Recognition of Music using Fourier Analysis and Time Correlation

Theodore Hirt Madden (tdhm1g20) 31600999

University of Southampton

An audio signal can be decomposed into the frequency domain via the use of Fourier analysis. This technique allows a computer to mimic the human ability to understand and interpret a complex audio scene. This paper investigates the basis for a computer song recognition algorithm by decomposition of an audio signal into the frequency domain and performing time correlation analysis on a sample of music. Given a sample of a song with a noisy background, the accuracy of frequency and time correlation techniques to identify a song is calculated for varying noise levels and types of background noise. The free parameters, sample duration, length of a time segment and bin number are explored and optimised.

Introduction

I have always been interested in the Shazam algorithm; how it could identify songs playing in a complex audio scene containing many different types of noise. I decided to develop my own algorithm from scratch to recreate the algorithms that are possibly used by many mainstream song identification algorithms (Wang, 2003).

Song identification has become a vital aspect of a consumer's life since the globalisation and accessibility of music through streaming services like Spotify. Many people are sharing songs or hearing songs that they enjoy when socialising, clubbing or just out of the house. With access to a powerful mobile computer, the ability to identify a song on the move becomes a reality.

Song Identification is difficult. There are many instruments and sounds that can make up general music. Furthermore, if a sample of a song is being recorded in real time, the difficulty in calculating what song might be playing grows with background noise, microphone quality and sensitivity, audio encoding, etc. There exist millions of songs in the world along with many covers of the same song. A song identification algorithm would not only need to be accurate, but also efficient to compare millions of songs.

The nature of this problem requires the use of two techniques to allow accurate identification of a song from a sample. The two techniques used were Fourier analysis and time correlation.

Fourier analysis is the decomposition of a function into an infinite sum of trigonometric functions. Given a function $f(t)$ the Fourier transform can be written (Kenna-Allison, 2021-2022),

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

Equation 1 The Fourier Transform, where ω represents the frequency of the sinusoid, t in this case is time.

In the case of digital music, the waveform is limited by the sample rate, the number of times per second that the continuous sound wave is sampled and evaluated to generate a digital signal, that the music is recorded at and thus the waveform function becomes discretised.

With a discretised signal the Fourier transform can only be evaluated at specific intervals, $n\Delta t,\ n = 0, 1, 2, 3, \ldots$ as the integral only exists at these values. The discrete Fourier transform can then be written,

$$F_{discrete}(\omega) = \sum_{n=0}^{N-1} f(n) e^{-i\omega n \Delta t}$$

Equation 2 The discrete Fourier transform, where ω represents the frequency of the sinusoid, f(n) represents a sample of the continuous function, Δt is the separation in time of the samples, N is the number of samples in the signal.

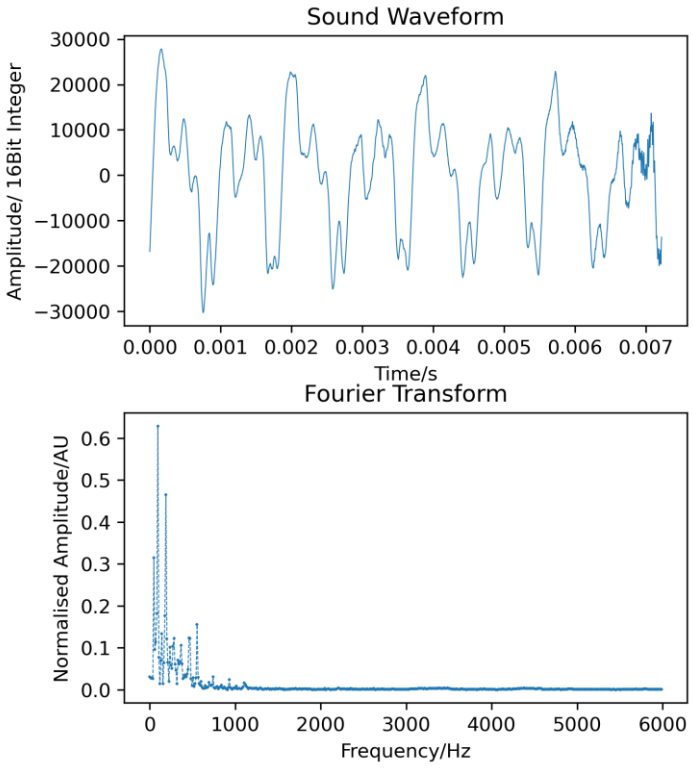


Figure 1 A plot of a sample waveform and it's corresponding Fourier transform. Showing the contained frequencies in the signal.

Fourier and audio go hand in hand. Digital music would not have been as prevalent without the use of Fourier analysis. Most notably, Fourier allowed for the compression of audio. By generating the fourier transform of audio data the highest amplitude frequencies can be maintained whilst removing the smallest amplitude frequencies and those that will not be perceptible to the human ear (Tewari, 2017). This means that space can be saved without impacting the quality of the sound.

The study of Fourier transforms is not limited to auditory analysis. Fourier transforms are also used widely in optics, especially shown by the prevalence of the field Fourier Optics. In Tyson (2014) The principles of Fourier optics are outlined. The effect of a lens as a physical Fourier transform is developed. This has impacts in optical image processing and cleaning of spatial interference from laser beams that has been generated from defects in equipment. There are further applications of Fourier in mathematics and computing, e.g., for the multiplication of polynomials and large numbers (Cantor & Kaltofen, 1991).

I will briefly outline the type of data that this paper deals with before discussing the methods and implementation of the algorithm. I discuss the free parameters and how they can be optimised. I will go over the types of noise that are most likely to occur when the algorithm is used and how this noise can be applied to data. I then show the results of the accuracies of the algorithm, the free parameter optimisation before discussing the advantages and limitations to the algorithm.

Data

This paper deals with digitised audio signals. I will give a brief overview of the type of data that this paper deals with and key terms.

Digitised music takes a continuous waveform and creates a digital version of this continuous wave. The digitisation is performed by sampling the wave at small time intervals $\approx 2 \cdot 10^{-5} s$ and the amplitude of the continuous function at that time is stored.

The amplitude of the continuous wave is stored usually as a 16-bit signed or 32-bit signed integer. The accuracy of the amplitude is based on the data type the continuous wave is stored as and this is known as the bit depth.

The sample rate is the number of times per second that the continuous function is measured, and a value stored. The sample rate affects the discernible frequencies in a song. The music and noise in this work used a sample rate of $48kHz$.

Methods

Basic Design principle

In the scope of the problem posed by this paper, the limitation of the Fourier transform is that the information in time is lost. If the transform is applied across the whole duration of a song, it will return all the frequencies occurring in the song with no information as to when they occurred.

