A REVIEW OF AUTOMATIC MUSIC TRANSCRIPTION LOW LEVEL PROCESSING TECHNIQUES AND THE EVALUATION AND OPTIMISATION OF MULTIRESOLUTION FFT PARAMETERS

BENJAMIN PETER EVANS

A thesis submitted to the University of Huddersfield in fulfillment of the requirements for the degree of Master of Science by Research

The University of Huddersfield

January 2012

Copyright statement

Abstract

The Fast Fourier Transform (FFT) is commonly used in the field of digital signal processing to move a signal from the time domain to the frequency domain. The FFT is popular as a low level processing technique in automatic music transcription algorithms, but there is a performance trade-off between suitable time and frequency resolutions for music transcription. To address this problem, multiresolution methods that employ several FFTs across the frequency spectrum have become popular. The purpose of this investigation was to assess the properties of the FFT in the context of Automatic Music Transcription (AMT) and to optimise the main parameters of a multiresolution FFT to improve the spectral output.

Background theory of AMT and current low level processing techniques is presented. Discussion of the FFT decomposition theory and multiresolution techniques are followed by a brief overview of spectral processing and current high level processing approaches. These topics are presented within the context of western music harmony as a foundation for the presentation of an optimised multiresolution FFT.

A novel method of scoring FFT parameters based upon frequency resolution, time resolution and the alignment of the fundamental frequencies for equal tempered musical notes with the frequency bins of the FFT was developed. A 4-band multiresolution FFT with optimised sub-band divisions and FFT lengths is derived from the exhaustive evaluation of parameters based upon the scoring method.

The optimised 4-band multiresolution FFT is evaluated against a single band FFT, a 3-band optimised solution, an existing 4-band multiresolution FFT solution and two variations of the existing 4-band multiresolution solution - comparing optimisation scores and performance in sinusoidal extraction tasks.

Theoretical results show the optimised 4-band multiresolution FFT does offer an improved performance for use in automatic music transcription compared to a non-optimised solution. Preliminary real world testing indicated issues that require further investigation.

Thesis Body Word Count – 27,842

List of Figures

List of Tables

Acknowledgements.

Firstly, I want to thank my wife Hannah for all her love, support, encouragement, cups of tea and the endless supply of biscuits. You've kept me going!

I would like to thank Dr. Jonathan Wakefield for supervising this project from beginning to end, and for all the direction, advice and time you have given to me along the way that has helped shape this thesis.

Where would I be without my parents? So, thank you Mum and Dad for being mum and dad.

Thank you to David Bray, my line manager for affording me the time to pursue this project.

I am also grateful for the help provided by Joel, Andy, Ben, Dave, Toby, Steve, Mark, Tom and Fred over the years that ensured the good ship kept sailing while I had my head in books. Thank you.

Introduction

Mozart would have attended many concerts and performances during his lifetime, but famously he attended a performance of Gregorio Allegri's "Miserere" in the Sistine Chapel in 1770. It is famous because upon leaving the performance he proceeded to write the entire score for the piece of music he had just heard, from memory. He then attended a second performance just to check he had scored it correctly (Gutman, 1999). This ability to listen to music and decode it is possible for all humans with a functioning auditory system in some measure, even if it is just identifying one sound as being higher than another, or it being different in tone. Although Mozart was highly trained and exceptionally good at transcribing music, that is analyzing an acoustic signal and writing down the pitch, onset time, duration and source of each sound that occurs in it, today's computers still struggle to succeed at even simple transcription tasks.

Some of the earliest handwritten scores are found in the early church where monks would transcribe chants from the performance they heard to written notation therefore allowing others to reproduce the original music having never previously heard it. Later, the invention of the printing press greatly increased the distribution of written music and so it developed over time to the standards found in the musical scores of today (Latham, 2002, pp. 842-849). Despite the great advancement in technology for printing and distributing written music, the human ability to detect and decode sounds into their basic characteristics of time duration and pitch is something which technologists are still striving to replicate.

The dawn of the computer age gave great pace to signal analysis, particularly when J.W Cooley and J.W. Tukey published their paper in 1965 utilising a computer to calculate the Discrete Fourier Transform (Cooley & Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, 1965). Interest specifically in music analysis and automatic music transcription increased from the 1980s, when advancements in processing power first allowed computers to feasibly model the human auditory system and analyze audio quickly and efficiently – allowing many processes to be performed simultaneously (Patterson & Moore, 1986). Now the area of automatic music transcription research is very active covering many aspects of music transcription and analysis.

The annual ISMIR conference is a bench mark for progress in the research community and the evidence of the papers submitted suggest that the initial processing stage in music analysis is moving an audio signal from the time domain to the frequency domain. There are varying methods to transform a signal from the time domain to the frequency domain, such as filter banks (Diniz F. , Kothe, Netto, & Biscainho, 2007) and wavelets (Azizi, Faez, Delui, & Rahati, 2009) but still the Fourier family of transforms is used widely as a low level frequency analysis process for music transcription (Tan, Zhu, & Chaisorn, 2010) (Hsu & Jang, 2010). So, despite it's age, the Fourier transform is still important in signal processing, and often is the foundation of other transform techniques such as Fast Filter Banks (Diniz F. , Kothe, Netto, & Biscainho, 2007) and the multiresolution FFT (Dressler, 2006).

The development of multiresolution processes such as wavelets and the multiresolution FFT are of significance. All windowed spectrum analyses, including the Short Time Fourier Transform suffer from a compromise between time resolution and frequency resolution This is related to the Heisenberg Uncertainty Principle (Roads, 1996) that states if an accurate measurement of a signal's timing is required, the accuracy of the signal's frequency measurement will be compromised. Conversely, if an accurate measurement of a sound's frequency is required, the accuracy of the sounds timing measurement will be compromised. Multiresolution analyses are designed to address the Heisenberg Uncertainty Principle and improve the time and frequency resolution simultaneously.

The first chapter of this thesis introduces the fundamentals of sound and music as well as how the human detects and perceives sound and pitch. The different disciplines within automatic music transcriptions are also discussed, as well as the problems associated with fundamental frequency tracking.

Chapter 2 presents current single resolution low level processing techniques for automatic music transcription. The main content of this chapter is a discussion of the Discrete Fourier Transform and the Fast Fourier Transform implementation.

Chapter 3 continues the discussion of low level processing, focusing on multiresolution techniques. The techniques presented are divided into two categories, those imitating the human auditory system and those based upon a 'constant Q' factor to vary time and frequency resolution across the spectrum.

Chapter 4 is a brief introduction to 'Peak Picking' methods for extracting frequency maxima representing note frequencies from a spectral representation. The frequencies selected by the peak picker are the 'note candidates' presented to the ‘high level’ processing. This chapter also discusses methods to manipulate the spectral information to improve the performance of the peak picker.

Chapter 5 provides an overview of current popular high level processing for automatic music transcription. The purpose of this chapter is put into context the low level processing discussed in previous chapters.

Chapter 6 is a discussion of the FFT parameters and characteristics of the FFT algorithm for use in automatic music transcription.

Chapter 7 proposes a novel method of choosing parameters for multiresolution Fast Fourier Transforms to optimise the output to create higher quality note candidates for higher-level automatic music transcription processing. The optimised multiresolution FFT is tested and compared to other implementations. Results are presented and discussed.

1 Fundamentals of Sound, Hearing, Music and Transcription

Before discussing pitch detection, identification methods and digital processing techniques, it is important that the fundamentals of sound and its properties are established. This section will introduce the fundamental properties of sound waves, to help understand what is being attempted in this work and the associated problems.

The basic properties of wavelength, frequency, loudness, amplitude and phase are defined in appendix 1.

1.1 Superposition of Waves – Constructive and Destructive Interference

Two waves traveling in opposite directions can pass through each other and emerge with their original form. This behavior is described by the principle of superposition (Rossing, Moore, & Wheeler, 2002, p. 44). Figure 1-1 shows 2 sine wave pulses passing through each other. At the point they meet their amplitudes are summed resulting in a single summed pulse. However, note that the frequency component stays consistent for each of the pulses. Once the two waves have passed, they maintain their original characteristics.

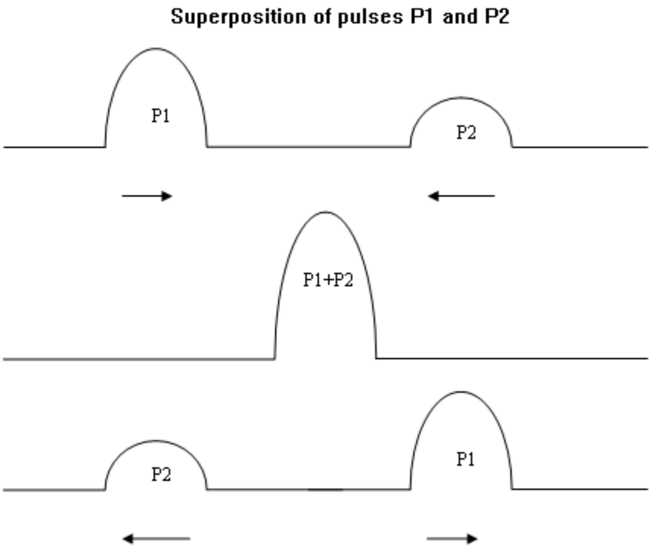


Figure 1-1 Superposition of waves

Image from (Stracha, 2008)

The same theory can be used for musical notes. A sine wave tuned to 440Hz played with a second sine wave tuned to 493Hz will create a wave consisting of the summation of a 440Hz and a 493Hz sine wave (Figure 1-2).

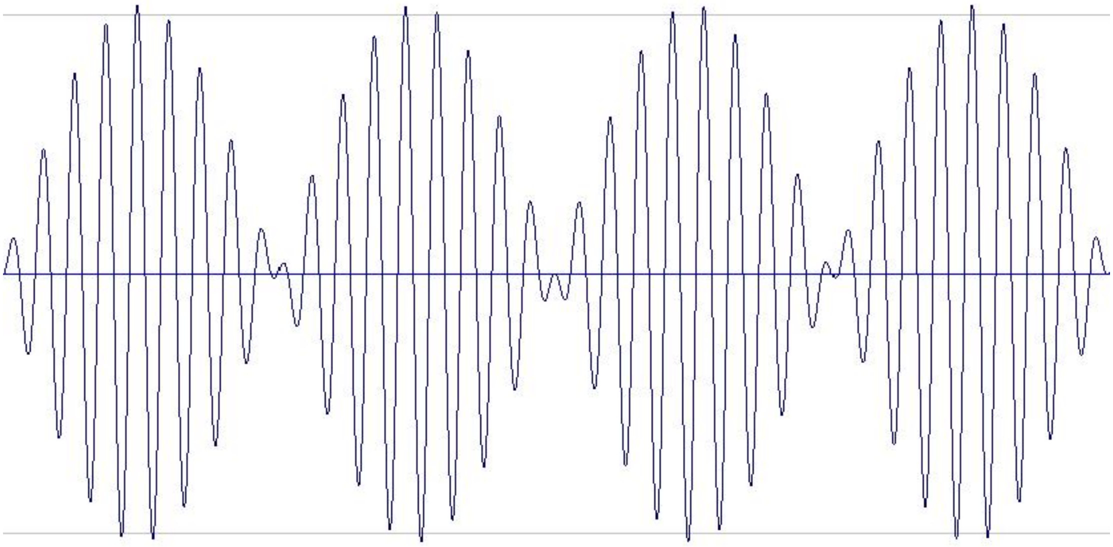


Figure 1-2 Wave summation

It is clear to see that the interaction of waves results in far more complex patterns than a simple sine wave. This is one of the difficulties associated with extracting pitch information from polyphonic waveforms as wave period information is harder to extract as waves interact and superimpose.

The amplitude, frequency and phase of individual waveforms contribute to the characteristics of the waveform produced when combined. The superposition of waves can result in cancellation. Figure 1-3 shows the complete cancellation of 2 waves with opposite phase resulting in silence – this is known as destructive phase interference.

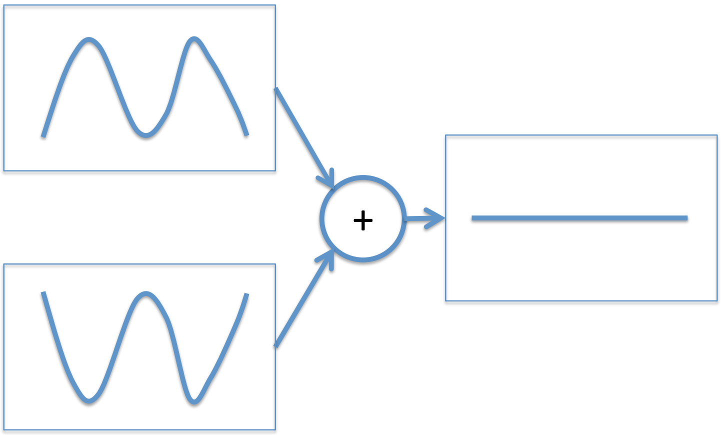


Figure 1-3 Destructive interference

If ‘in-phase’ waves are combined constructive interference occurs producing a wave with increased amplitude equal to the sum of the combined amplitudes (Figure 1-4).

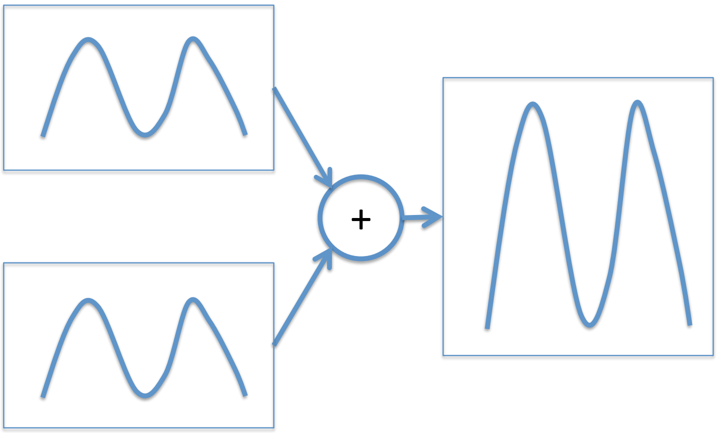


Figure 1-4 Constructive interference

The above examples are very simple, but serve to demonstrate that although a frequency is produced by a sound source, overlapping waves can cause complete or partial cancellation. Waves are combined with waves from other sources, resulting in some frequencies not being present or detectable in a mixed waveform.

