data was in a well designed database. Being able to query the data using a query language such as SQL could have made it substantially easier to label the data for a classifier with more specific classes. It could also make it easier to write less code as currently reading the files takes up a large proportion of the lines in the classifier. With a database query approximately sixty to eighty lines of code could be reduced to less than twenty. Another benefit of using a database would be that tags could be stored in the database rather than the file name, the file name could then just be the same as it's ID in the database (a large unique random string) in a flat folder structure. This would make the data available about an audio file more descriptive an more homogeneous meaning greater clarity when querying datasets. I would however approach the project in exactly the same way (and not produce a database) with the given time constraints as too much time would have been lost at the beginning trying to understand the data and its limitations. I am in a much better position after completing this project to design a fully featured database as I have also gained a better appreciation for what tags may be useful in classification and what data might be.

Another problem encountered during this project was lack of time, the modification of the ICF program was fraught with problems (primarily trying to fix the out of bounds errors amongst other things) and took a very long time. As such fret and string prediction were only completed toward the end of the implementation phase, which meant that there was not enough time to re-design the plucking intensity classifier after learning how great the effect of fret position was on the sum of the harmonic content.

Results

Various tests were designed to discover if it is possible to discern different features from guitar audio. This was done using techniques such as machine learning and conventional DSP.

Classification of different guitar types

A test was conducted to see if it was possible to recognise different guitars using LibSVM. Single note samples were used from a variety of guitars. The feature vector was extracted from these guitars using an ICF.

To obtain the optimal parameters for machine learning *grid.py* was used. This script tests a range of parameters for a training set and outputs the most effective parameters that can then be used with the repeated random sub-sampling tests mentioned in the design section.

This method has achieved mixed results, initially a 39.3357% rate of classification for an eight class classifier made up of different guitar sample sets was produced using *grid.py* to automatically generate the optimal parameters. However when these classes were consolidated into two distinct categories (acoustic and electric) a rate of 68.7251% classification was achieved again using the *grid.py* tool. By removing a nylon string guitar sample set (because the tonality is different to a steel string acoustic) and replacing it with a standard steel string acoustic guitar sample set an 80.0712% result was achieved when run with *grid.py*. Whilst this is certainly a result that can be improved on it is not bad and shows that there is a measurable difference between electric and acoustic guitars.

The results are interesting as it shows that only two general models would be needed in transcription programs and that trying to classify any further would be a fruitless task. This may make approaches to transcription simpler, meaning that the harmonics profile can be considered similar across a range of the same notes for the two guitar models.

This result may also have benefits for adding extra semantic content to pre-recorded guitar audio and may be of use in music recommendation engines for automatically tagging whether an acoustic or electric guitar was used on a song.

String and fret prediction

The hypothesis for this experiment was that playing a note higher on the guitar's neck reduces the ringing of this note and that this ringing is a characteristic of the overtones of the guitar.

By extracting the harmonics of the given note (as a linear feature vector) and normalising the note based on the amplitude of the fundamental frequency, it is possible to compare the total energy of a feature vector against a curve or line that compares fret position to note.

The test that was executed was simply to give the classifier a note that had an attached string and fret position in its labeling. These samples were marked up in a way that made

the comparison very simple due to the fact that fret positions were given as integer values from zero to twelve (the range of the frets that the notes spanned).

The average accuracy is 87.1795%, which means that sixty eight out of the seventy eight samples on the guitar were classified correctly.

Table 1: String and fret prediction accuracy

String (Note)	Accuracy Variable length, tension and diameter string in model (%)	Accuracy Constant length, tension and diameter string in model (%)
E2	92.3077	92.3077
A2	92.3077	92.3077
D3	92.3077	92.3077
G3	92.3077	92.3077
B3	76.9231	76.9231
E4	76.9231	76.9231

As shown in the Table 1 the accuracy is clearly better for the lower frequency strings. This means that there is a more defined difference between the amount of harmonics for the lower strings than the higher strings and that the values match more closely to the average sum of the total harmonic content for each fret for those given strings.