Whilst the instruments and the recording scenario will be slightly different for different music, this audio information is mainly characterised in the higher frequency sinusoids of the Fourier transform and is highly affected by noise. The information of a note played, like A4 which is usually at 440Hz will generate a strong amplitude in the Fourier plane. As many songs use similar key signatures and instruments the overall Fourier transform could look very similar. Hence, time correlation needs to be used to improve accuracy.

Time correlation is a broad subject. For this problem, the frequencies of a Fourier transform need to be matched to the time in the sample that the frequencies occur. Then, when comparing a sample to a database of music, if the frequencies are matching, we can match the times that these same frequencies occurred. If there is a tight cluster of matching frequencies in time, then it is more likely that the sample has been drawn from the song that it is being compared to.

A whole waveform (sample or a whole song) is divided into time segments, and the Fourier transform is applied to each segment. It seems educated to assume that a slice would be optimum when the loss of information in time across the slice can be neglected compared to the duration of a sample and a song's length. However, I will explore how the length of a time segment affects the accuracy of the algorithm.

The algorithm used was developed in 3 iterations building upon each previous iteration to improve the accuracy to which songs could be identified.

I first give a basic summary of the process that the algorithm follows. Further in the text I discuss in detail each step of the method. The fundamental method is:

1. A given song is chunked into segments of time within the song.
2. Each chunk of the signal is converted from the time to the frequency domain
3. A sample of the song is taken, and noise is digitally added.
4. The sample is chunked into time segments and converted to the frequency domain.
5. The maximum amplitude frequencies in each time segment are stored in an array as a 'fingerprint'
6. The maximum amplitude frequencies for a whole song and the sample are compared to obtain the 'similarity'

Algorithm development

Each iteration of the algorithm dealt with how the 'similarity' between a sample and a song in a database can be generated to give accurate estimations on which song the sample is taken from.

I briefly discuss what each iteration looked at and then discuss why the final iterations were used.

Iteration 1

The first iteration looked at the occurrence of matching frequencies of the sample to a song in the database and estimated the correct song by the largest number of occurrences of matching frequencies.

Iteration 2

The second iteration of the algorithm built upon the previous iteration. Rather than simply counting the number of matching frequencies between a song and the sample, this iteration looked at the relative time in the song and the sample where the matching frequencies occurred. A cluster of matching frequencies in a short time frame is used as an indicator to which song the sample came from. A cluster is found by the maximum density of matching frequencies in a small timeframe.

Iteration 3

The third iteration looked at the frequency difference between the frequencies in the sample and a song in the database. Frequency differences outside of a specific range are discounted and then I calculated the time correlation between these filtered frequency differences. A cluster of small frequency differences in a small timeframe gives an indication of which song the sample is from. A small frequency difference is essentially a frequency difference of zero. However, to remove the pitfalls of floating-point inaccuracy I chose the range to be $\pm 1Hz$. A small timeframe would be defined by a high density within a time frame of length equal to at maximum the duration of a sample. A true cluster can be defined by finding a much higher density of small frequency differences, defined above, in a small timeframe, defined above, comparatively to the average density of frequency differences along the entire time axes.

The third iteration of the algorithm fixes three issues that occurred in the previous implementations.

1. Analysis only of the matching frequencies can give false positives for musical pieces written in the same key or with similar instruments as there will be many similar musical notes used in each piece.
2. 'Matching' frequencies requires exact evaluation of data point values which does not account for computer rounding errors, small unforeseen perturbations to the frequency domain, discrete Fourier transform errors.
3. Longer duration songs, longer being longer comparatively to the duration of the shortest song in the database, will lead to a higher average density of similar frequencies which could create false positives. When the true song duration is on the lower end of song durations in the database, a cluster could still fall below the average density of other longer songs in the database.

The third iteration of the similarity algorithm was the following, the exact values for the parameters used are not discussed until further into the paper, an outline of the method is shown here.

1. The maximum frequencies of the song and sample are found by splitting the frequency domain into a set number of bins, number of bins is discussed further down as a free parameter to be optimised and finding the maximum amplitude frequency in each bin.
2. The frequency difference between maximal frequencies for every bin in each time slice, in both the sample and a song from the database, is calculated.
3. The time difference between the time that the frequency difference occurred in the database song and the sample of a song is matched to each frequency difference.
4. The frequency and time differences where the absolute frequency difference is more than a specific value, as discussed above this was set to $> 1Hz$, are discounted
5. A histogram of the time differences where the frequency differences are less than $1Hz$ is computed with 250 bins chosen arbitrarily.
6. The difference between the peak in the histogram of time differences and the average of the histogram is calculated and gives the estimation of the similarity of the sample to the database song.
7. The song in the database that gives the highest difference between the peak and the average is the most likely song that the sample is drawn from.

Iteration 3.1

Iteration 3.1 correlates the times for the sum of the magnitude of frequency differences in each bin. This reduces the dataset size from iteration 3 by the number of bins used. This, without a significant loss of information, would improve the computation time and would be a more easily applicable algorithm to real world problems.

Iteration 3.1 reduced the size of the computation, whilst not necessarily reducing the overall accuracy significantly. I look to compare the accuracies of iteration 3 and iteration 3.1 further in the paper.

This change requires a different filtering process than for the previous iteration. It is not as clear cut as finding the frequencies where the difference is zero. When a sample and a song match there will be time segments with very low summed magnitude frequency differences comparatively to the average. Initially a filtering process choosing only data points less than the average or some factor of the average was chosen. This filtering method was used to visualise the

data, as shown below. This method is unstable as it does not use the data to calculate prominent datapoints and when the sample and song do not match the summed magnitude frequency differences will be distributed much closer to the average as there is no pattern. Hence, to improve stability the filter takes points that lie below two standard deviations from the average. When there is a pattern, the surviving points will be clustered together, and a match is found. When the sample and song do not match the surviving points will be randomly distributed.

The following figures show the time correlation between matching frequencies in a sample of audio and the song that the sample came from. The figures are based on iteration 3.1 of the algorithm as iteration 3 deals with too many datapoints for the visualisation of the data. The samples have been reduced to only the points that are less than $0.75 \cdot (mean\ frequency\ difference)$ as otherwise there are too many points to see a pattern. This gives a visualisation of what the algorithm searches for.