1.2 Real World Notes

So far sine waves have been used to explain sound phenomena however, in the real world, acoustic signals are rarely perfect sine waves. Many factors are present in real world instruments and environments that ‘colour’ or ‘shape’ the sound. The source of vibration e.g. a string, or a reed will vibrate with different properties, the shape and material of the instrument will alter the way the sound resonates. How the player plays the instrument will change the resultant sound wave. The room in which the instrument is played will alter the sound reaching your ears. These and variants result in much more complex waves than a sine wave. Figure 1-5 demonstrates this by showing the waveform for an acoustic guitar playing a note at 440Hz. The guitar was recorded using an AKG 414 microphone placed approximately 12 inches from the sound hole in an acoustically treated studio environment, which minimizes the quantity of reflections and reverberation detected by the microphone.

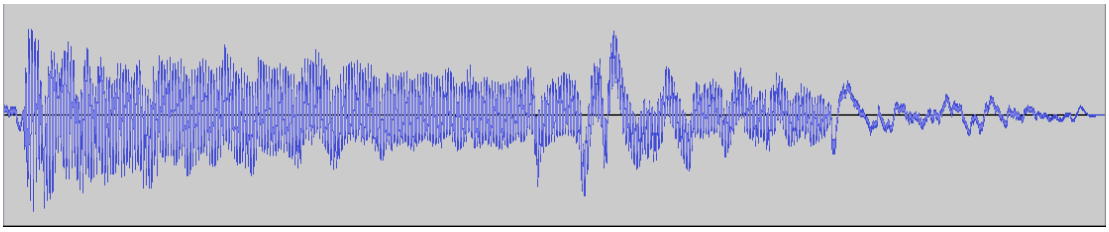


Figure 1-5 A 440Hz Guitar Note

Figure 1-5 demonstrates the complexity of ‘real world’ sound waves of instruments when compared to ‘ideal’ sine wave representations. The complexity of the waves is in part due to frequencies other than that of the pure note frequency (in this example 440Hz) being generated by the sound source and being received by the microphone.

1.3 Harmonics, Fundamental and Pitch

A frequency domain analysis of the 440Hz guitar note shown in Figure 1-5 reveals many more frequencies present in the signal than that of 440Hz [Figure 1-6].

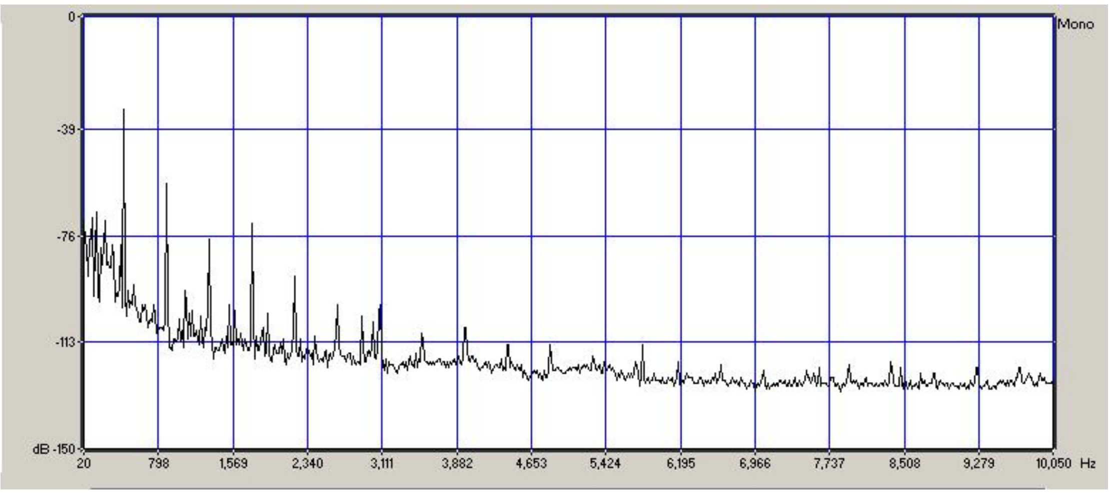


Figure 1-6 Guitar harmonics

Within the frequency make up of the guitar note are the harmonics of the fundamental frequency. Most oscillators such as a plucked string, human voice or trumpet naturally oscillate at not only one, but several frequencies. These are known as *partials*. When an oscillator vibrates with partials at integer multiples of the fundamental frequency, they are know as *harmonics*. Partials whose frequencies are not at integer multiples of the fundamental frequency are known as i*nharmonic.*

The fundamental frequency is the note frequency, in this example 440Hz and is typically the lowest frequency component, although not always as some instruments have sub harmonics. A harmonic is a frequency component that is an integer value of the fundamental, e.g. for a fundamental f, a series of harmonics could be 2f, 3f, 4f etc. For a note of 440Hz (fundamental) the first harmonic is at 880Hz, second harmonic at 1320Hz and so on. Figure 1-7 marks the fundamental frequency and the 4 harmonics of the guitar note of 440Hz.

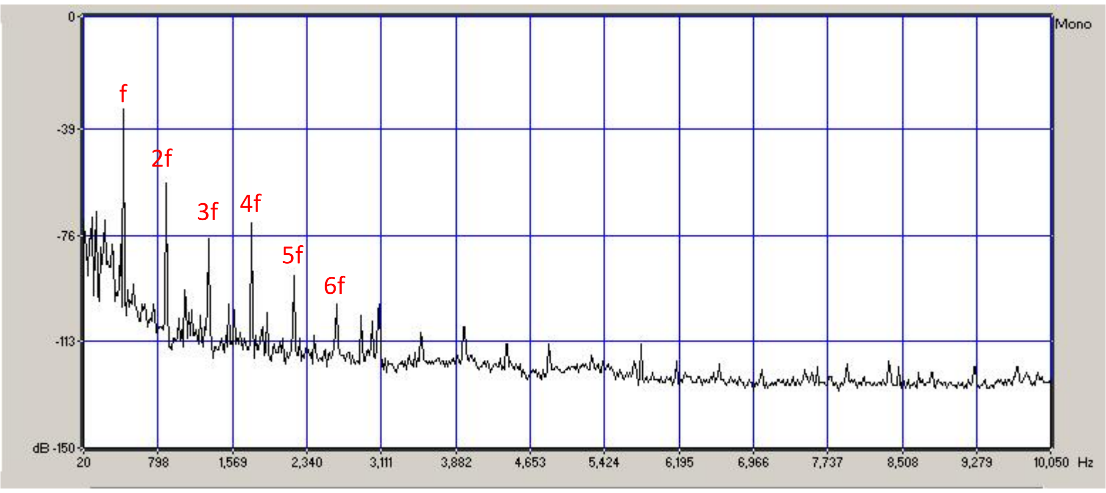


Figure 1-7 Numbered guitar harmonics

Although the untrained human ear does not detect harmonics as separate notes, harmonics do contribute to the perceived quality, timbre (the attribute used to discern two sounds as being dissimilar using criteria other than pitch, loudness or duration (Rossing, Moore, & Wheeler, 2002, p. 135)) and pleasantness of a sound, which are all influenced by the relative strength of the individual harmonic frequencies (Mesaros, Lupu, & Rusu, 2003).

The complexity of the sound wave, both in the time and frequency domains increases when multiple notes are sounded at the same time. Figure 1-8 shows an excerpt from the waveform of 2 notes played on an acoustic guitar recorded with an AKG 414 microphone. The notes are 440Hz and 493Hz.

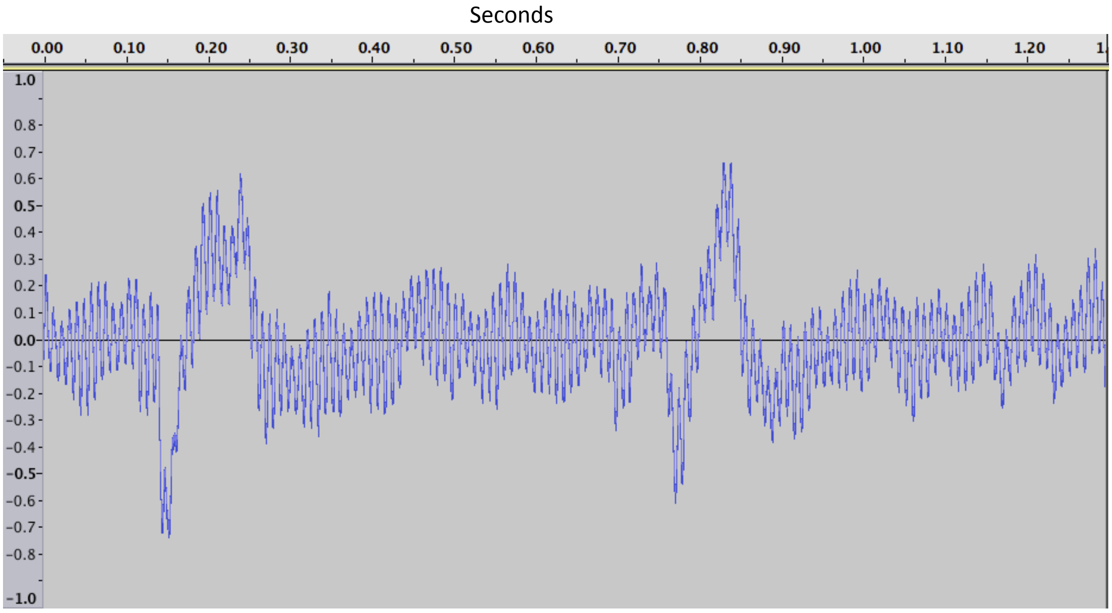


Figure 1-8 A section of a wave generated by 2 notes on an acoustic guitar

The resultant waveform of the 2 notes in unison is significantly more complex than that of a single note as the phases and amplitudes of the 2 notes and their harmonics interact with destructive and constructive interference. The frequency domain analysis shows the complexity of the interaction of the 2 notes and their harmonics (Figure 1-9).

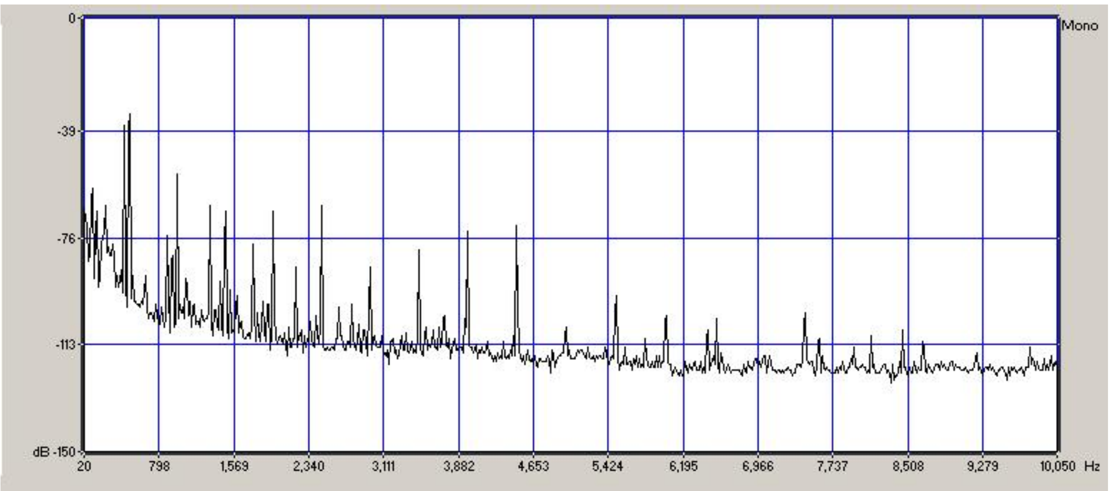


Figure 1-9 Interaction of harmonics

Interaction of the harmonics can make it difficult to discern the fundamental frequency of a note being played as it is masked by harmonics of other notes. However, as discussed later, the presence and pattern of the harmonics can be used to 'authenticate' the presence of a fundamental. In a similar way, the human auditory system uses the upper harmonics of a sound to determine the pitch of a note, and can even determine the fundamental frequency from the pattern of the harmonics even if the actual fundamental is not included in the wave (Rossing, Moore, & Wheeler, 2002, p. 126).

1.4 Human Auditory System

The human auditory system and the brain of a trained musician is the most reliable audio transcriptions system available (Klapuri, 2006b, p. 229). It has the ability to discern pitch, the timbre of a sound, separate sound sources and locate sound in an environment with great ease and accuracy that currently cannot be rivaled by computer technology. Therefore, when investigating pitch analysis it is informative to understand how the human auditory system functions.

The input to the peripheral system is an acoustic signal and the output is a collection of neural spikes that enter the brain (Gold & Morgan, 2000, p. 195).

Figure 1-10 shows a simple diagram of the human ear.