Table 1 shows that there is no affect on the accuracy of the fret prediction when using a constant inharmonicity. This is extremely useful as it means that it can be treated like any classifier and have only an audio sample input rather than needing approximate string lengths and fret positions.

A problem with this test is that it lacks a large amount of data to back up these claims. One guitar at a single plucking intensity isn't enough to truly validate this model. Multiple guitars with a variety of playing styles (finger picked, plectrum picked and slap) and note lengths would really be needed to be tested before declaring complete accuracy of the model.

The benefit of this method has interesting implications for guitar transcription and guitar to computer interfaces. Currently, it is the case that if one wants to transcribe a guitar directly from their playing a Musical Instrument Digital Interface (MIDI) interface with their guitar must be used. MIDI interfaces require modification to the guitar in the form of a hex pickup. This signal is then converted into MIDI by a microcontroller. The downside to guitar MIDI it only works with and outputs a monophonic signal.

If a purely software solution is chosen more information about the specific position may be derived, but I cannot find any evidence of another system that can accurately estimate the position on the neck of the guitar further than translating the fundamental frequency into the lowest position that a note occurs.

In some senses MIDI is not the optimal format to translate this signal to as it cannot

account for the different fingering positions of the same note on the guitar. This means that a format such as MusicXML [16] may be more appropriate. The different fingering positions are probably most relevant when it comes to structuring chords as these can be played at different parts of the neck. These may also be useful for monophonic sequences where a player would like to express their preference in the tablature about where the sequences of notes be played.

As most commercial models are kept secret, one could speculate that current models use a statistical method combined with a standard frequency analysis using a FFT. This still treats the guitar as if it were a chromatic instrument like the piano (which it is not because of the multiple playing positions for most notes). This means that it is possible to tell the difference between an open and fretted string on a guitar something for which I have not been able to find any research suggesting that this has been done before.

Whilst writing this report out of curiosity (and mild panic) I did decide to quickly record some test samples with a *Squier Stratocaster* (with no amplification or effects and using a *Focusrite Scarlett 8i6* and *Audacity*). All of the four notes were recorded on the low E string, with two plucking intensities recorded for each position, making a total of eight samples, fret positions 0, 3, 7 and 12 were used to get a reasonable range. These samples were used as an input into the *predictPosition.m* function (along with all of the values used as input for the previous sample set). The result was extremely positive, with 100% classification for these samples. More samples would need to be tested to rigorously verify this model but this result should be of some note as the guitar used is different to that of the guitar originally used for developing this model.

Classification of different plucking intensities

Initial research

The results of a comparison between the plucking intensity of an open A string played at mf and ff are shown in Figure 5. As this chart clearly shows there is a difference in the magnitude of the harmonics of the histogram as was initially hypothesised. There are some small dips below zero which indicates that the magnitude of the more gently plucked sample did exceed that of the more heavily plucked sample at those points, this could however be due to very slight differences in the fundamental frequency of the note, causing a small shift in some of the harmonics.