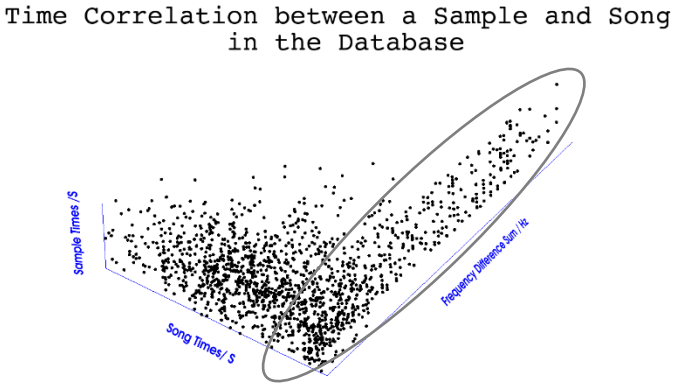
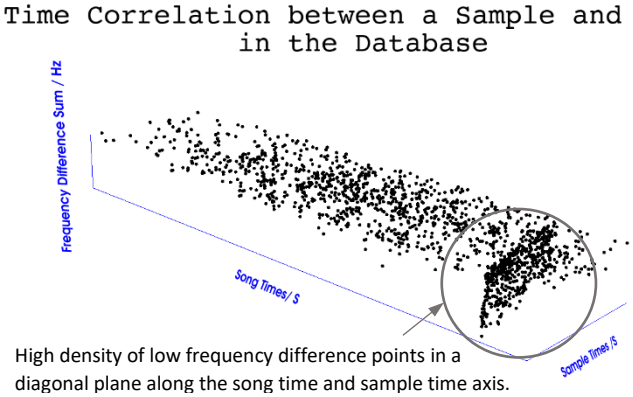


Figure 2 A plot of the total frequency difference against the time slice that the total frequency difference occurred in both the sample to be compared and the song from the database that the sample is being compared to. The total frequency difference is the sum of all the frequency differences of the maximal amplitude frequencies in the Fourier plane that were chosen from each range of frequencies. The grey oval highlights the points that are contributing to the pattern. The axes tick labels have been omitted as the exact values across each axis are not valid, I am giving a view of what the pattern looks like.

In Figure 2 it is difficult to see the exact pattern that the program is searching for. Highlighted by the grey oval, there is a higher density of low frequency difference points along a slice of the sample and song times. I scaled both the song times and sample times axis to highlight the pattern that form.



High density of low frequency difference points in a diagonal plane along the song time and sample time axis.

Figure 3 A replot of Figure 2 with the time axes scaled so that the frequency axis becomes compressed. The pattern showing a correlation between the sample and the song in the database becomes clearer.

Figure 3 starts to show the relationship that the program is searching for, a high density of points along a diagonal plane in the song time and sample time axis. The diagonal line is hard to visualise in the 3D space. I scaled the times axes further until the frequency axis becomes essentially a single plane and the 2D relationship can be shown.

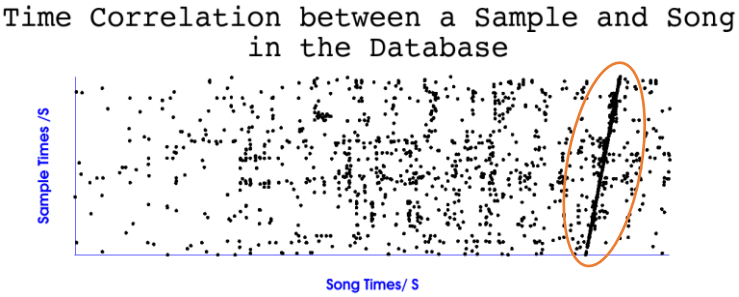


Figure 4 A replot of Figure 2 with the time axes scaled so that the frequency axis becomes a plane. The diagonal correlation pattern in the data is highlighted here by the appearance of a strong diagonal line in the time.

Figure 4 Shows the time correlation aspect of matching a sample to a song. The diagonal correlation in time of many matching frequencies between the sample and the song in the Fourier plane highlights a match.

Parameter optimisation

The algorithm depends upon many freely chosen parameters. The duration of a sample used to compare, the length of time of each slice that the sample and database songs are divided into and the method for choosing maximal frequency amplitudes. These parameters will have an impact on the performance of the algorithm.

The choice of duration of a sample has two aspects, the longer a sample the more information is available and therefore accuracy of the algorithm increases. However, the duration of a sample represents the length of time a person would have to spend recording a sample and this is affected by the questions, how long is left of the song playing? How long does a person want to spend holding their phone to record the sample? This aspect means that the duration of a sample is a free parameter that should be minimised. Thus, the question to be answered is, what is the shortest length sample of a song necessary that does not have a significant impact on the accuracy of the algorithm?

The length of time of a slice of a sample is not limited by application problems and therefore there are a few questions that need to be answered. Firstly, what is the optimum length of time that a slice should be? This question has a nuance, a decrease in the length of each slice will increase the computation as each slice adds to the overall data that needs to be run through the algorithm. Hence, once an optimum time slice is found, i.e., the time slice is minimised, the question becomes, what is the upper limit of the length of a

time slice before there is a significant decrease in the accuracy of the algorithm?

A further aspect of time slice optimisation is that the frequency spacing in the Fourier plane has a resolution of,

$$\Delta f = \frac{f_s}{N}$$

Equation 3 The frequency spacing in the Fourier plane, where Δf represents the frequency spacing, f_s is the sample rate and N is the number of datapoints that the transform is applied on.

The equation can be simplified as the number of data points is equal to $f_s \cdot (time\ segment\ length)$. Therefore, the frequency spacing is proportional to

$$\Delta f = \frac{1}{time\ segment\ length}$$

Equation 4 The frequency spacing in the Fourier plane, Δf is the frequency spacing.

Therefore, there will be a lower limit to the length of a time slice as the frequencies that can be distinguished in a sample will be lost if the length of a time slice is too low. There is a balance between improving either the accuracy in time or resolution of frequencies. The effect of altering either on the overall accuracy needs to be assessed.

The method for choosing maximal frequency is not quite as straightforward as a minimisation or maximisation of a value. There exist multitudes of ways that the maximal frequencies can be chosen. I have used a uniform binning method in this work. The method divides the frequency domain into uniformly spaced bins and draws the maximum amplitude frequency from each bin. Whilst simple this doesn't consider any prior information about the nature of the frequencies that will be generated in a sample. As will be discussed in the limitations. There is still the question, what number of bins gives the best accuracy?

Optimisation of all the parameters is difficult as optimisation of one parameter will affect the optimum of another chosen parameter/ method. The optimum also must be calculated with varying noise levels. I look at optimisation of the free parameters with time segment length and sample duration

Background Noise

The background noise in this problem is not necessarily random noise such as white or brown noise. The type of noise that is most likely to occur in the background of a sample would be people talking, car noises, rain, etc. and is the noise that will most affect the algorithm. This would be categorised as unstructured, pitched noise. To generate this digitally, I obtained background noise waveforms from YouTube and added the waveforms to the samples of the song with varying signal to noise ratios in order to calculate the accuracy.