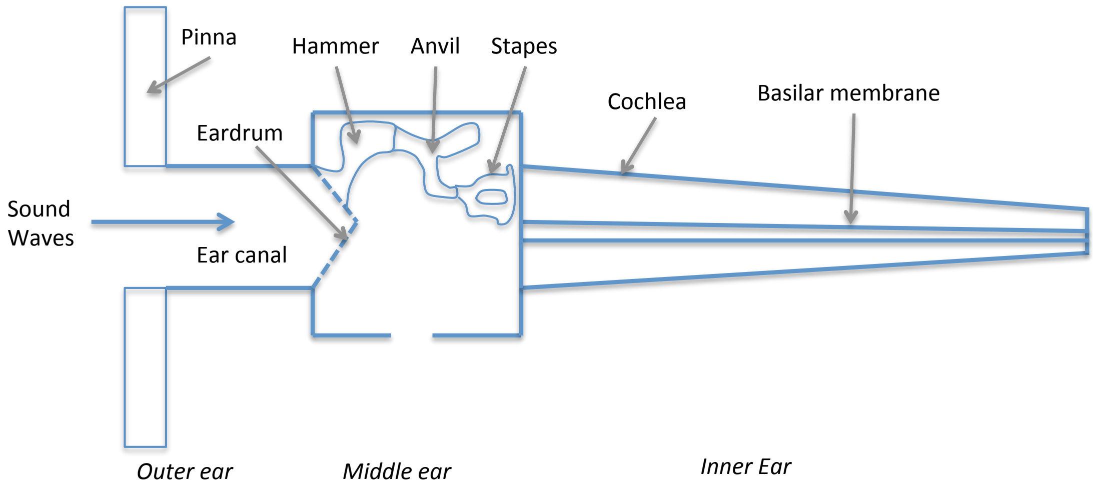


Figure 1-10 The human auditory system

A simplified diagram of the human ear. Image modified from (Rossing, Moore, & Wheeler, 2002, p. 84)

Sound enters the ear and travels down the auditory canal and is transmitted to the eardrum where the acoustic energy is transformed to vibrational mechanical energy in the middle ear. The hammer, anvil and stapes transfer the vibration from the eardrum to the inner ear. The stapes motion impinges on the oval window of the inner ear, which is a flexible membrane, and its motion sets the fluid within the cochlea in motion. The motion of the fluid is transferred to the basilar membrane within the cochlea. This is where frequency detection occurs.

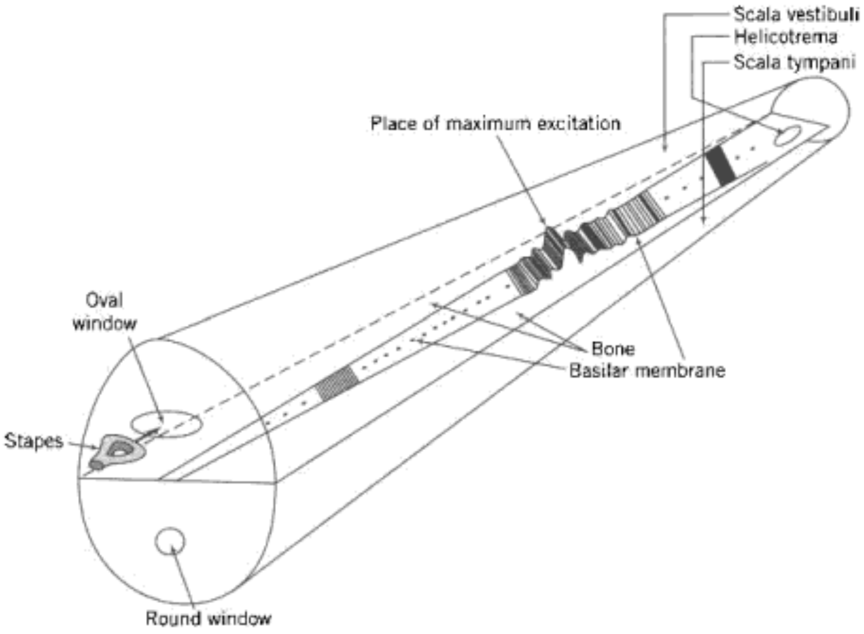


Figure 1-11 The cochlea

A simplified model of the cochlea. Image source: (Gold & Morgan, 2000, p. 193)

The position where the stapes impinges on the oval window of the cochlea is called the base; the far end of the cochlea is the apex. Near the base of the cochlea the basilar membrane is relatively narrow and stiff, and at the apex it is wider and less stiff. This structure results in high frequencies exciting the basilar membrane at the base but vibrations subside as they approach the apex. Low frequencies enter the cochlea at the base but agitate the basilar membrane to maximum amplitude at the apex. The vibration of the basilar membrane at different points indicates different frequency content. It is this function that leads to the supposition that the basilar membrane action is akin to a filter bank (Klapuri, 2006b, pp. 234-237) (Figure 1-12).

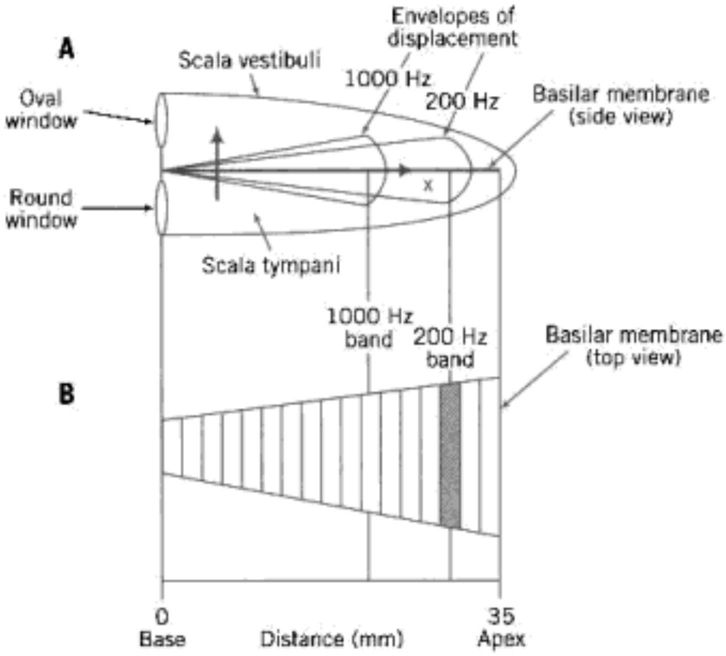


Figure 1-12 The basilar membrane as a filter bank

A representation of the activity along the basilar membrane. The filter bank comparisons of the basilar membrane are clear. Image source: (Gold & Morgan, 2000, p. 193)

The motion of the hairs, or stereocilia on the basilar membrane causes firing of the auditory nerves that connect to the hair cells and it is the spikes produced by the auditory neurons that relay all auditory information to the brain for interpretation. To transfer this model of the ear to a pitch perception algorithm, the basilar membrane can be considered the low level processing, and the brain as the high level processing – interpreting the data from the low level processing.

The model of the peripheral human auditory system as a sophisticated filter bank is the basis of a significant amount of pitch detection research and theory (Fletcher, 1938) and is still common in modern audio analysis algorithms (Klapuri, 2008). Klapuri reasons that as it is the most accurate transcription system known, then it is sensible to imitate its functionality (Klapuri, 2006b, p. 229)

1.5 Critical Bands

The functionality of the basilar membrane as a filter bank is the basis of 'Auditory filter' research. The America physicist Harvey Fletcher was a leader in the field of auditory filters and in the 1940s introduced the term 'critical band', which referred to the then loosely defined bandwidths of the auditory filter (Swets, Green, & Tanner, 1962).

A pure tone input to the basilar membrane will not agitate just a single hair, but a large number of hairs. If 2 pure tones of similar frequency are present, the agitation of the hairs in the basilar membrane will be similar for both tones, i.e. they will stimulate the same receptors. When there is significant overlap of which hairs are stimulated, it is said the 2 tones fall in the same *critical band*. The effect of 2 frequencies being present in the same critical band is linked to the inability of the auditory system to resolve 2 frequencies that are close together as the louder of the 2 frequencies will mask the other in the same critical band (Campbell & Greated, 1994). Critical bands allow the discrimination of different sounds simultaneously only when the 2 or more frequencies fall within separate critical bands to each other. (Roland-Mieszkowski, 1994)

The basilar membrane has 24 critical bands, with each band roughly equating to a width of 1 third of an octave (Zwicker, 1961). However, when a single sound source is heard in isolation (where there is no issue with masking), the ear can discern pitch variances of less than 1 critical band. (Roland-Mieszkowski, 1994)

The total number of pitch steps perceptible by the human auditory system is approximately 1400, which is far greater than the number of notes in the range of traditional western harmony and musical instruments. (Olson, 1967, pp. 248-251)

Having discussed the properties of sound waves and how the human body detects and perceives pitch, it is important to now consider pitch in musical terms. There are basic properties and fundamentals of western musical tonality that are of importance when discussing and designing music transcription algorithms.

1.6 Western Musical Tonality

Modern popular western music is composed using the equal temperament. The equal temperament tuning divides each octave into 12 semitones which are all equal on a logarithmic scale and is *usually* tuned relative to a standard pitch of around 440Hz, which is widely accepted as concert A. Although the exact frequency of concert A does vary between orchestras, the equal tuning ensures the intervals between notes remains constant. The frequency ratio between 2 adjacent notes is the twelfth root of 2, or 2 to the power of 1/12.

For the purposes of this thesis the most important property of the equal tempered scale is the logarithmic relationship between adjacent note frequencies. This logarithmic property results in low frequency notes being closer together in terms of Hertz, than high frequency notes. This is significant when considering the frequency resolution of low level processing for music transcription algorithms.

Figure 1-13 shows 4 octaves of equal tempered notes starting at 440Hz, demonstrating the logarithmic increase in frequency of the equal tempered scale

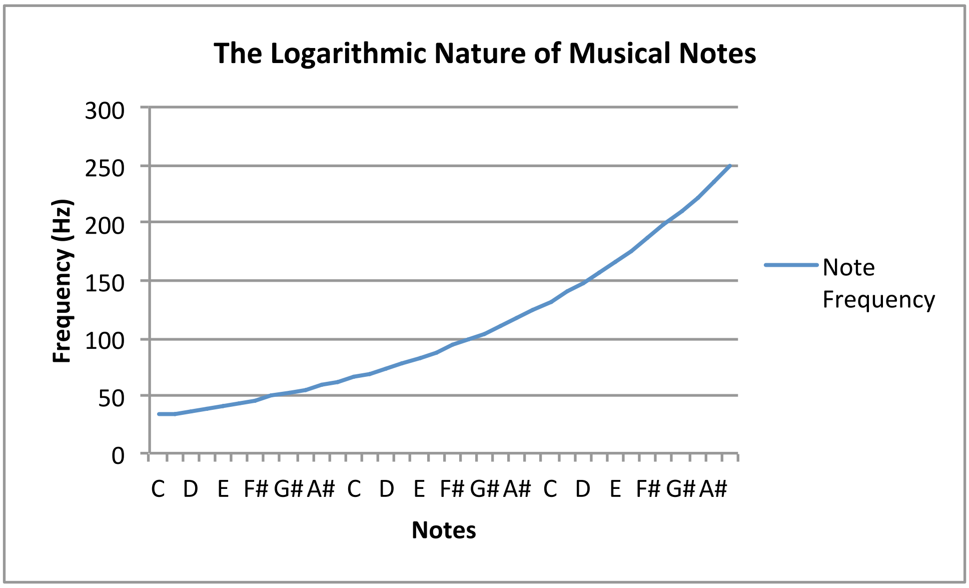


Figure 1-13 Logarithmic note frequencies

Automatic music transcription is part of a larger area entitled Music Information Retrieval, which can be sub divided into different categories This next section will introduce the different categories and also some of the problems and issues that make the process of using computers to transcribe music so difficult.

1.7 Categories of Music Information Retrieval

The Music Information Retrieval Evaluation eXchange (MIREX), an annual evaluation of music information retrieval systems (MIREX, 2010b) has categorised different areas of music information retrieval for the purposes of the conference. Table 1 is a brief explanation of the key categories as defined by

Mirex for the 2010 conference, and while it is not categorisation of music information retrieval per se, it does provide a useful set of definitions.

Category	**Task**
Audio Key Detection	Identify the musical key of pre recorded music
Audio Cover Song Identification	Identify other versions/recordings of an original query audio track.
Real-time Audio to Score Alignment	Requires the algorithm to align an incoming music signal to the corresponding musical score.
Query by Singing/Humming	Using a sung or hummed input signal the algorithm will identify the correct score from a database.
Audio Chord Estimation	Requires the algorithm to extract or transcribe a sequence of chords from a musical recording.
Audio Melody Extraction	Identify and extract the melody line from a polyphonic recording.
Audio Beat Tracking	Track each beat location in a sound file.
Audio Music Similarity and Retrieval	Queries music files to group similar music together.
Structural Segmentation	Identify the segments or 'form' of a piece of music.
Audio Tempo Extraction	Extract the tempo of a piece of music.
Audio Onset Detection	To find the time locations of musical events e.g. Notes in a recording
Multiple Fundamental Frequency Estimation & Tracking	Estimate the fundamental frequencies present in a piece of audio and track their changes over time.

Table 1 Categories of music information retrieval

(MIREX, 2010c)

The Multiple Fundamental Frequency Estimation Tracking task is of particular interest for this thesis. The task deals with the concept that a complex music signal can be represented as a series of fundamental frequency contours. The goal of this discipline is to identify the fundamental frequencies present in each time frame, and use this information to track notes through a complex music signal. This is a complex task and tracking all fundamental frequencies in an audio mixture is very difficult. Therefore, MIREX limit the problem to 3 cases:

- Estimate active fundamental frequencies on a frame-by-frame basis.
- Track note contours on a continuous time basis. (As in audio-to-MIDI).
- Track multiple timbres on a continuous time basis.

(MIREX, 2010c)

The category of fundamental frequency estimation and tracking is a good example of the importance of both time and frequency resolution in automatic music transcription, and is primarily the category of interest for this thesis. A good frequency resolution is required to accurately detect fundamental frequencies, but also a good time resolution is required to accurately identify the timing of frequency onset. The next section outlines some of the challenges associated with multiple fundamental frequency estimation.

1.8 Challenges Associated with multiple fundamental frequency estimation

Figure 1-14 shows a spectrogram of a recording of a conversation taking place in an environment containing many other background conversations and noises.