[16] http://www.musicxml.com/tutorial/tablature/fret-and-string/

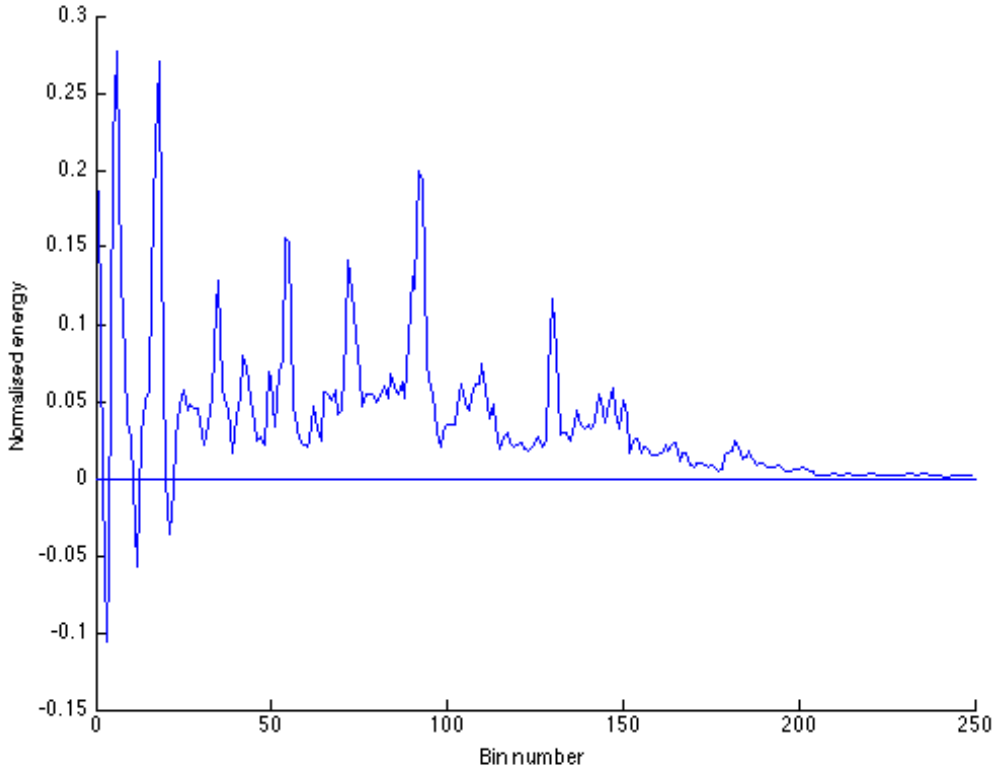

Figure 5: Difference between plucking intensity of open A string of a guitar

This followed that if the intensities from a set of recorded samples are normalised and labeled correctly then a classifier could theoretically be built to tell the difference between samples.

Classifier

After various attempts to get a positive classification between samples with a different plucking intensity this method does not seem to be as easy to distinguish as initially thought. A classification rate of 37.8705% has been achieved using *grid.py*. This shows the classifier as it is currently designed is not anywhere near effective.

The best visual representation to explain this that could be produced was a scatter plot of the sum of the feature vector shown in Figure 6. Plotting each feature vector in a three dimensional bar char would have been too cumbersome due to the large amount of feature vectors and as the distribution is not even the If the data was easily separable by the classifier it would display a somewhat stepped pattern with one cluster in the bottom left showing a low energy, one cluster in the center showing a higher average energy and one cluster in the right-most band showing. However as the energies are all over the place.

This phenomena is possibly explained by the fact that less total energy is measured in

higher fret positions on the neck of the guitar (as proven in the string and fret position experiment). An obvious improvement to make to this model would be to separate the classifier into fret and plucking intensity. The problem with this idea is that for a twenty two fret guitar with three different levels of plucking intensity per fret there would be three hundred and ninety six classes which may be an unworkable number of classes. The problem with having such a large amount of classes in an SVM is that it may be hard to separate the data if there are not very distinct boundaries between each class. Another problem with using that many classes may be that classifier itself is not optimised for this many classes and as such may perform poorly. This is just speculation however, I cannot find any solid evidence to back this claim up so further testing would need to be done to verify this.

An interesting note about Figure 6 is that a large proportion of the values lie between two and one. This may suggest that little change needs to be made to the fret position classifier.