The samples of background noise can be found in the link in the appendix, consisting of a motorbike sound effect, party crowd background noise, people talking, rain, small crowd talking and a football crowd.

Results

6 songs were chosen to test my algorithm, the six songs contained two pairs of songs that were the same but were from different artists. The genres were jazz and jazz rap as that is the music I know and like. The music used can be found in the link in the appendix.

In each results case the exact parameters that were used are stated for reproducibility as there were no fixed parameters from the start.

To start, I chose to use 20 bins with uniform bin widths of frequency to choose the maximum from and starting values for the time slices and duration of a sample of 0.085s and 20s respectively. The bins of frequency were only applied up to 4700Hz as frequencies above this are above the highest note that can be played on a full-size piano and are unlikely to be in a piece of music. The accuracy of the program was tested over signal to noise ratios ranging from $10^{-9} \rightarrow 10$ containing 50 data points drawn from a logarithmic scale between $10^{-9} \rightarrow 10$. The accuracy was tested over 996 samples of songs with a crowd of people talking as the background noise. 996 samples were chosen as then an equal number of samples of each song in the database can be drawn. This gave a baseline to the accuracy of iteration 3 of the algorithm.

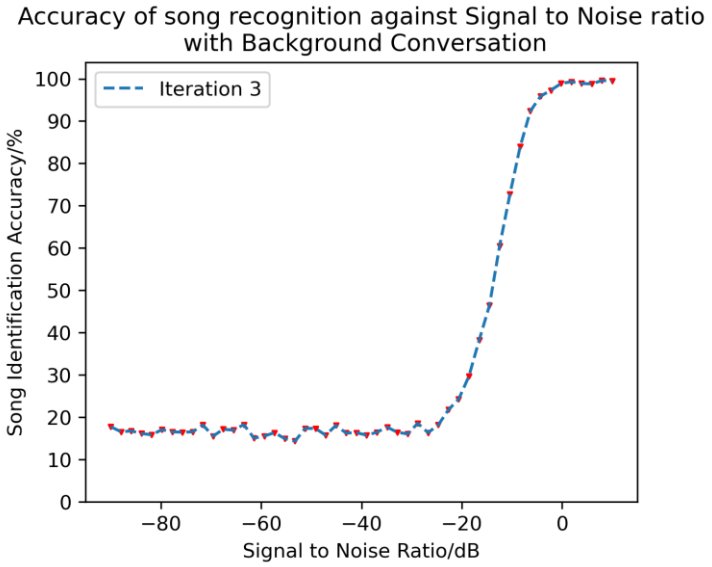


Figure 5 Plot of song identification accuracy obtained against the signal to noise ratio of the sample using iteration 3 of the algorithm. The accuracy values were obtained from 996 samples of songs from the database with a crowd talking as the background noise. We can see that the algorithm is very accurate when the signal to noise ratio is around 0 decibels. The accuracy drops when the noise is 100 times the power of the signal.

The accuracy evens out below -40dB signal to noise ratio so it is not worth looking at those ranges. The song identification accuracy levels out at $\approx 20\%$ this is because there are 6 songs in the database and therefore if songs were chosen randomly,

the correct song would be chosen $\approx 17\%$ of the time. Therefore, the levelling out is essentially the algorithm picking a song at random from the database.

The baseline accuracy test prompted a possible problem in computing the accuracy. Music is digitised mainly in 16-bit integers which can take signed values up to 32767. This means that adding noise digitally at varying levels can induce overflow of the max value that each datapoint can hold and would mean that waveform information is lost, and the accuracy of the algorithm cannot be determined. To counteract this the datatype was scaled to a 64-bit integer and the amplitude of the sample of a song was scaled by a factor of 1/5 to reduce the possibility of overflow. A 64bit integer is sufficient as this can hold a value up to $9 \cdot 10^{18}$.

Iteration 3.1 was then compared to iteration 3 for accuracy. The background noise was background conversation. The free parameters used in the algorithm were the same as for Figure 5. The accuracies for iteration 3.1 were only calculated down to a signal-to-noise ratio of -40dB as Figure 5 showed that the algorithm became essentially random below this noise level.

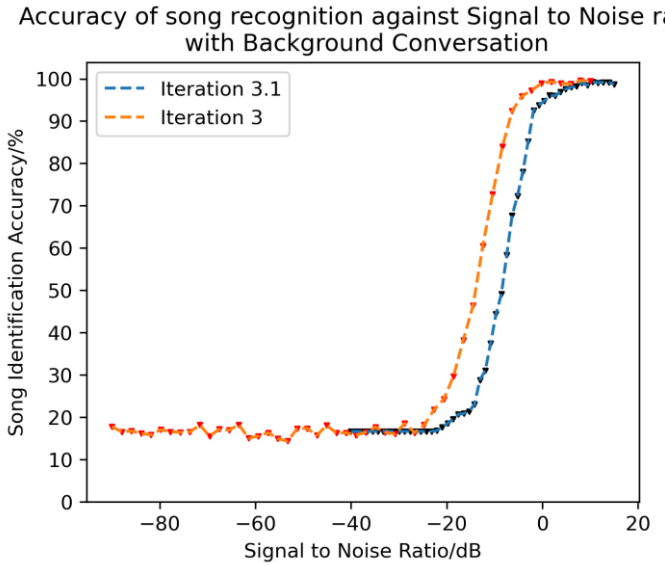


Figure 6 Plot of song identification accuracy for iteration 3 and 3.1 of the algorithm against the signal to noise ratio of the samples. Whilst slower to compute, iteration 3 is much more accurate than iteration 3.1. The accuracies were calculated from 996 samples of songs. Iteration 3 filtered for summed frequency differences below two standard deviations from the average summed frequency difference.

The accuracy of iteration 3.1 was below that of iteration 3 however, a free parameter has been introduced in iteration 3.1 for the cut-off of the summed magnitude frequency difference across a time segment. Currently, values that lay below 2 standard deviations below the mean were chosen for time correlation. This removes $\approx 95\%$ of the data points and a matching frequency in a song and a sample will sit below this value. This could be optimised which could improve the accuracy of iteration 3.1.

The speed of computation is much faster using iteration 3.1, therefore for the process of parameter optimisation iteration 3.1 will be used. Obtaining better accuracies for iteration 3.1 will also give better accuracies when using iteration 3.

Using a signal to noise ratio of 1 for initial optimisations and a background noise of background conversation. I tested time slices of lengths between $0.05 \rightarrow 1$ seconds, and durations ranging from $5 \rightarrow 30s$. The number of samples is reduced to 96 rather than 996 as optimisation of duration of a sample and length of a time slice requires many more computations. This is an initial search over the space of time segment lengths and sample durations. When an optimum is found then the accuracy can be increased, and the search space size decreased.