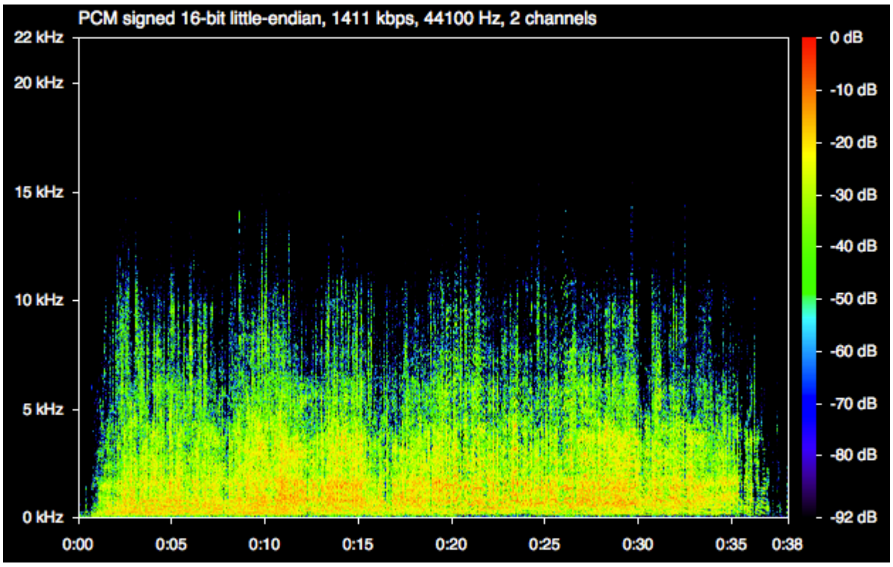


Figure 1-14 Spectrogram of a conversation

The task of reading a complex spectrogram as in Figure 1-14 and extracting a single sound source would be impossible for even an expert spectrogram reader (Bregman, 1994), even though the human auditory system can decipher the sound.

The difficulty of reading the spectrogram is due to sounds overlapping in both time and frequency – this is a fundamental difficulty in multiple Fundamental frequency estimation (Multiple F0 Estimation). The overlapping of sounds causing one not to be heard is termed *Auditory Masking* in psychoacoustics (Wegel & Lane, 1924). Wegel and Lane's investigation of masking focused on the auditory system response to sound masking (as discussed in section 1.5) and the effect of partials of lower frequency sound interfering with higher frequency fundamentals (Figure 1-15).

Wegel and Lane found that the masking is greatest for tones nearly alike. When the masking tone is loud it masks tones of higher frequency better than those of frequency lower than itself. When the masking tone is weak, there is little difference. (Wegel & Lane, 1924)

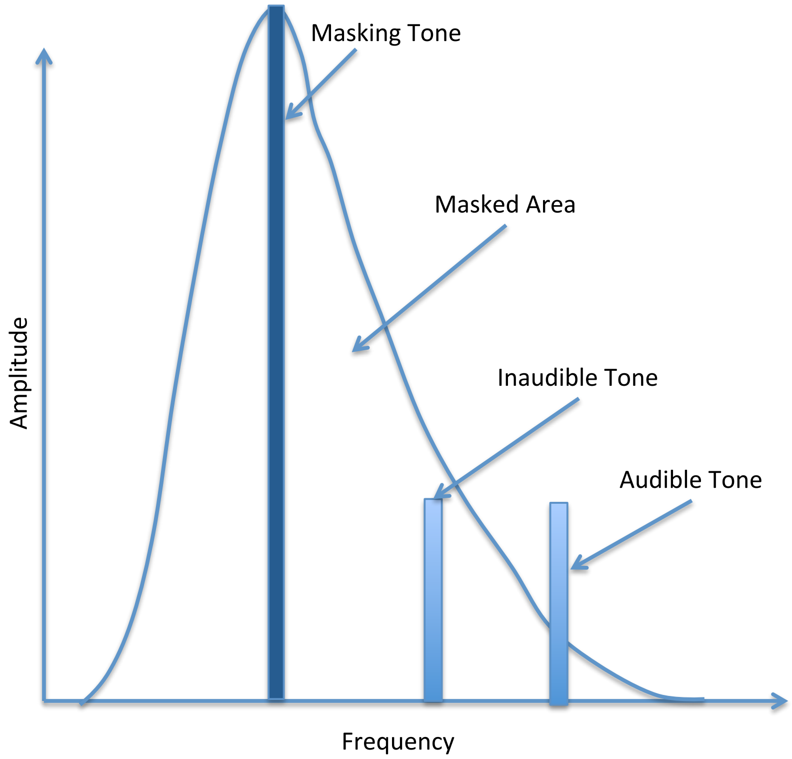


Figure 1-15 Auditory masking

Wegel and Lane’s work refers to masking of sounds which humans cannot detect, but Yeh (Yeh, 2008) refers to masking of sounds which increase the difficulty for computers to track fundamental frequencies, but which the human ear can hear.

Yeh refers to the difficulty of overlapping time and frequency components of sound sources in more musical terms. He states that when musical notes are played in harmonic relations, i.e. in the same key or scale (which is typical of western popular music), the harmonics or *partials* of the higher notes may completely mask, that is overlap, those of lower notes (Yeh, 2008). This, combined with the diverse spectral characteristics of musical instruments, results in greater ambiguity in the estimation of partial amplitudes increasing the difficulty of accurately extracting and tracking fundamental frequencies through a piece of music. A spectrogram of a monophonic (single source) recording (Figure 1-16) compared to that of a polyphonic source (Figure 1-17) clearly shows the difficulty of polyphonic fundamental frequency estimation.

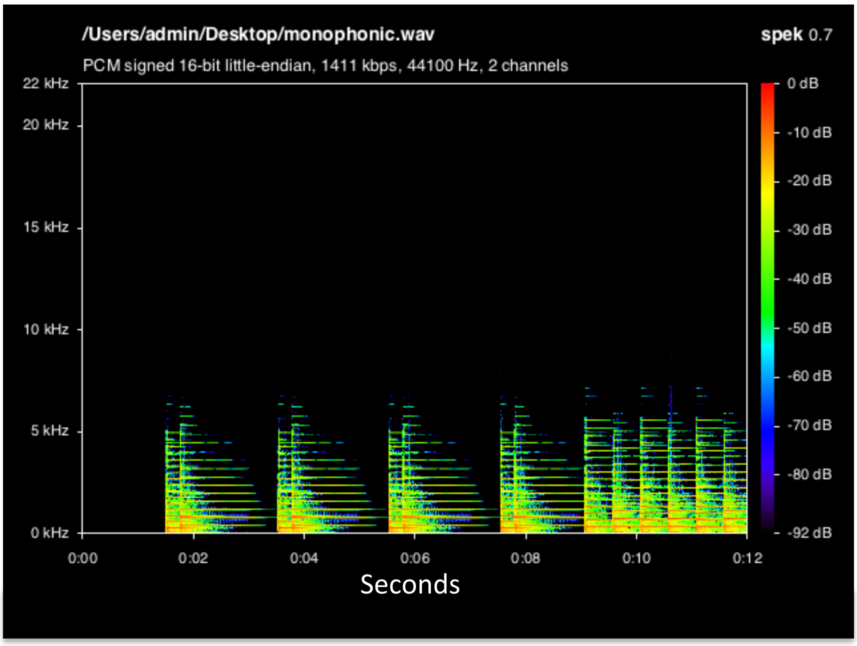


Figure 1-16 A monophonic Piano Line

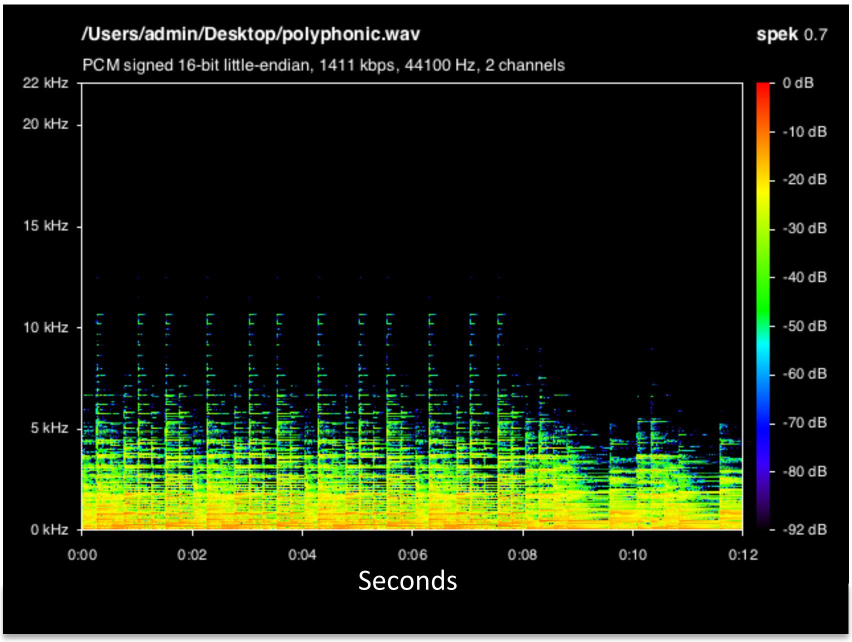


Figure 1-17 a polyphonic piano line and bass line

The time and frequency overlap of sound sources is the crux of the multiple fundamental frequency estimation problem, particularly when considering the harmonic structure of music and sound. In a given piece of music, perhaps 4 or more notes may be overlapping in time, but given the theory of western tonality, the fundamentals of these notes may be in simple integer ratios, leading to a collision of their harmonics in spectral terms. This results in complex constructive and destructive interference in the frequency domain (Poliner, Ellis, Ehmann, Gomez, Streich, & Ong, 2007), which in part contributes to the complex spectral representation of music.

The clear deciphering of fundamental frequencies from spectral representations is the starting point for a very active area of research as people explore different techniques and methods to transcribe notes from an audio mixture.

1.9 Structure of Fundamental Frequency Estimation Algorithms

Numerous single and multiple Fundamental Frequency (F0) Estimators have a similar basic processing structure as that shown in Figure 1-18.

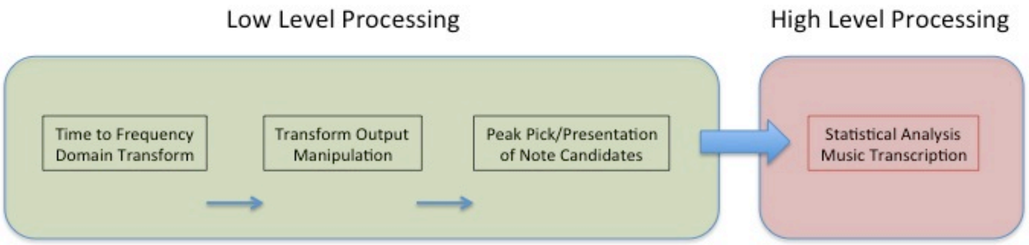

Low Level Processing
High Level Processing
Time to Frequency Domain Transform
Transform Output Manipulation
Peak Pick/Presentation of Note Candidates
Statistical Analysis Music Transcription

Figure 1-18 A commonly used structure of automatic music transcription algorithms

Digitized audio is transformed from the time domain to the frequency domain. From the output of this transform frequencies are selected as note candidates. Further processing is performed to determine the correct notes from the note candidates and the chosen notes are transcribed into a score – which is typically generated as a MIDI file.

Although common, this structure is not exclusively followed (e.g. (Cheveigne & Kawahara, 2002) but the presented structure in Figure 1-18 will be assumed as the starting point for the following discussion and form the inspiration for the optimisation investigation.

For the purposes of this thesis *low level processing* refers to the transformation of a digital music source from the time domain to the frequency domain, any manipulation of the transform output and the peak picking process to present note candidates. *High level processing* refers to any analysis of the output data from the low level transform to present a series of fundamental frequencies that represent the original acoustic signal.

2 Low Level Processing – Single Resolution Analysis

Low level processing refers to the techniques used to extract frequency information from a time domain musical signal. The frequency domain information is critical for automatic music transcription as it is the frequency content that determines the note pitch to be transcribed. The techniques described in the following section are implemented as the initial stage of the majority of music information retrieval algorithms, but will be discussed specifically in the context of fundamental frequency estimation and onset detection.

The purpose of the low level processing stage is to present the spectral information of the signal being transcribed as accurately as possible, representing the fundamental frequencies and harmonics (dependent on the type of high level processing used) clearly. If the initial low level processing can present strong 'note candidates' i.e. clear spectral maxima to the high level processors, then the likelihood of those candidates being 'true' is increased from the outset, resulting in an easier high-level process to discern notes.

The characteristics of a desirable low-level process are:

- A good time resolution to accurately locate a frequency in time
- A good frequency resolution to accurately represent adjacent note frequencies

Low level processing can be divided into 2 categories, single resolution transforms where a single time-frequency resolution is used across the entire spectrum, and multiresolution transforms which use a variable time-frequency resolution across the frequency spectrum – typically by splitting the initial signal into different frequency bands. Single resolution transforms can be sub divided in to frequency domain and time domain methods.

2.1 Time Domain Low Level Processing

Time domain low level processing methods look for repetitive patterns in the waveform to determine a periodicity, and therefore a frequency. Time domain approaches to pitch extraction have been used with successes for monophonic pitch estimation (Rabiner, On the Use of Autocorrelation Analysis for Pitch Detection, 1977) but such approaches are not suitable for multiple pitch estimation due to the spectral complexity of the signal. However, for completeness it is useful to have an understanding of these basic methods.

2.2 Zero Crossing

The zero crossing is the point where a waveform intersects the zero point, changing sign from positive to negative (Figure 2-1).