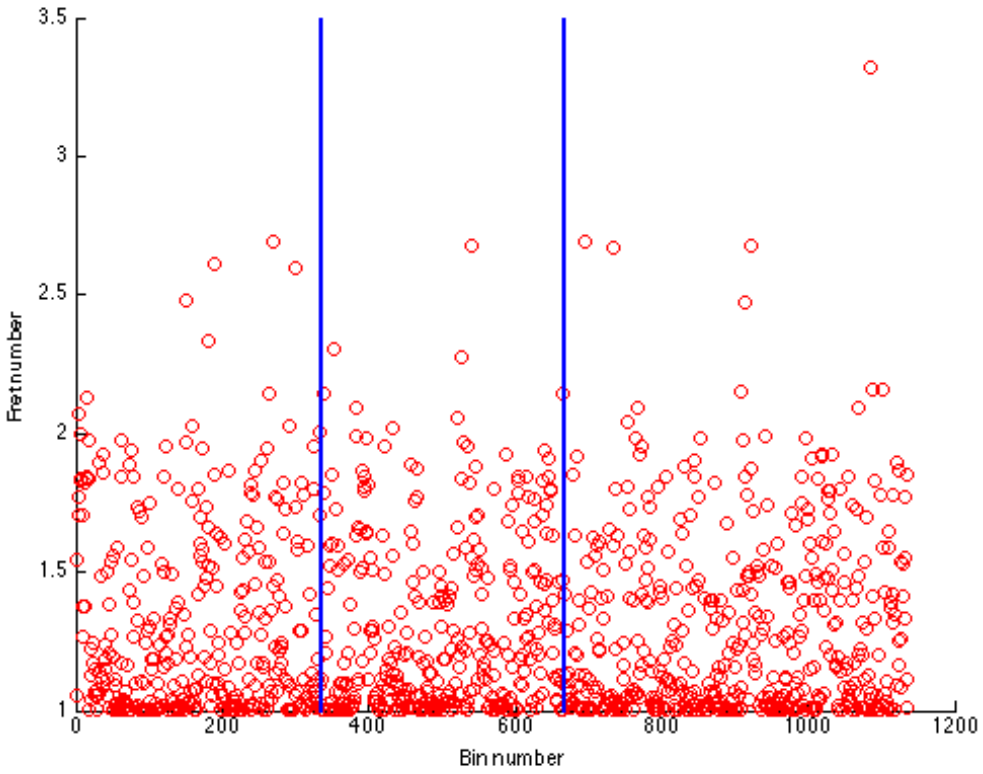

Figure 6: Total harmonic content of a multiple guitars for different fret positions

As this problem has not yet been fixed a few reasons for the classifiers lack of success can be speculated:

- The different guitar sample sets have different metrics as to what is considered a heavier and a lighter pluck on the string, despite normalisation this will still be a problem due

to the fact as this seemed to vary between guitar sample sets. Different notation for different guitar sets was assumed to be the same, which is not necessarily a safe assumption.

- The classification rates of individual data sets (each set contains samples recorded on the same guitar) have also been tested, all of the results still exhibited extremely low classification rates.

- Different harmonic profiles of different guitars, one type of guitar may exhibit (this is very probably the case as it has been shown during this report that guitar type classification works extremely well).

- As seen from the tests, total harmonic energy of a feature vector decreases as it's fret position increases, therefore due to the fact that a range of fret positions are used in the model it may be advisable to split this model up into more classes by fret position.

Another way to achieve a plucked intensity may be just to use amplitude measurement. A nominal amplitude would have to be chosen for quiet or loud notes and the sample would then be compared against this. This would not take into account the level of gain in the recorded sample though, this is problematic as amplitude normalisation could not be used as it would get rid of the difference between amplitude that we would be trying to measure. One way of choosing a nominal value would be to measure the average energy of the signal and use this as the nominal value. This brings it's own problems and assumes that playing would be at a standard level intensity which may not be a correct assumption. This approach may struggle when multiple frequencies are played at once (specifically in chord tones). In this case an EPCP may be useful as it could detect the number of notes being played and adjust the amplitude prediction accordingly

Further work

Possible extensions to this software could include any number of the ideas listed here.

Data acquisition and storage

Ideally multiple sample libraries would be contacted, with the aim to get a discount for buying sample packs in bulk. Licenses would also have to be verified to see if they are appropriate for this application.

Information that could be requested when purchasing sample packs:

- Accurate documentation of what all of the annotations in the file names stand for (rather than having to make educated guesses). The documentation on naming conventions would be especially useful for database creation.
- The type of guitar used (down to the exact model if not already known).
- The string make, type and diameter (commonly referred to as gauge) and potentially even the level of wear on the strings. Knowing the average guitar and string properties would make the string modeling more accurate and would possibly bring more accuracy to that model. Whilst only mean values are currently necessary a further use may also be derived for this information.