The colours and the colour bar in each of the subsequent plots are in reference to the accuracy value. Lighter colours represent higher accuracy as highlighted by the colour bar.

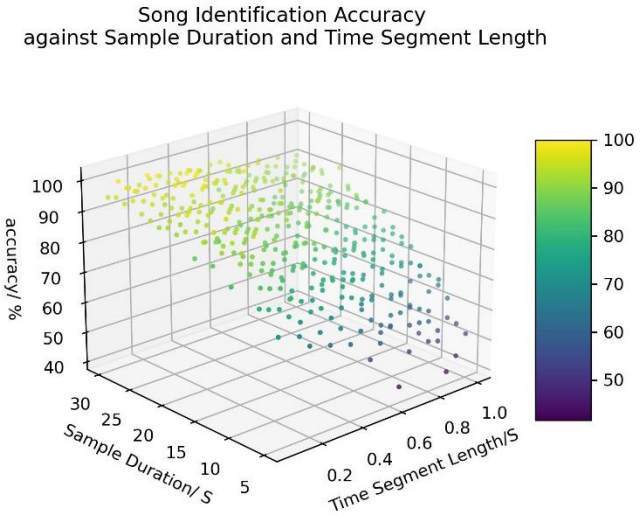


Figure 7 A plot of the song identification accuracy against time segment duration and sample duration using iteration 3.1 of the algorithm. The accuracies are determined from 96 samples of data. The plot shows, as expected, the accuracy decreases as duration decreases and time segment length increases. However, with smaller time segment length the accuracy can be maintained for lower sample durations.

Figure 7 shows that whilst an increase in duration and decrease in time slice length increases the accuracy, a lower time slice length can allow for a lower sample duration. The sample duration needs to be at a minimum for identifying music in real time.

The search space was then reduced to between $0.01 \rightarrow 0.1s$ for time segment length, $5 \rightarrow 10s$ for sample duration and a signal-to-noise ratio of 1 was maintained. The samples were increased to 300 tested samples for comparison of accuracy.

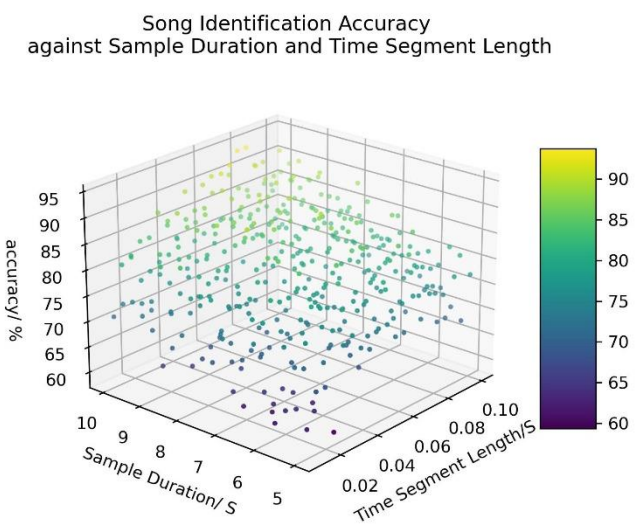


Figure 8 A plot of accuracy against Sample duration and time segment length. The accuracies have been computed over 96 samples using iteration 3.1 of the algorithm. The plot shows that there is a lower limit to the time slices that will give better accuracies.

From Figure 8 we can see the boundaries on the lower limit of the length of a time segment. The optimum point looks to occur between 0.06s and 0.1 seconds for the duration of a time slice with a 10 second duration.

The final search of the space looked at time segments of lengths between $0.08 \rightarrow 0.2s$ using a linear spacing of 20 points. The sample duration was between $10 \rightarrow 20s$. 300 samples were tested for each accuracy.

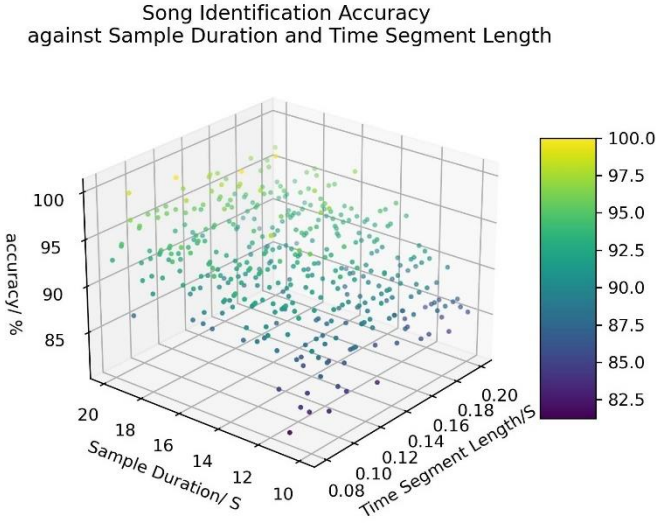


Figure 9 plot of the song identification accuracy against the time segment length and the sample in a reduced range using iteration 3.1 of the algorithm. the accuracies are determined across 300 samples. The plot shows that the accuracy drops off significantly below 12 seconds of song duration.

When running iteration 3.1 with a signal to noise ratio of 1 and with the original choice of slicing and duration, 0.085s and 20 seconds respectively, the accuracy obtained after 1000 samples was 94.5%. Figure 9, shows that a similar accuracy can be achieved with a reduced sample duration and changing of the time segment length. The optimisation space has been reduced to a range of duration from $12 \rightarrow 20s$ with time segment lengths of between $0.1 \rightarrow 0.2s$.

To explore the effect of the number of bins, I chose to use iteration 3.1 again for speed of computation. The accuracies are calculated for bin numbers between 4 and 70 bins in steps of 2 bins, and signal-to-noise ratios from -60dB to 15dB. The accuracies are calculated from 300 tested samples.

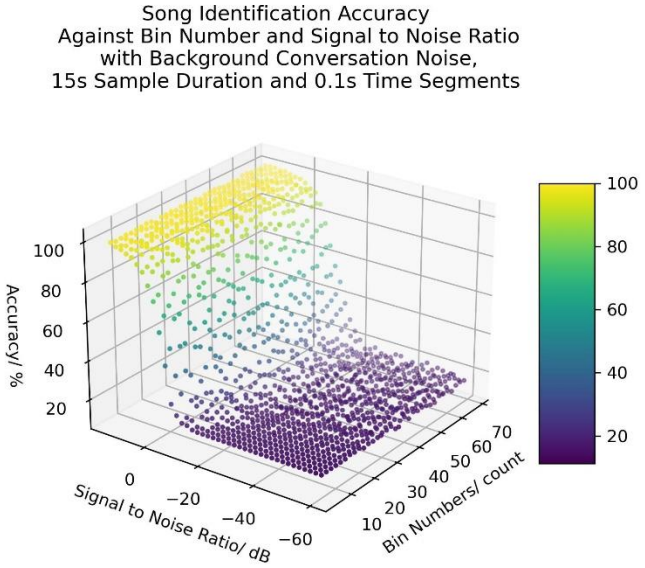


Figure 10 A plot of song identification accuracy against number of bins used to divide the Fourier plane. The accuracy values are drawn from 300 tested samples, the duration of a sample was 15 seconds, and the length of a time segment was 0.1s. The accuracy does improve with bin number however the improvement in accuracy appears to increase asymptotically with bin number.