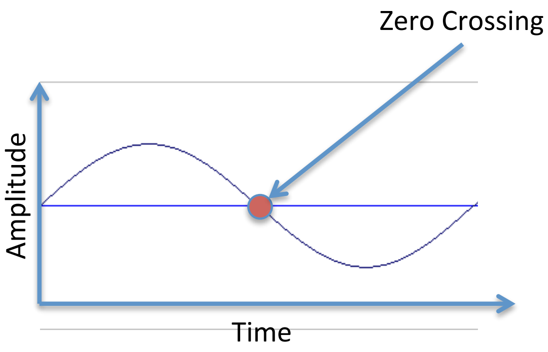


Figure 2-1 Zero Crossing

By tracking the time between zero crossings the period of the waveform and therefore frequency can be calculated. This technique has been used as a crude fundamental frequency estimator for speech processing (Veeneman, 1988) as well as other disciplines such as the classification of percussive sounds (Gouyon, Pachet, & Delerue, 2000), but as a stand-alone low level process for polyphonic pitch extraction the complex waveforms render it wholly inaccurate (Roads, 1996, p. 508).

2.3 Autocorrelation

Correlation functions compare two signals with the goal of finding similarity between the two signals (Roads, 1996, p. 509). Autocorrelation compares a signal with versions of itself delayed by regular intervals. The comparing of delayed versions results in finding underlying periodic signals from noisy signals (Figure 2-2).

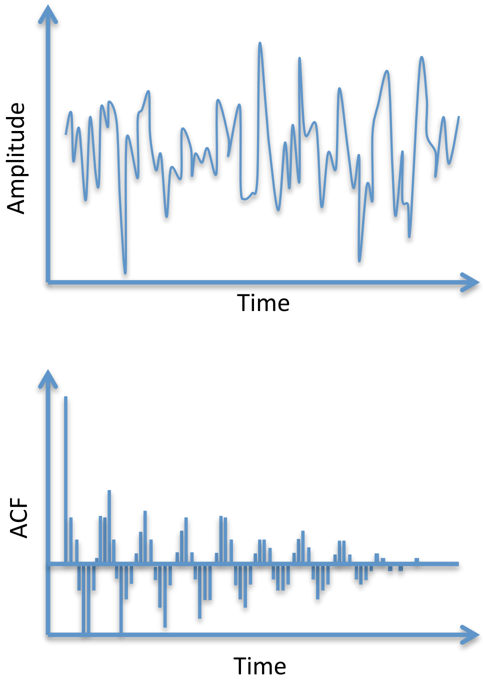


Figure 2-2 Autocorrelation

The top diagram shows a time domain signal with a 'hidden' sine component. The bottom diagram shows the result of the autocorrelation function on the time domain signal.

Autocorrelation has been used as the main process with success in monophonic pitch estimation (Rabiner, On the Use of Autocorrelation Analysis for Pitch Detection, 1977), particularly in speech recognition (Kida, Sakai, Masuko, & Kawamura, 2009) and is still a powerful tool for auditory model based methods for multiple F0 estimation (Klapuri, 2006b). These examples all use the same basic autocorrelation process for pitch detection (Figure 2-3)

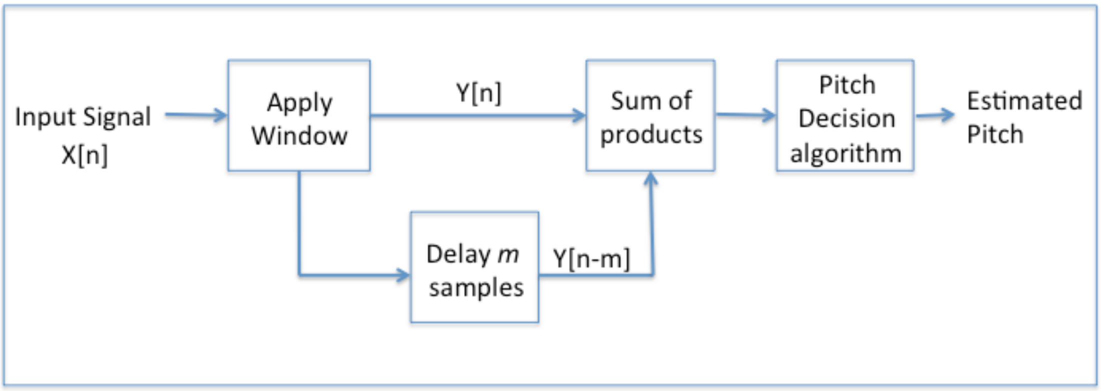


Figure 2-3 Auto correlation process

Part of the input signal is delayed in a buffer, and as more of the input signal comes in, the detector attempts to match a pattern in the incoming signal with the part of the waveform delayed in the buffer (Roads, 1996, p. 510). If the detector finds a match between the two signals periodicity is indicated. The time interval between the two waveform patterns is measured and the frequency is calculated.

Although various autocorrelation algorithms exist (Moorer, 1975) a typical function is

$$Autocorrelation[lag] \; = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N}} signal[n] \; x \; signal[n + lag]$$

Equation i

Where:

N   is the length of the input signal.

The magnitude of the *autocorrelation[lag]* is determined by the similarity of the values of *signal* at different points *n* and *n+lag.*

When attempting to detect periodicity in more complex signals, the 'pitch decision' algorithm will search for recurrent peaks in the autocorrelation (Roads, 1996, p. 511).

The difficulty with autocorrelation techniques is that peaks can occur at sub harmonics, making it difficult to determine which are fundamental frequencies (Gerhard, 2003). Modification of the basic autocorrelation function is not uncommon to minimize the errors generated from the basic implementation (Cheveigne, 1991).

Cheveigne and Kawahara presented the YIN estimator, which uses an adapted version of the autocorrelation function as it's low level processing (Cheveigne & Kawahara, 2002). YIN utilises a 'cumulative mean normalized difference function', which is a squared difference function normalized with it's average over short lag values. This modification reduces error rates from 10% to 1.69% compared to the standard autocorrelation function (Cheveigne & Kawahara, 2002) and has become a much-cited algorithm in the field.

The following section will discuss low-level frequency domain processing, and primarily the Fast Fourier Transform.

2.4 Frequency Domain Low Level Processing

Frequency domain low level processing refers to methods that present spectral information as an output by transforming the time domain signal into the frequency domain. The most famous of these algorithms is the Fourier

Transform. Many of the leading music transcription algorithms use the Fourier Transform to view the spectral components of a signal (Goto 2006, Klapuri 2006), so it is important to discuss the Fourier family of transforms, their properties and characteristics to understand their positive and negative attributes for the purpose of fundamental frequency estimation and onset detection.

2.5 Fourier Analysis

The Fourier transform is a mathematical operation that decomposes a time signal into its component frequencies, generating a corresponding spectrum representation (Roads, 1996, p. 550).

Fourier analysis is named after Jean Baptiste Joseph Fourier (1768-1830), a French mathematician who contributed significantly to the field.

2.5.1 The Fourier Family of Transforms

The differentiation between the categories of transforms in the Fourier family is based on the signal in can transform.

A signal can be either continuous or discrete, and it can be either periodic or aperiodic. These properties generate the 4 categories of Fourier transform which are described in the following diagram (Smith S. W., 1997)

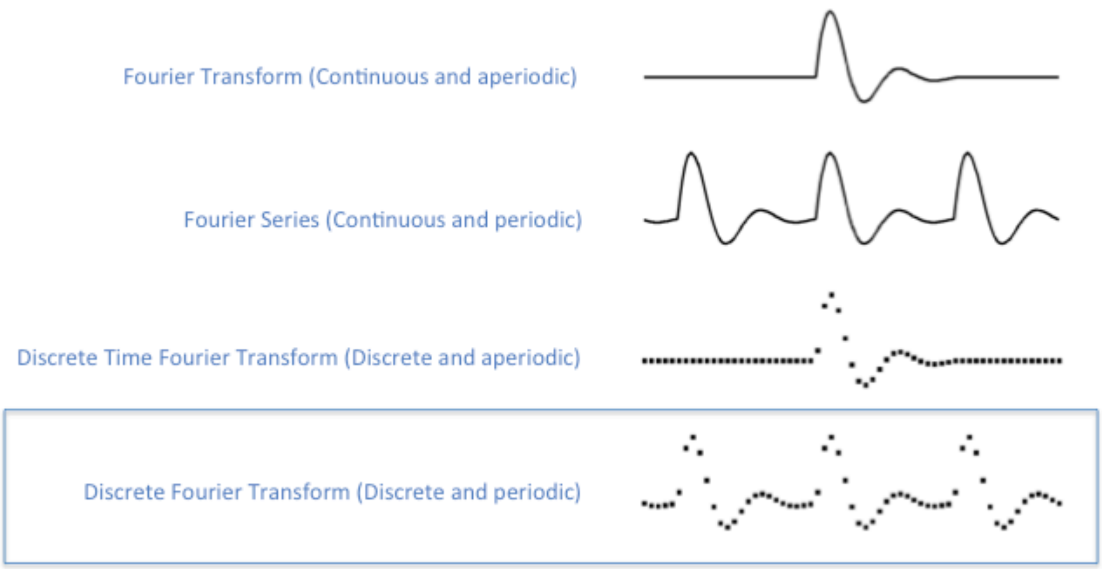

Fourier Transform (Continuous and aperiodic)
Fourier Series (Continuous and periodic)
Discrete Time Fourier Transform (Discrete and aperiodic)
Discrete Fourier Transform (Discrete and periodic)

Figure 2-4 The Fourier family

Image modified from (Smith S. W., 1997, p. 145)

The Discrete Fourier Transform (DFT) (boxed in blue in Figure 2-4) is utilised in DSP as digital computers can only work with a discrete and finite amount of data (samples), therefore ruling out the use of the other 3 transforms.

The above 4 categories of signal including the DFT, in mathematical terms all extend to negative and positive infinity, and what is shown in Figure 2-4 is only a small section of a mathematically infinite signal. However, only a finite number of samples of a signal are used during a DFT, therefore this discrepancy needs to be resolved, as discussed in the following section.

2.5.2 Periodicity of the DFT

As shown in Figure 2-4 the DFT is periodic i.e. it views both the time and frequency domain as periodic. This may seem unsuitable for use in DSP as most signals used in DSP are not periodic but constantly changing, but a mathematical characteristic of the DFT is that it views a time domain signal as a section of a periodic signal which extends to infinity. To use the DFT to analyse a finite signal, the finite signal is made to look infinite by duplication of the finite signal as imaginary points either side of the actual signal. This results in the signal appearing to be discrete and periodic, thus matching the criteria for the DFT (Figure 2-5).

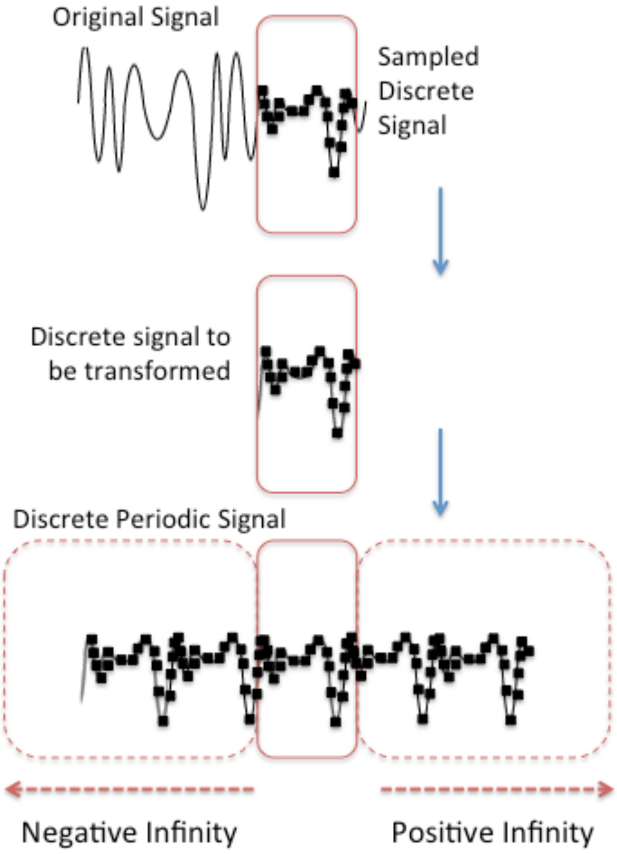


Figure 2-5 Discrete periodic signal

2.5.3 The DFT Decomposition – An Introduction

The DFT decomposes a time domain signal into a series of component sine and cosine waves.

Each member of the Fourier family of transforms can be sub-dived into *real* and *complex* versions. The real version does not use complex numbers for the decomposition process and is therefore relatively simple. The *complex* version requires the use of *complex numbers,* which is the method of the FFT.

Smith (Smith S. W., 1997) is a useful single and easy-to-follow source for DSP fundamentals. The following section on the DFT and FFT is a summary of the content that Smith presents in his widely referenced book regarding the DFT and FFT.

The DFT can be calculated in three different ways. The first is by simultaneous equations, but this method is too inefficient to be of practical use. The second method is by correlation and the third method is by using the Fast Fourier Transform.

Although simultaneous equations and correlation methods will arrive at the same result as the FFT, the speed and efficiency of the FFT is significantly better, improving computation times by hundreds. The following section introduces the decomposition method used in the Fast Fourier Transform.

2.6 The Fast Fourier Transform

Tukey and Cooley are credited for introducing the FFT in 1965 (Cooley & Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, 1965), but in reality others such as Karl Friedrich Gauss (1777-1855) had discovered the technique many years earlier (Smith S. W., 1997, p. 225). This early work was forgotten as the tools were not available to make it practical, but Cooley and Tukey's introduction of the FFT coincided with the computer revolution.

The FFT calculates the *complex* DFT. The practical mathematics of the complex DFT and the FFT is complicated, but it is useful to have a basic understanding of how the FFT calculates the DFT.

2.6.1 The Complex DFT

The complex DFT transforms an N point time domain signals a real part, and an imaginary part in to two N point frequency domain signals (Figure 2-6).