In addition to this, it may be useful to record a variety of guitars, this could be done professionally or by the researcher. This may be useful as more information could potentially be verified and captured to increase the richness of the tags in the database. This extra information may make it possible to have more specific classifiers. Owning the samples may provide some additional benefits, it would allow the data to be built into a web accessible API for] researchers and programmers. Making these data sets available could allow a greater range of researchers to work with the data and may even help to improve the field of computational audio.

Something that would be extremely beneficial to the future of this project would be a database. It was decided from the beginning of my implementation of the project that there simply would not be time to complete the analysis if two or three weeks were dedicated to trying to structure the files and build a database. This assumption was reasonable one as only the three main research aims were completed in the given time.

Benefits of a database would include:

- Improvements to project structure, the files would be put into one folder with a unique identifier given to each sample. A link to this folder would then be stored in the database. This would mean that if the folder needed to be moved it would simply require changing one database value.
- Simplifying the process of selecting file for building a new classifier. A database query would probably be a few lines of code, whereas reading each directory and dealing with the

- More completeness in each model classifier. As filtration to the chosen samples it would be easier to reach into sample sets further and have more
- Pre-processed data could be stored in the database. As the ICF only produces a maximum of thirty points this would not be a problem for any database to handle. This would reduce overheads associated when creating classifiers. This could easily be created when data is uploaded to the database.

Classification

Only three classifiers were built during this project. These features may also be classifiable using SVMs or other techniques.

Features that could potentially be classified in the future include:

- Palm muting.
- Picking type, eg. finger picked, hard plectrum, soft plectrum (this could have been successful with the bass guitar in particular).
- Different stringed instruments such as violin, mandolin, ukulele.

Some features may also benefit from a time domain analysis so it may be beneficial to produce an Short Term Fourier Transform (STFT) for these signals instead of using the ICF. Some time warping would probably need to be applied as the length of time for these features can vary greatly.

- Bent notes should in theory be quite easy to predict as the frequency change will be smooth rather than sharp.
- Notes played with vibrato should exhibit smaller frequency variations than bent notes but could be analysed in a similar same way.
- A slide will firstly exhibit a change in frequency over the given time like any other note change, but it will have a different attack at the start of the note as there is not pluck. This may benefit from beat a detection algorithm to show for the given frequency that there is no attack at the beginning of the new note.

Improvements to the fret prediction algorithm

Instead of using a linearly spaced array to predict fret positions, the total harmonic energy for a normalised feature for a range of different notes on different guitars could be put into a LibSVM model with the label for this data being the fret position. The *grid.py* optimisation program should be used to obtain the settings that yield the highest accuracy.

Classification was not attempted as there are only seventy eight appropriately labeled samples. There has simply not been enough time to record more samples as this was the last

piece of research that was worked on. Using an SVM may lead to a more flexible model that can deal with a wider range of guitars, plus it would remove the need for having the linearly spaced array which works quite well for this guitar, but may not work as well if other guitars are used.

Support for alternate tunings

This would be a simple program to build as it would simply consist of deducing the possible range of notes for guitars. The values could be calculated computationally a string of a given open string note. Alternatively an open string note and number of frets could be given for a string and the frequency value for each note of that string could be retrieved from a matrix of all possible notes for a guitar.

Transcription engine using fret prediction model

The fret prediction model works reasonably well (87% classification rate which may even be improved upon), even if the notes produced are not at the correct fret position, they will still be in another fret position.

A quick test was completed (but not included) to try and attempt to produce a worked solution to this problem. The solution was very simple, a simpled monophonic guitar sample was produced and a window size was specified for the ICF to work for. This however did not work as the ICF does not work with data that is zeroed, such as the silent parts at the beginning of the audio track (and in any other parts). To get this to work correctly the silent part of the audio needs to be removed. The best way to do this might be by using beat slicing algorithms (such as calculating the average energy of a sample and computing the instant sound energy to see where possible note starts could occur), to predict when a note is played and then isolate this note into. Another method could be to use a STFT to compute the frequencies for the a complete audio track and use the ICF exclusively on areas that definitely contain frequency content.