It's difficult to see the increase in accuracy when the bin number increases in Figure 10. To highlight the accuracy gain I looked at the 3D space with a top-down view.

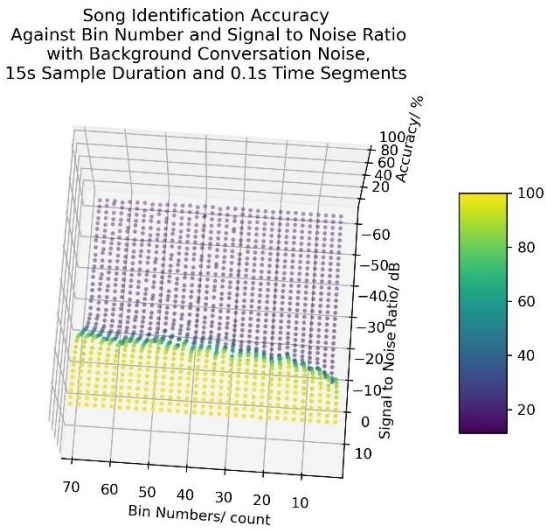


Figure 11 A different view of Figure 10 showing song identification accuracy against bin number and signal to noise ratio. The accuracy values are drawn from 300 tested samples. The duration of a sample was 15 seconds, and the length of a time segment was 0.1s. The asymptotic relation of bin number to accuracy is highlighted.

I replotted Figure 10 with logarithmic bin numbers to highlight the asymptotic relationship between accuracy and bin number. This highlights the asymptotic relationship between the bin number and the accuracy increase.

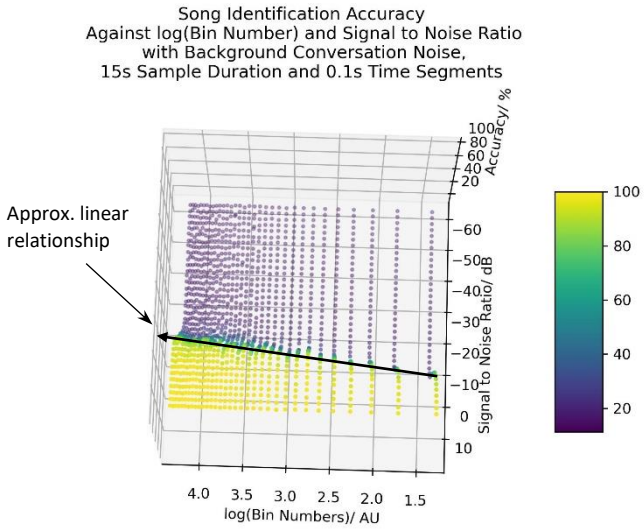


Figure 12 A replot of Figure 10 with the bin numbers logged showing a linear relationship between the accuracy increase with logarithmic increase in bin number. The black arrow shows the linear relationship between log(bin number) and the increase in the accuracy of the algorithm.

With increasing bin number, the accuracy does improve. The signal-to-noise ratio at which the accuracy of the algorithm drops, decreases. Due to the sample rate of the music being $48kHz$, the bins can only be increased until the bin width is still above at least twice the frequency spacing of the discrete Fourier transform otherwise the algorithm will not be able to discern any maxima as every discrete frequency in the Fourier plane will be in its own bin. As stated before, the maximal frequencies are only evaluated up to $4.7kHz$ as the highest note on a full size, 88-key piano is roughly $4186Hz$.

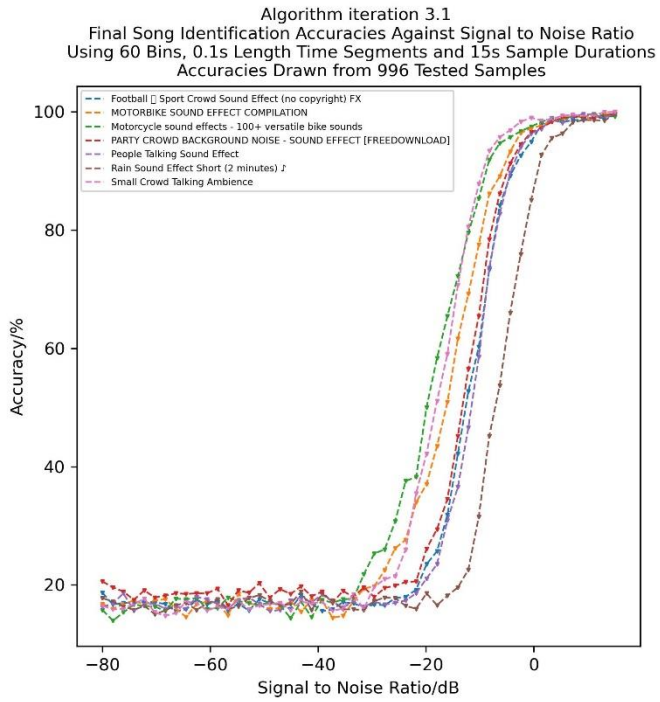


Figure 13 The final plot of accuracies using time slices and bins in the range of where the optimised parameters seemed to lie. There were 60 bins, 15s duration samples and 0.1s time slices, the accuracy was tested using 996 samples for each accuracy value. There is a wide variety of accuracies for varying noise types. This was using iteration 3.1 of the algorithm.

Figure 13 Shows the final plots with the more optimised free parameters. The accuracy varies with different background noise type and is worst when the background noise is rain and thunder. The accuracy has been improved such that the accuracy drops off, below 95%, at around -0.5dB for background conversation comparatively to before Figure 6. The accuracies drop off between -6 and 3dB depending on the background noise. The algorithm was most accurate for the noise file Small Crowd Talking Ambience, with a drop below 95% accuracy occurring at $\approx -6dB$. The algorithm performed the worst on the rain sound effect, a drop off to 95% occurring at $\approx 3dB$.

I could not run accuracy measurements with iteration 3 and the optimised parameters as the computation time exceeded the time that I had available. The algorithm works on single samples but running many samples is not feasible.