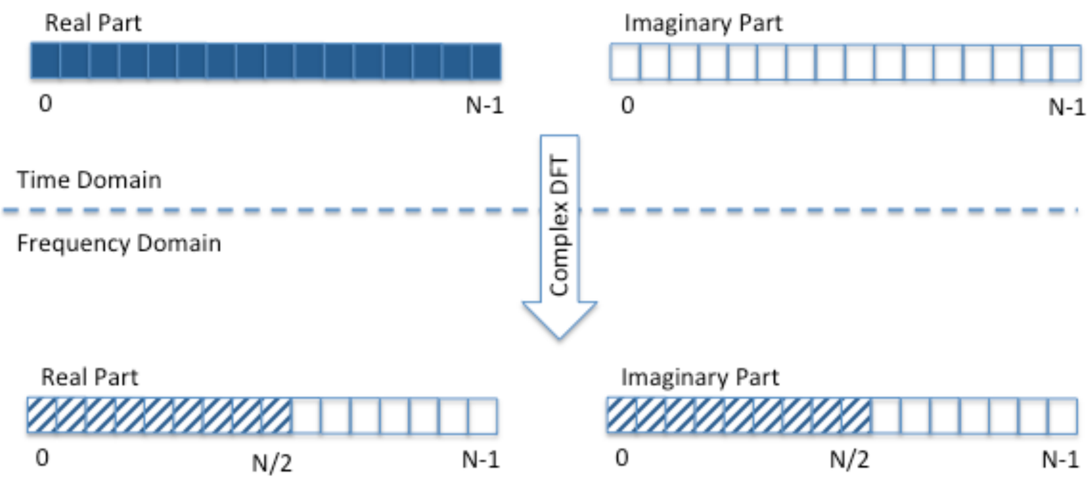


Figure 2-6 Complex DFT

The complex DFT decomposition transforms both real and imaginary parts in the time domain to the frequency domain. Shaded areas show values common to the real DFT.

The real and imaginary parts of the time domain signals are represented in the FFT collectively as N *complex* points. Complex points are composed of 2 values, the real and imaginary parts. As each complex point holds two numbers, when one complex point is multiplied by another the four components need to be combined to form the two components of the produced complex variable. This brief introduction to complex numbers in the FFT is useful to know when discussing the FFT decomposition process in section 2.6.2, and more specifically the FFT butterfly.

2.6.2 FFT Decomposition

The decomposition performed by the FFT is what makes the FFT fast in comparison to the simultaneous equations and correlation methods. The following is a summary description of the FFT decomposition process, emphasizing its speed and efficiency, rather than the complexities of functionality, which are not relevant to the purpose of this project.

There are three stages to the FFT decomposition

- Decomposing an N point time domain signal into N time domain signals each a single point
- Calculate the N Frequency corresponding to each of the N time domain signals
- Synthesize the N Spectra in to a single frequency spectrum

Smith's 16-point time domain signal example will be used as a simple explanation of the FFT decomposition process.

The first stage divides the 16 point signal in a pyramid structure where one signal of 16 is split in to two signals of 8, is split in to four of 4 until there are sixteen signals of 1 point. Duhamel and Vetterli refer to this as the 'divide and conquer' method (Duhamel & Vetterli, 1990). Each time a signal is separated an *interlace decomposition* is used to separate the signal in to its odd and even numbered points. The figure below shows this process.

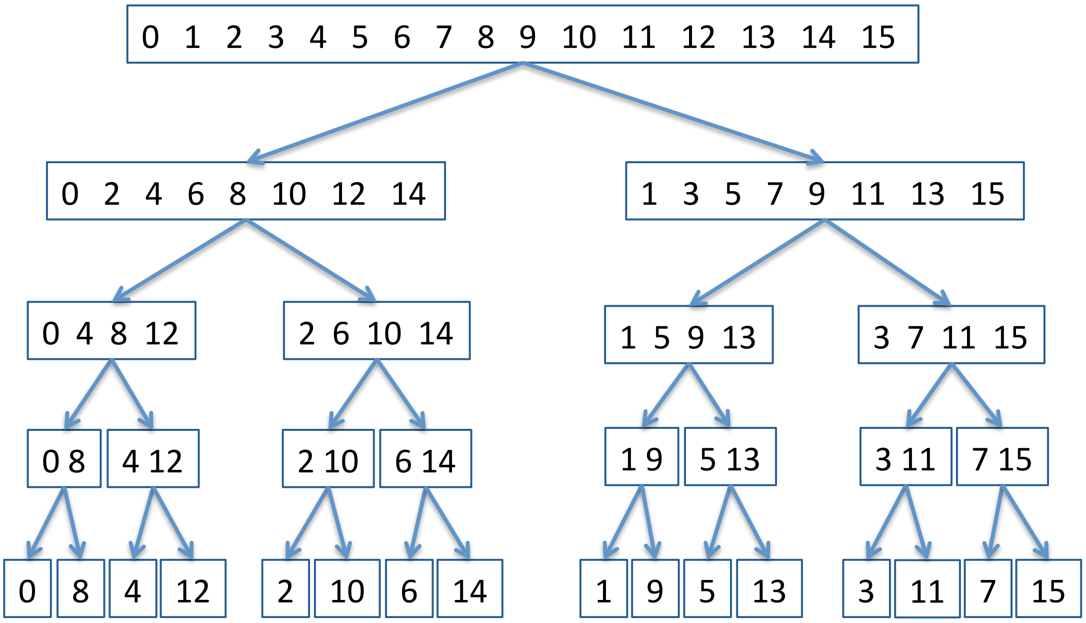


Figure 2-7 FFT Sample ordering

The output of the N point decomposition process shown in Figure 2-7 is essentially the result of a bit reversal sorting algorithm. Bit reversing involves rearranging the 16 time domain samples based on the flipping of their binary representations (Table 2)

Samples In 'normal' order	
Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Samples after bit reversal	
Decimal	Binary
0	0000
8	1000
4	0100
12	1100
2	0010
10	1010
6	0100
14	1110
1	0001
9	1001
5	0101
13	1101
3	0011
11	1011
7	0111
15	1111

Table 2 Bit reversal

The table on the right shows the decimal numbers reordered as a product of reversing the binary numbers from the table on the left.

Stage two of the FFT is to determine the frequency spectra of the 1-point time domain signals. This is the simplest step as the frequency spectrum of a 1 sample signal is equal to itself, therefore nothing is involved is this step to take the 1 point signal from the time domain to the frequency domain. Each 1 point signal is now a frequency spectrum, not a time domain signal

The third step of the FFT algorithm is more complicated as it involves combining the 16 points of the frequency spectra in exactly the reverse order that the time domain decomposition took place, undoing the interlaced decomposition performed in the time domain. However, the bit reversal method is not applicable. Instead, the process must be performed one step at a time, synthesizing the sixteen 1-point spectra in to eight 2-point spectra, in to four 4-point spectra etc. The last stage results in the output of the FFT being a 16-point frequency spectrum.

2.6.3 Frequency Domain Reordering and Butterflies

The method for combining the points of the frequency spectra involves diluting the N point time domain signals to be decomposed/synthesized with zeros. Lets take the process of combing two 4-point signals into a single 8-point signal to explain the process.

A four-point signal *abcd* becomes *a0b0c0d0* and when combined with a second signal of *0e0f0g0h* the synthesis of the two former 4-point signals becomes a single 8-point signal of *aebfcgdh.* Diluting the time domain signal with zeros results in a duplication in the frequency spectrum. The FFT combines the frequency spectra by duplicating the spectra and then summing them.

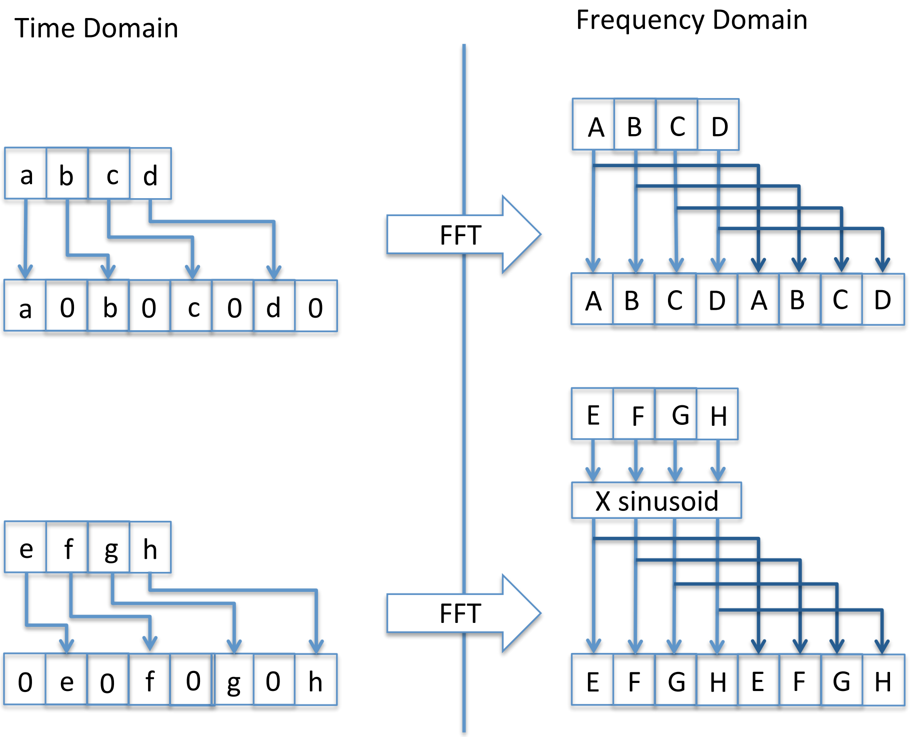


Figure 2-8 Spectral combination

Image modified from Smith (Smith S. W., 1997, p. 230)

One signal has been diluted at the even points, the other at the odd points to ensure the signals match up when added. An alternative way to view the dilution with zeros is the second signal has been shifted to the right by one point. This shift in the time domain corresponds to multiplying the spectrum by a sine wave. The diagram below shows the method of combining two 4-point frequency spectra into a single 8-point spectrum. ‘xS’ denotes the operation of multiplying the signal with a sinusoid of an appropriate frequency determined by Fs/N.

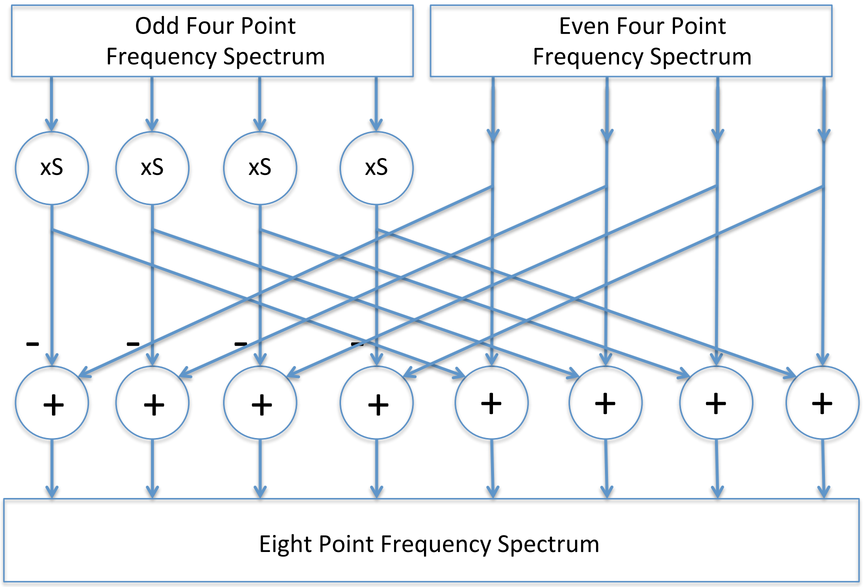

Odd Four Point Frequency Spectrum
Even Four Point Frequency Spectrum
xS
xS
xS
xS
-
-
-
-
+
+
+
+
+
+
+
+
Eight Point Frequency Spectrum

Figure 2-9 combining two 4-point frequency spectra into a single 8-point spectrum

Figure 2-9 combining two 4-point frequency spectra into a single 8-point spectrum *modified from* (Smith S. W., 1997, p. 231)

The diagram above is formed from a single basic calculation, which is repeated many times. This basic calculation is known as the FFT 'butterfly' and is the most fundamental element of the FFT, converting 2 complex points into two other complex points (Figure 2-10).

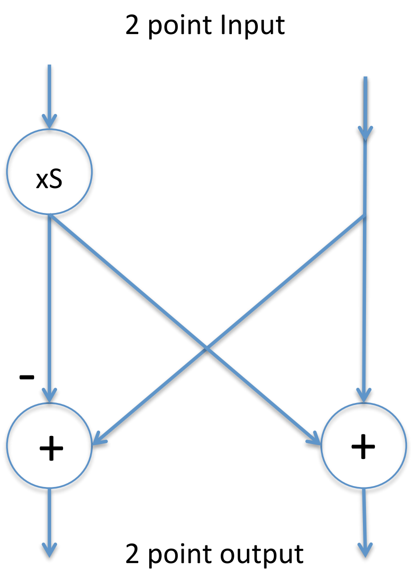


Figure 2-10 FFT Butterfly

This method of FFT decomposition is based on the Cooley and Tukey radix-2 FFT (Cooley & Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, 1965). Power of 2 FFTs are popular due to their speed and efficiency, but other Fast Fourier Transforms have been developed which allow for non-power of 2 numbers of samples.

2.7 Non Power of 2 FFTs and the Fastest Fourier Transform in the West

Tukey and Cooley, when they presented their paper used a power of 2 decomposition as an example (Cooley & Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, 1965), but the algorithm actually included a 'twiddle factor' which allowed for non power of 2 sample sizes to be used. It is only because of their example that it assumed to be a radix-2 only transform (Duhamel & Vetterli, 1990). In very basic terms, the difference between the different FFT algorithms in use is the process of transforming from N time domain samples to N samples of the frequency domain. The usual measurement of success is the efficiency in which it can be done (Duhamel & Vetterli, 1990).