Polyphonic transcription using summed energy approach

As it has already been shown in the report a reasonable approximation can be made for the finger position of played notes on the guitar. Using this principle it may be possible to use the energy prediction for different chord positions on the fretboard in conjunction with template feature vector methods such as EPCP. By calculating an approximate profile for a chord, this could then be matched to possible chord positions on the guitar for a given template, allowing for an algorithm that can predict chord positions.

Statistical modeling to produce more accurate fret position metrics

This method would only work for improving the accuracy for sequences of notes. This would not be a problem for a full transcription engine which are designed to specifically to analyse

sequences of notes.

The distance between notes would be measured computationally and the closest note would be picked likely mimicking how a guitarist would play a sequence.

Study the effects of frequency shifts

A small study into the effect of intonation on frequency for different guitar samples may be useful. This may help further with high frequency fret prediction. If a small frequency shift has occurred and there the harmonic content is not much greater than the fundamental this may give an indication that this sample is played at a high fret. Pitch shift can also be observed when plucking a note harder, this effect can be observed when tuning a guitar with a digital chromatic tuner and plucking a two very different intensities.

Classifier comparisons

Testing different machine learning approaches, LibSVM seems to have worked quite well once it was set up correctly but it would be interesting to see if optimisations such as genetic algorithms would have been a suitable approach and if the data would have been needed to be processed differently. K-means clustering is an unsupervised learning technique that could be tried, this is useful as it means there is no need for separate testing and training sets, the labels for the data are the verification.

Determining type or style of guitar playing for complete songs

Mel-Frequency Cepstrum Coefficients (MFCCs) can be used as a similarity measurement for audio files [4] [5]. This may be particularly useful when used in conjunction with SVMs in trying to deduce the general style of of guitar playing. For instance MFCCs could be used to analyse hundreds of different types of songs that contain guitar music where the style has already been labeled. This would be especially useful for adding tags in a music database such as Musicbrainz [17]. and would mean that these specific tags would not need to be added manually by record labels, owners or users of the system. This could be an extension to the backed of services such as Spotify [18] or Last.fm [19]. This technique would capitalise on the fact that guitar is often processed in a specific way for the genre.

MFCCs may be most practical at determining the difference between types such as:

- Funk guitar, which has a heavy use of wah-wah filters and often palm muting.
- Heavy metal, which predominantly contains moderately to heavily distorted guitar signals so should exhibit a stronger noise characteristic.
- Slap bass tones, which are very distinctive with sharp attacks on low notes.

[17] http://musicbrainz.org/
[18] https://www.spotify.com/
[19] http://www.last.fm/

Software such as Celemony's Melodyne [20] allows tracks to be split into individual instruments. The guitars from these tracks could be analysed separately from the rest of the track, this may provide a useful way to provide further functionality to an existing program.

[20] http://www.celemony.com/en/melodyne/what-is-melodyne

Conclusion

Three hypotheses have been tested:

1. Different guitars have different tonal characteristics and as such it is possible to differentiate between them using signal analysis.

2. The harder a note is plucked, the more energy that can be measured in the harmonics of a normalised sample.

3. Different fret and string combinations of the same not have different tonal characteristics which lead to a more accurate string and fret measure being produced

The results of classification were on the whole positive, a relatively accurate classifier for classifying electric or acoustic guitars was built, as was a classifier that can locate a fret position more accurately. The classifier for plucking intensity did not work as well, however methods have been suggested for improving this. the initial

One of the significant problems faced during this project was obtaining guitar audio samples that were of a high quality and that were well annotated with references that were common amongst the sets. Sample packs range greatly in price. Some are free (like all of the samples used in this project) but prices go into the hundreds of pounds.

Annotation was not always particularly obvious for the formats and there was a small amount of guessing done by ear trying to perceive the difference between samples. Whilst not particularly tricky, it made the whole process of selecting and verifying samples a bit more problematic as patterns had to be manually searched for rather than looking through documentation that could have provided a better insight into the naming conventions of the sample bank.