Discussion

The results are good, without optimisation the two algorithms performed well as shown in Figure 6. The accuracy of iteration 3 of the algorithm dropped to 95% at around a signal-to-noise ratio of -4.0dB and the accuracy of iteration 3.1 dropped to 95% at a signal-to-noise ratio of 1.5dB.

The reason that iteration 3.1 was maintained as a viable algorithm initially was the efficiency of the algorithm. Whilst less accurate the algorithm is much faster at comparing samples to songs, with optimisation of the free parameters the accuracy could be improved to be equal to iteration 3. For parameter space optimisation, it was only feasible to use iteration 3.1 of the algorithm due to the many computations required.

Iteration 3.1 also introduced another free parameter, the filtering process of the sum of the magnitude frequency difference of each time segment. Accurate results were obtained when only maintaining points that were more than two standard deviations below the mean, however this was not optimised in the scope of this paper.

The exploration of optimising the free parameters of the algorithm showed that all the free parameters are highly correlated. A proper treatment of the optimisation of these parameters would be to calculate them together. This creates a 3D optimisation space when comparing bin number, time segment length and duration of a sample. Further, including the free parameter in the filtering process of iteration 3.1 and the effect on accuracy for varying signal-to-noise ratios, the optimisation space becomes 5 dimensional. The treatment of the free parameter space was beyond what could be done in the time frame of this project. The parameter space was instead computed in slices from the 5D parameter space.

The optimisation of time segment length and sample duration were computed together. The aim was to minimise the sample duration and maintain accuracy. From Figure 7 we can see that to minimise the duration of a sample the length of a time segment must be decreased to maintain accuracy.

From Figure 8 we can see that there is a lower limit to the time segment length where the accuracy starts to drop off irrespective of the sample duration, around a time segment length of 0.06s. This is due to the reduction in the resolution of frequencies that can be discerned at these time lengths. The figure also shows that the accuracy drops rapidly when the duration of a sample drops below 8 seconds. Figure 9 Shows that when the time segment length is optimised the duration of a sample can be decreased showing accuracies of above 95% down to 12 second sample durations.

The optimisation of the bin number showed, as expected, a higher number of bins led to better accuracies overall. The relationship is asymptotic meaning there is an upper limit to the bins that gives good accuracy without bloating the algorithm, appearing to be around 60 bins. As shown in Equation 4 the frequency spacing in the Fourier plane is dependent on the length of a time segment. This means that the upper limit to the number of bins is also dependent on the length of a time segment. Therefore, the number of bins and length of a time segment are correlated, and optimisation should be calculated including both parameters. Given more time the 2D optimisation space of bin number and time segment length could be explored.

The parameter optimisation space is treated simplistically in this work. I have explored some of the space however I have not looked at it in detail. As the space is 5D it is a complex space to explore and could be simplified using multivariate interpolation. The reduced dimension parameter spaces dealt with here, time segment and duration of a sample and bin number with noise level could be fit as a surface. The free parameters of these surfaces could be found and the formula. The combination of these surfaces could then be used to interpolate the 5D space of the free parameters that exist in this work. Given more time, I could have developed an interpolation algorithm to span the 5D free parameter space. The 5D parameter space could be built from the surfaces generated in 3D.

The final plots showed good accuracies even with a high presence of noise. The worst case was a drop in accuracy at $\approx 3dB$ with the rain sound effect as background noise. This shows a good proof of concept for the algorithm. The accuracies obtained for some types of noise were even better than the accuracies generated by the original shazam algorithm (Wang, 2003).

The algorithm has been developed for song identification however I believe it could have uses beyond music identification. Two possible wider applications of the algorithm would be for voice recognition or identification of astronomical objects from satellite data.

For voice recognition, each human has a specific accent and tone when speaking. By creating a database of sound fingerprints, such as vowel sounds, for a person the algorithm could identify the person speaking in a room of conversation from the generated fingerprint of their vowel sounds.

For the identification of astronomical objects, such as pulsars. If a small sample of data that is known to contain a pulsar, for example, is put through the algorithm and a fingerprint generated. Then, given an arbitrary set of data from a satellite, after an astronomical survey, the algorithm would be able to chunk through the data and search for points in the data where a pulsar could exist. This algorithm could allow an automated form of characterisation of significant data within general data generated from astronomical surveys.

Algorithm Limitations

The algorithm works well as a proof of concept; however, implementation of the algorithm physically has more factors to consider.

Firstly, this algorithm would most likely be used on a mobile device. A person would want to be able to identify music wherever they are. This poses a problem as the accuracy of the program will not be dependent on just the signal to noise ratio but also on the quality of the microphone. If a phone microphone cannot pick up the whole range of the data, then the algorithm will be working with corrupted data which could affect the accuracy.

Secondly, the algorithm has been developed using six songs in the database. If the algorithm were to be used for proper song identification, then a database of millions of songs would be necessary for a general-use song identifier. The number of data points that are generated by the algorithm and the way that it deals with searching the song database is inefficient for large databases. This poses a problem in bringing the proof of concept into a real-life application.

The algorithm uses a uniform binning method for finding the maximum amplitude frequencies in the Fourier plane. However, the most used method to tune an instrument is based on a method called equal temperament. A reference frequency is chosen for a note on the musical scale, normally A4 which is tuned to 440Hz., the other note frequencies are then calculated from this reference frequency using the equal temperament formula

$$f_n = f_0 \cdot 2^{\frac{n}{12}}$$

Equation 5 The equal temperament formula, where f_n is the frequency of a note n semitone steps from the fundamental frequency, f_0 represents the fundamental frequency, usually 440Hz, and n is the number of semitone steps from the fundamental frequency, the division of n by 12 is because there are 12 semitones in an octave.

The frequencies of notes, as shown from the formula, is nonlinear. This poses a possible problem with the uniform binning method. For low frequencies, a bin may span a couple of octaves whilst in the higher frequencies a bin may span only a couple of notes. This could mean that the fingerprint loses information of notes and chords in a song.

A possible improvement to the bins could be to make each bin span the equivalent of an octave i.e., the width of a bin doubles each time. This method uses prior information about the nature of the sound to make a possibly more accurate algorithm that would be less susceptible to background noise. This could be extended to make a bin span the equivalent of some factor of an octave and have a geometrically growing bin size with frequency. If I had another month, I would have explored the possibility of using nonuniform binning methods.

Conclusion

This paper aimed to create a proof-of-concept song identification algorithm. I have managed to create two variations of a song identification algorithm, one based on speed and one based on accuracy. The first variation can accurately identify songs 95% of the time down to a signal-to-noise ratio of -4dB, the second can accurately identify songs down to 1.5dB using 20 bins, sample durations of 20 seconds and time segment lengths of 0.085 seconds.