A popular algorithm in current DSP practice is known as 'The Fastest Fourier Transform in the West' or FFTW. The FFTW is an open source software library that is widely regarded as the fastest FFT by adapting its performance to the N points it is presented with and the hardware it is run on (Frigo & Johnson, 2005). It is the FFTW included with the Matlab software (Moler, 2005) that is used in the investigation of FFT parameters and characteristics in Chapter 6.

Considering automatic music transcription as the application for a FFT the number of FFT points is of significance as it directly relates to the time resolution – that is the length of time the spectrum represents, and also the frequency resolution – that is how many component sine waves are available to represent the frequency content. To address the time and frequency resolution the Short Time Fourier Transform is popular for music analysis.

2.8 The Short Time Fourier Transform

If a DFT is performed on the entirety of a pop song, there is no way of knowing which frequencies in the spectral information occurred at the start of the song, in the first line, or the first word - there is no time information to localize frequency maxima to a point in time.

The Short Time Fourier Transform (STFT) functions as the DFT does, but analysis is performed on small 'windows' of the signal being analysed. Once the content inside the 'window' has been transformed, the window will move along the signal by a number of samples (usually equal to the window length or less) where the next part of the signal will be transformed. This method allows the spectral information to be associated with a finite amount of time equal to the window length within the context of the entire signal being analysed. Spectrograms are constructed by aligning adjacent STFT windows.

This positive aspect of localizing frequency spectra to a point in time is also the major negative of the algorithm. Due to the decomposition method of the DFT, if a short time frame is used, i.e. fewer samples, there are fewer sinusoids to represent the frequency components, therefore the size of each 'bin' is greater and the accuracy of the frequency values compared to the actual values in the signal is compromised. To improve the spectral accuracy the window must be enlarged, but then the ability to localize a frequency domain event in the time domain is compromised as is the ability to detect fast changes. This is discussed further in section 6.3.

Despite the time-frequency trade off, the STFT is a highly popular method of extracting spectral information from an audio signal for purposes of automatic music transcription. An analysis of the algorithms submitted to MIREX 2010 show a large number use the STFT algorithm (Table 3). Page numbers refer to the MIREX 2010 complete proceedings (MIREX, 2010a).

Authors	**Title**	**Pages**	**Low Level Processing**	**Window Size & Other Information**	**Higher Level Processing**
Grindlay, Ellis	A PROBABILISTIC SUBSPACE MODEL FOR MULTI-INSTRUMENT POLYPHONIC TRANSCRIPTION	20 - 26	STFT	1024	NMF
Coz, Lachambre, Koenig, Obrecht	A SEGMENTATION-BASED TEMPO INDUCTION METHOD	27-31	STFT	Not stated	Comb Decision' - Harmonic Analysis

Joder, Essid, Richard	AN IMPROVED HIERARCHICAL APPROACH FOR MUSIC-TO-SYMBOLIC SCORE ALIGNMENT	39 - 44	STFT generated Chroma Vectors		HMM
Yoshii, Goto	INFINITE LATENT HARMONIC ALLOCATION: A NONPARAMETRIC BAYESIAN APPROACH TO MULTIPITCH ANALYSIS	309 - 314	Wavelet Transform	60ms Time Resolution	Bayesian Variation
Eyben, Bock, Schuller, Graves	UNIVERSAL ONSET DETECTION WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY NEURAL NETWORKS	589 - 594	MRFFT	1024, 2048	Neural Networks
Wang, Li, Ogihara	ARE TAGS BETTER THAN AUDIO FEATURES? THE EFFECT OF JOINT USE OF TAGS AND AUDIO CONTENT FEATURES FOR ARTISTIC STYLE CLUSTERING	57 - 62	STFT		NMF
Humphrey	AUTOMATIC CHARACTERIZATION OF DIGITAL MUSIC FOR RHYTHMIC AUDITORY STIMULATION	69 - 74	22 Band Cochlea Filter Bank		Chroma
Rump, Miyabe, Tsunoo, Ono, Sagama	AUTOREGRESSIVE MFCC MODELS FOR GENRE CLASSIFICATION IMPROVED BY HARMONIC-PERCUSSION SEPARATION	87 - 92	40 Band Mel Filter Bank		MFCC analysis
Abeber, Brauer, Lukashevich, Schuller	BASS PLAYING STYLE DETECTION BASED ON HIGH-LEVEL FEATURES AND PATTERN SIMILARITY	93 - 97	STFT		Support Vector Mechanism (SVM)

Weiss, Bello	IDENTIFYING REPEATED PATTERNS IN MUSIC USING SPARSE CONVOLUTIVE NON-NEGATIVE MATRIX FACTORIZATION	123 - 128	STFT		NMF Variation
Mauch, Dixon	APPROXIMATE NOTE TRANSCRIPTION FOR THE IMPROVED IDENTIFICATION OF DIFFICULT CHORDS	135 - 140	STFT, Hamming Window	4096, 11Khz, 2048 Hop	Bayesian Network
Granseman, Scheunders, Mysore, Abel	EVALUATION OF A SCORE-INFORMED SOURCE SEPARATION SYSTEM	219 - 225	STFT	2048, 44.1Khz, 512 hop	NMF Variation
Karydis, Radovanovic, Nanopoulos, Ivanovic	LOOKING THROUGH THE "GLASS CEILING": A CONCEPTUAL FRAMEWORK FOR THE PROBLEMS OF SPECTRAL SIMILARITY	267 - 272	STFT	512, 11Khz, 256 hop	MFCC / Gaussian Mixture Model
Lidy, Mayer, Rauber, Leon, Pertusa, Inesta	A CARTESIAN ENSEMBLE OF FEATURE SUBSPACE CLASSIFIERS FOR MUSIC CATEGORIZATION	279 - 284	STFT		Various Spectrogram Analysis
Oliveira, Gouyon, Martins, Reis	IBT: A REAL-TIME TEMPO AND BEAT TRACKING SYSTEM	291 - 296	STFT, Hamming Window	1024, 44.1Khz, 512 hop	Agent Based tempo tracker
Han, Raphael	INFORMED SOURCE SEPARATION OF ORCHESTRA AND SOLOIST	315 - 320	STFT, Hann Window		Various Spectrogram Analysis
Schnitzer, Flexer, Widmer, Gasser	ISLANDS OF GAUSSIANS: THE SELF ORGANIZING MAP AND GAUSSIAN MUSIC SIMILARITY FEATURES	327 - 332	STFT	1024, 22kHz	MFCC / Gaussian Mixture Model
Marolt, Lefeber	IT'S TIME FOR A SONG – TRANSCRIBING RECORDINGS OF BELL-PLAYING CLOCKS	333 - 338	Constant Q Transform		NMF

Hamel, Eck	LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS	339 - 344	STFT	1024, 22.5kHz	DBN Neural Networks
Jo, Yoo	MELODY EXTRACTION FROM POLYPHONIC AUDIO BASED ON PARTICLE FILTER	357 - 362	STFT, Hanning	2048, 44.1Khz, 512 hop	Bayesian Particle Filter
Raczynski, Vincent, Bimbot, Sagayama	MULTIPLE PITCH TRANSCRIPTION USING DBN-BASED MUSICOLOGICAL MODELS	363 - 368	STFT		NMF/DBN Neural Networks
Murao, Nakano, Kitano, Ono, Sagayama	MONOPHONIC INSTRUMENT SOUND SEGREGATION BY CLUSTERING NMF COMPONENTS BASED ON BASIS SIMILARITY AND GAIN DISJOINTNESS	375 - 380	Wavelet Transform		NMF
Chang, Jang, Iliopoulos	MUSIC GENRE CLASSIFICATION VIA COMPRESSIVE SAMPLING	387 - 392	Octave Subband STFT		Various Spectral Analysis inc. MFCC
Tjoa, Liu	MUSICAL INSTRUMENT RECOGNITION USING BIOLOGICALLY INSPIRED FILTERING OF TEMPORAL DICTIONARY ATOMS	435 - 441	STFT, Hamming Window	2048, 44.1Khz, 512 hop	NMF Variation
Dessein, Cont, Lemaitre	REAL-TIME POLYPHONIC MUSIC TRANSCRIPTION WITH NON-NEGATIVE MATRIX FACTORIZATION AND BETA-DIVERGENCE	489 - 494	STFT, Hamming Window	630 Data, 1024 FFT, 12.6kHz, 512 hop	NMF
Mak, Senapti, Yeung, Lam	Similarity Measures for Chinese Pop Music Based on Low-Level Audio Signal Attributes	512 - 518	STFT	2048	MFCC / Gaussian Mixture Model

Hsu, Jang,	SINGING PITCH EXTRACTION BY VOICE VIBRATO/TREMOLO ESTIMATION AND INSTRUMENT PARTIAL DELETION	525 - 531	MRFFT (Dressler 2006)	2048, 1024, 512, 256	Partial Trend Tracking
Gkiokas, Katsouros, Carayannis	TEMPO INDUCTION USING FILTERBANK ANALYSIS AND TONAL FEATURES	555 - 558	Mel Filter Bank		Convolution
Duggan, Shea	TUNEPAL - DISSEMINATING A MUSIC INFORMATION RETRIEVAL SYSTEM TO THE TRADITIONAL IRISH MUSIC COMMUNITY	583 - 588	FFT Hanning window	2048, 22.05kHz, 1024 hop	Klapuri Harmonic Analysis
Schuller, Kozielski, Weninger, Eyben, Rigoll	VOCALIST GENDER RECOGNITION IN RECORDED POPULAR MUSIC	613 - 618	DFT	50% overlap	NMF/Bayesian Networks
Paulus, Muller, Klapuri	AUDIO-BASED MUSIC STRUCTURE ANALYSIS	625 - 636	Discrete Cosine Transform		MFCC
Kelly, Gainza, Dorran, Coyle	LOCATING TUNE CHANGES AND PROVIDING A SEMANTIC LABELLING OF SETS OF IRISH TRADITIONAL TUNES	128 - 134	STFT		Chroma
Niedermayer, Widmer	A MULTI-PASS ALGORITHM FOR ACCURATE AUDIO-TO-SCORE ALIGNMENT	417 - 422	MRFFT	4096,1024,	NMF
Panagakis, Kotropoulos, Arce	SPARSE MULTI-LABEL LINEAR EMBEDDING WITHIN NONNEGATIVE TENSOR FACTORIZATION APPLIED TO MUSIC TAGGING	393 - 398	Wavelet Transform		NMF Variation

Table 3 – MIREX 2010 processing techniques

The STFT is popular front end to automatic music transcription systems, but the time frequency trade off remains as a compromise. An alternative method to a single resolution transform such as the STFT is the multiresolution transform.

The following section introduces the concept of multiresolution analysis and techniques for the purpose of automatic music transcription.

3 Low Level Processing - Multiresolution Analysis

Multiple resolution analysis for automatic music transcriptions consists of 2 main approaches, multiresolution in time, and multiresolution in frequency (Duxbury, Bello, Davies, & Sandler, A Comparison Between Fixed and Multiresolution Analysis for Onset Detection in Musical Signals, 2004).

Time varying multiresolution signal analysis is based on varying the analysis window used for Fourier transform based frequency estimation methods resulting in a variable time-frequency scale (Dressler, 2006).

The multiresolution in frequency approach comprises of splitting the frequency spectrum in to subbands and then analysis is performed on each separate band. This allows short analysis windows to be used at higher frequencies where the fast transients reside, while a longer window can be implemented for the lower frequencies resulting in frequency resolution adequate to separate closely space fundamentals.

The following section is an introduction to multiresolution analysis.

3.1 Approaches to Multiresolution Analysis

Broadly speaking, approaches to multiresolution analysis can be categorized into methods based upon modeling the critical bands of the human auditory system, and methods based upon a 'quality' factor, referred to as 'Constant Q', which is defined as the center frequency (Hz) divided by the bandwidth (Hz) (Diniz F. , Kothe, Netto, & Biscainho, 2007).

$$Q = \frac{Fc}{Bw}$$

Equation ii

Where:

Q	is the 'quality factor'
Fc	is the center frequency
Bw	is the bandwidth

As the center frequency of each band increases, so too does the bandwidth, therefore maintaining a constant quality factor. The human auditory system reflects an approximately constant Q frequency resolution in its critical bands (Garas & Sommen, 1998), but it is convenient for this thesis to categorize approaches as those that aim to achieve auditory functionality, and those that aim to achieve constant Q functionality.

The concept of constant Q is significant and important for automatic music transcription as it reflects the logarithmic nature of music and harmonics.

The constant Q transform (CQT) refers to any method of generating a time frequency representation where the frequency bands or bins are geometrically spaced and the Q factors of all bands/bins are equal (Schorkhuber & Klapuri, 2010). A constant Q transform results in the frequency resolution being improved in the low frequency ranges compared to higher frequencies – reflecting the logarithmic nature of western music.

The following sections present some common methods and approaches for multiresolution analysis. First the auditory system based methods are introduced, followed by constant Q approaches.

3.2 Auditory System Methods Filter Banks

A filter bank is an array of band pass filters that separate the original signal into multiple frequency bands (Roads, 1996, p. 193). The output of each filter is a sub-band containing the frequencies determined by the parameters of the filters used.

The center frequencies of the filters used for the lower frequencies can be closer together than the filters used for the higher frequencies. The arrangement results in a finer frequency resolution in the low frequencies where note fundamentals are closer together.