As suggested in the further works there is a lot that could be done with the ICF. One of the most important next steps would be to start the building of a database. Firstly this would reduce the complexity of code but if this database was exposed to other researchers via an application programming interface (API) it may have wider reaching consequences for future of guitar audio machine translation.

After building a database, constructing classifiers would become more trivial and the speed of design would only be limited to the time taken to construct database queries.

Reflection on learning

Extracting relevant semantic data from guitar audio is a vast and complex problem space, this project if nothing else has made me appreciate the complexity of this trying to extract this data, but has definitely given me good ideas of how features could be extracted from guitar audio samples for further analysis.

Machine learning

Without this project I would have had a much more simplistic view of the practical capabilities of machine learning. It has helped me to appreciate the need for properly filtering the input data to ensure that clustering is possible and given me a further interest of how machine learning and similarity and clustering algorithms can be applied to other problems. I have always been of the firm belief that a strong knowledge of tools and APIs makes a computer scientist's job easier as I think it is important to use other developers and researchers work as a foundation. As Sir Isaac Newton famously wrote, *'If I have seen further it is by standing on the shoulders of giants'.*

Digital signal processing

A broad understanding has be gained for the techniques that can be used to analyse audio and extract feature vectors. The project has helped solidify the mathematical and analytical techniques covered in both the Scientific computing and Multimedia modules and has provided an interesting application outside of what could have been achieved in any single module.

Organisation

Whilst the initial plan was followed to an extent, delays in this were primarily because bugs and programming. Due to the project there was a very linear nature of testing ideas and hypotheses and as such when problems were faced it could mean days of trying to rectify one bug with nothing else to work on. Now the tools to complete the analysis have been developed it would be much easier to build and run experiments in parallel this however wasn't really an option during the project. I did not get a chance to test the linearisation of the feature vectors which may have provided some benefits to the classification The modification of the ICF was not as trivial as I first thought as making it work for higher frequency sample sets. It would also have been desirable to build two or three more classifiers to see how broadly the approach could be applied but there was not enough time left to either collect the data or to implement the classifiers.

Programming ability

The Matlab programming language has been used almost exclusively for this project and as such I have developed better ways of working and designing programs. This mainly consisted of knowing when to separate code into functions for a better separation of concerns. Mistakes were made during the implementation phase that were not caught until writing this report, these gave a completely inaccurate set of results for one of the classifiers. I am extremely glad that I caught this mistake when I did however it is worrying that one simple variable being named incorrectly meant that all of the testing data had been written incorrectly and was training data. As soon as this error was spotted it was quickly rectified and new tests were run. Errors such as this are possibly due to me not adhering strictly enough to the don't repeat yourself (DRY) principle.

Bibliography

[1] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] A. Galembo and A. Askenfelt. Signal representation and estimation of spectral parameters by inharmonic comb filters with application to the piano. *Speech and Audio Processing, IEEE Transactions on*, 7(2):197–203, Mar 1999. ISSN 1063-6676. doi: 10.1109/89.748124.

[4] Jesper Højvang Jensen, Mads Græsbøll Christensen, Daniel PW Ellis, and Søren Holdt Jensen. Quantitative analysis of a common audio similarity measure. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):693–703, 2009.

[5] Sefki Kolozali, Mathieu Barthet, György Fazekas, and Mark Sandler. Automatic ontology generation for musical instruments based on audio analysis. 2013.

[6] Kyogu Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *Proc. of the International Computer Music Conference*, 2006.

[7] Harry Ferdinand Olson. *Music, physics and engineering*, volume 1769. Courier Dover Publications, 1967.

[8] Martin J. Pritchard. Musical analysis of audio. Master's thesis, Cardiff University, 2002.

[9] M Ungureanu, C Bigan, R Strungaru, and V Lazarescu. Independent component analysis applied in biomedical signal processing. *Measurement Science Review*, 4(2):18, 2004.