I have run parameter optimisations on the free parameters, including the duration of a sample, the length of a time segment that samples and songs are divided into, and the number of bins that are used to draw the maximum amplitude frequencies in the Fourier plane. The exploration of the parameter space showed a high correlation between all the parameters which would indicate the necessity to optimise the parameters together. This was too difficult given the time frame so the correlation between only some of the parameters has been explored.

The optimisation of time segment length and sample duration showed that the best accuracies were obtained between 0.08 to 0.2 seconds for the segment length with sample durations ranging between 12 to 20 seconds. The aim of optimising these parameters was to minimise the duration of a sample and maximise the length of a time slice. Higher durations could be used than quoted here however the gain in accuracy was shown to be asymptotic, Figure 9. For this algorithm's purpose as a viable product the sample duration would need to be minimised.

The optimisation of bin number showed that with an increase in the number of bins the accuracy increases asymptotically with bin number. The bins are also correlated to the time segment length as the frequency spacing in the Fourier plane is inversely proportional to time segment length, as shown in Equation 4.

Final accuracy measurements across all the different noise samples showed that the algorithm was sensitive to the type of noise, Figure 13. The algorithm performed worst on the rain sound effect and best on the people talking sound effect. The algorithm was still accurate despite the sensitivity to different types of background noise.

Overall, I have managed to create a working proof of concept of a song identification algorithm. An exploration of the optimisation of the algorithm's free parameters showed gave good ideas about the effect of the free parameters on the accuracy of the algorithm.

Future work on this algorithm would look at using non uniform bin widths with a geometric increase in bin size to match the frequency spacing of octaves used in music. I would look at parameter optimisation using multivariate interpolation using the surfaces generated in 3D to generate values spanning the 5-dimensional free parameter space. I would look at the possibility of being able to remove noise from the signal based on further demodulation techniques.

References

Andersen, B. C., & Ransom, S. M. (2018). A Fourier Domain "Jerk" Search for Binary Pulsars. *The Astrophysical Journal Letters, 863*, L13.

Cantor, D., & Kaltofen, E. (1991). On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica, 28*, 693–701.

Emmanuel, G., & al, e. (2021). Information hiding in images using Discrete Cosine. *IOP Conference Series: Materials Science and Engineering*, 1098.

Kenna-Allison, M. (2021-2022). *Math2015.*

Li, Y., Wang, T., Sun, B., & al., e. (2022). Detecting the lead–lag effect in stock markets: definition, patterns, and investment strategies. *Financial Innovation, 8*, 51.

Pandley, S., Singh, M. P., & Pandey, V. (2015). Image Transformation and Compression using Fourier Transformation. *International Journal of Current Engineering and Technology, 5*, 1178-1182.

Tewari, P. (2017). Audio Compression Using Fourier Transform. *International Journal of Science and Research, 6*.

Turner, R. E. (2010). *Statistical Models for Natural Sounds.* Obtenido de University of Cambridge Computational Perception Group: http://cbl.eng.cam.ac.uk/Public/Turner/Publications

Tyson, R. K. (2014). *Principles and Applications of Fourier Optics.* Bristol: IOP Publishing.

Wang, A. (2003). An Industrial Strength Audio Search Algorithm. *ISMIR 2003, 4th International Conference on Music Information Retrieval.* Baltimore: ISMIR. Obtenido de https://www.researchgate.net/publication/220723446_An_Industrial_Strength_Audio_Search_Algorithm

Zhao-yun, Z. (2011). Study on time-varying velocity measurement with self-mixing laser diode based on Discrete Chirp-Fourier Transform. *Journal of Physics: Conference Series, 276*.

Appendix

The sample music and noise can be found at this link: https://sotonac-my.sharepoint.com/:f:/g/personal/tdhm1g20_soton_ac_uk/EqCR3DFVm01HpbqqJqxECu0BOFu-PX4LyS62b65VwcakEQ

Available to only people within the University of Southampton.

The Fourier transform has many uses beyond song identification. In industry the Fourier transform can be used for image compression as shown in (Pandley, Singh, & Pandey, 2015). Where the fast Fourier transform is used to reduce the data required to store an image by generating a 2D Fourier transform of an image. The data stored is then simply coefficients and frequencies of sines and cosines that generate the original image.

Time correlation techniques are also used more broadly in industry as shown by (Li, Wang, Sun, & al., 2022) where the idea of time lag within the stock market is explored in detail. Time lag in the stock market is when a securities price will affect another securities price with some time delay. This can be measured on the lower time frame ticks and is used mainly in high frequency trading.

The Fourier transform is also used in steganography as shown in Emmanuel, et. al. (2021), where the discrete cosine transform, a variation of the discrete Fourier transform using only cosine waves, is used to hide sensitive data within an image such that the sensitive data is obscured but also the data transmitted does not look suspicious to an external viewer.

Discrete Fourier transforms are used in Laser Diode self-mixing interferometry (Zhao-yun, 2011). Self-mixing interferometry can be used for velocity measurements. When part of the laser beam is reflected back inside the cavity the interference causes the laser beam profile to modulate. This modulation can be measured by measuring the intensity of the laser beam over time and using the discrete fourier transform to extract the frequencies. A peak in the frequencies is at the doppler frequency of the moving object.

The Fourier transform can also be used for multiplication of large integers and polynomials as shown in Cantor & Kaltofen (1991). The multiplication of two polynomials can be written as a convolution of the two polynomials. Convolution becomes a simple multiplication when the functions are Fourier transformed. This means that coefficients of the polynomial generated from the product of two polynomials can be generated from the coefficients generated by taking the inverse fourier transform of the product of the Fourier transforms of the two polynomials. This can be extended to multiplication of large integers by representing the large integers as a polynomial in a specific base, e.g. $a_0x^0 + a_1x^1 + a_2x^2 + \cdots a_nx^n \; where \; x \; represents \; the \; choice \; of \; base, n = 0,1,2,3, \ldots$. The complexity of this operation is $O(nlog(n))$ whereas for simple multiplication the complexity is $O(n^2)$.

Fourier transforms are also used in the analysis of pulsars (Andersen & Ransom, 2018). Andersen & Ransom discuss the use of the fast fourier transform on time series of an isolated pulsar. The power series will have sharply defined peaks at harmonics of the spin frequency. When the pulsars are in some form of orbit the doppler shift causes a change in the freqeuncies which ‘smears’ the power spectra. In the paper they discuss the application of a Fourier domain ‘jerk’ acceleration search to account for the orbit of the pulsar. They assume that the acceleration of the pulsar is constant across the observation time and this allows to generate the change in frequency across the observation and account for this in the Fourier domain search for the pulsar. They found that with the new search algorithm the sensitivity to pulsar binaries was increased when the observation measurement is 0.05-0.15 the orbital period.