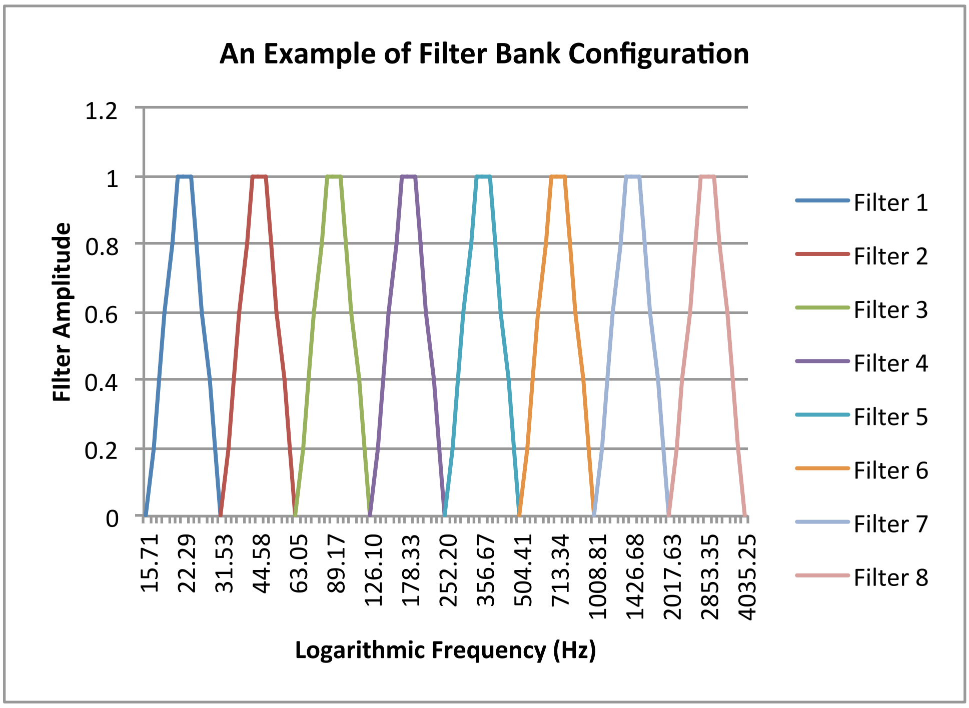


Figure 3-1 Filter bank

3.2.1 Auditory Filter Banks

Filterbanks are popular for auditory system approaches to pitch detection due to their behavior being similar to the cochlea (section 1.4). A typical model uses about 100 filters (Klapuri, Signal Processing Methods for Music Transcription, 2006b), with their center frequencies uniformly distributed along the logarithmic frequency scale, but various configurations of filters are used for multiresolution analysis of music signals.

3.2.2 Mel Filter Banks

Mel filter banks consist of filters with triangular magnitude response whose bandwidths reflect the Mel scale. Stevens, Volkman and Newman are credited with the *Mel scale,* which is a scale of pitches as perceived by humans (Stevens, Volkman, & Newman, 1937). The scale reflects the increasingly larger intervals above 500Hz judged by humans to produce equal pitch increments, implying humans have less resolution at high frequencies, and finer resolution at lower frequencies.

The spacing of the bands gives the Mel filter bank it's multiresolution property (Figure 3-2).

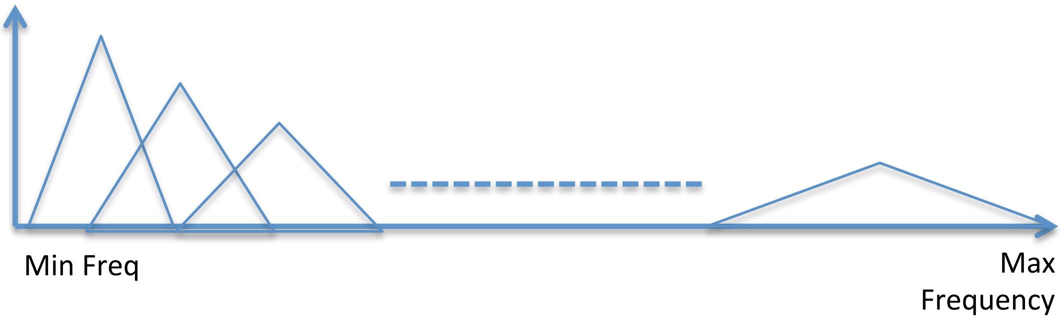


Figure 3-2 Mel filter bank

Uchida and Wada successfully use Mel filters to identify pitched instruments by comparing the output of the Mel filter bank to a trained database of sample instruments and pitches (Uchida & Wada, 2010).

3.2.3 Bark Scale

The Bark scale was proposed by Eberhard Zwicker in 1961 and is closely related to the Mel scale. The bark scale ranges from 1 to 24, where each point on the scale represents one of the first 24 critical bands of hearing, for 20Hz to 15.5kHz (Figure 3-3).

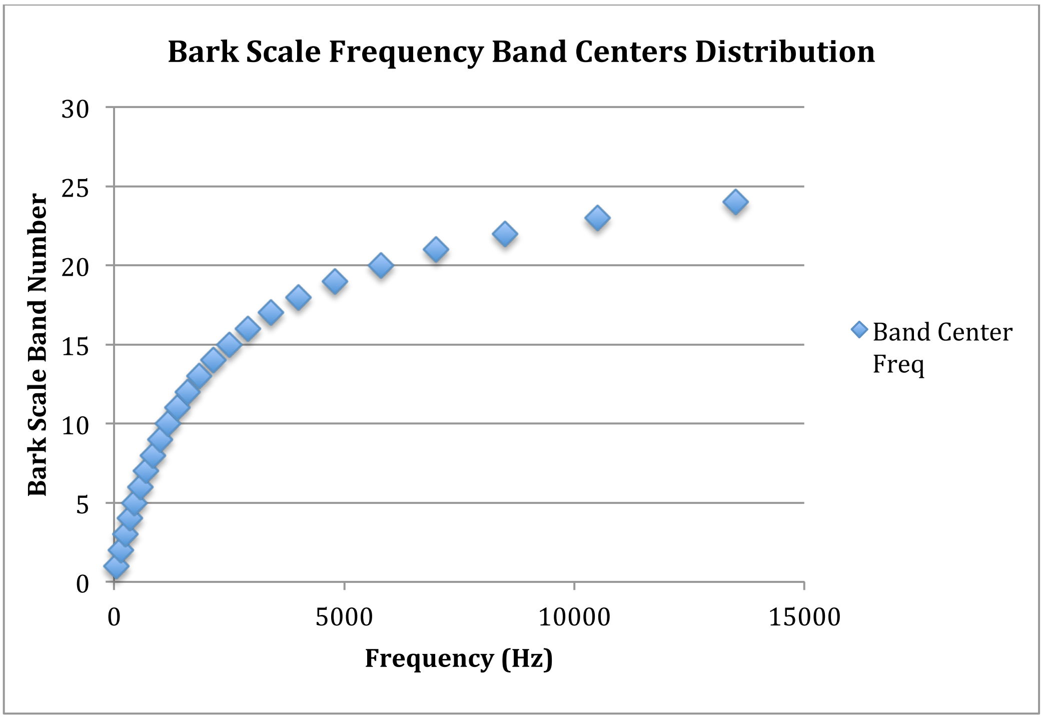


Figure 3-3 Bark scale frequencies

Shannon and Paliwal state in their investigation that despite the popularity of the Mel scale, there is little difference between that and the bark scale (Shannon & Paliwal, 2003). Indeed, Dressler uses the Bark scale rather than the Mel scale to determine frequency cut off points for his multiresolution sinusoidal analysis (Dressler, 2006).

3.2.4 Gammatone Filter Bank

The gammatone filter was introduced by Johannesma to imitate the filtering performed by the human ear by recreating the impulse response of the auditory system ((Johannesma, 1972) cited by (Lyon, Katsiamis, & Drakakis, 2010)) and has been popular for auditory modeling systems. This is mainly due to its simplicity (Lyon, Katsiamis, & Drakakis, 2010) and accuracy in imitating the filtering performed by the human ear (Ellis, 2009).

Although the filter models the auditory system impulse response, the filter itself only represents a single band, so for a full auditory system model it is implemented as a bank of gammatone filters.

Klapuri, building on work from Patterson et al. implemented a bank of gammatone filters for a perceptually motivated fundamental frequency estimation system (Patterson, Nimmo-Smith, Holdsworth, & Rice, 1987) (Klapuri, 2005).

Klapuri defines the bandwidth of 72 gammatone filters along the critical bands between 60Hz and 5.2kHz. This low level processing provides the initial spectral information Klapuri uses for an iterative harmonic detection and elimination method of fundamental pitch estimation. In testing, the method proved to be efficient and out performed its competitors in multiple fundamental frequency estimation tasks (Tolonen & Karjalainen, 2000).

The implementation of filters in the time domain for auditory modeling is a significant topic (Lyon, Katsiamis, & Drakakis, 2010), but the filtering of audio signals for music transcription purposes is not restricted to auditory models.

The following section presents constant Q motivated approaches to multiresolution frequency analysis.

3.3 Third Octave Banks

Third octave filter banks consist of a bank of filters that divide up each octave of the musical scale in to thirds. Each third of the octave is covered by a single band-pass filter, which results in a non-liner frequency resolution across the frequency spectrum.

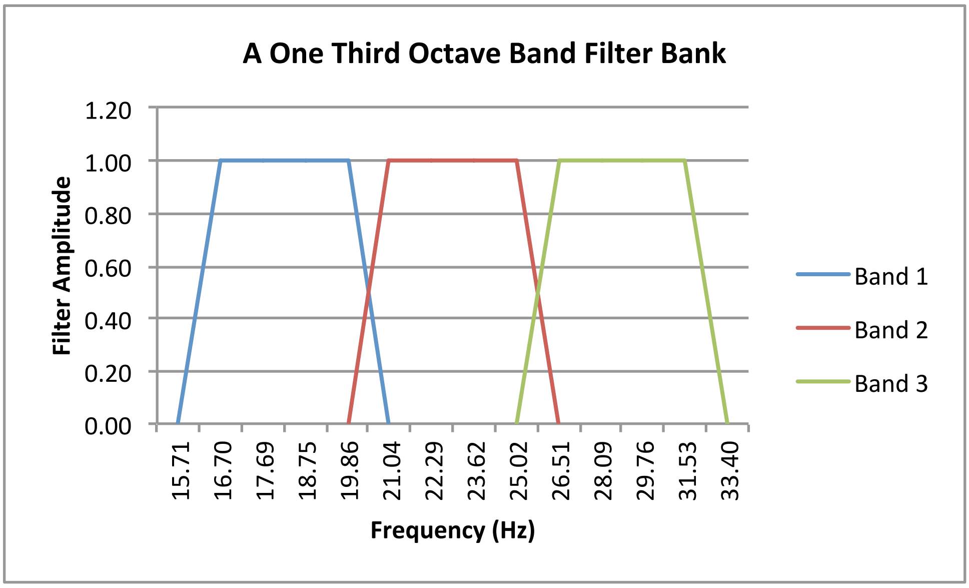


Figure 3-4 Third octave banks

Third octave filter banks are commonly used in graphic equalizers, but historically they have also been popular in auditory system modeling (Barabell & Crochiere, 1979) as the bandwidths of the filters approximately represent the bandwidths of the human auditory system (Cassidy & Smith, 2008).

The result of the one-third sub division of the frequency spectrum is what is referred to as a constant Q transform.

Pertusa et al. implement a one-semitone band pass filter bank on the output of a STFT (Pertusa & Inesta, 2009). The one semi tone filter bank is *tuned* to the western scale with the center frequency of each band pass corresponding to a note in the musical scale. The tuning of the filters creates strong note candidates for the onset detection and peak picking algorithms.

3.4 The STFT as a Filter Bank

The STFT so far has been viewed as a *windowed DFT representation* but the STFT can also be viewed as a *filter bank representation* (Smith J. O., 2010). By rearranging the STFT equation the output of the STFT can be interpreted as a frequency-ordered collection of narrow band time domain signals. This variation in the decomposition method of the FFT results in the input signal being converted to a set of N time-domain output signals, one corresponding to each bin (or channel) of the STFT (or filter bank) (Roads, 1996, p. 1096) (Figure 3-5).

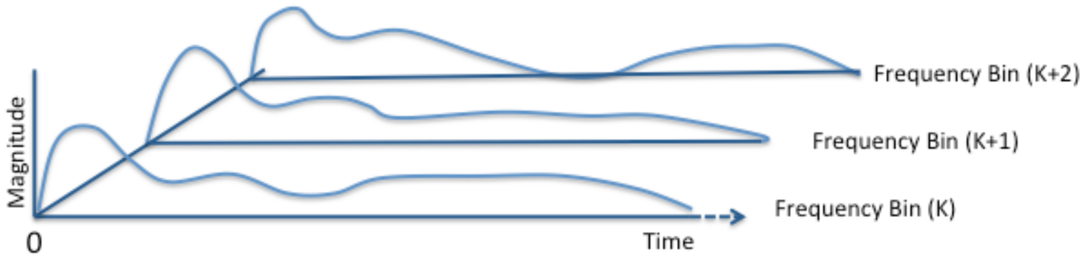


Figure 3-5 STFT Filter bank

The time frequency resolution of this basic STFT filter bank is still linear, but can be implemented as a part of a constant Q system.

3.5 The Constant Q Fast Filter Bank

The constant Q fast filter bank (CQFFB) as proposed by Diniz et al. is an attempt to utilise the speed and selectivity of the FFT based filter banks, with the ideal constant Q properties previously described, but without the computational overhead of other constant Q transforms (Diniz F. , Kothe, Netto, & Biscainho, 2007).

The CQFFB is based on the Fast Filter Bank (FFB) as proposed by Lim et al. (Lim & Farhang-Boroujeny, 1992). The FFB takes advantage of the tree structure of the FFT, but modifies the 'butterfly' to increase the selectivity of the channels in the frequency domain. By implementing filters in the FFT decomposition with very steep pass band-stop band transitions the FFB decreases any interference between adjacent bins, thus presenting strong maxima in the bins, from which note candidates can be more easily 'peak picked'. Although this increases the computation time of the FFT, it is still relatively efficient.

The design of the steep band filters follows the Frequency Response Masking Method (FRM) (Lim, 1986), which results in a highly optimised, low complexity filter. The FRM generated filters are generated across the FFT structure in a formation that results in each interpolated filter being masked by subsequent filters in the cascade. This is the FFB (Lim & Farhang-Boroujeny, 1992).

Although the original FFB implementation was highly selective and efficient, it still suffered from linear bin alignment in the same way as the STFT filter.

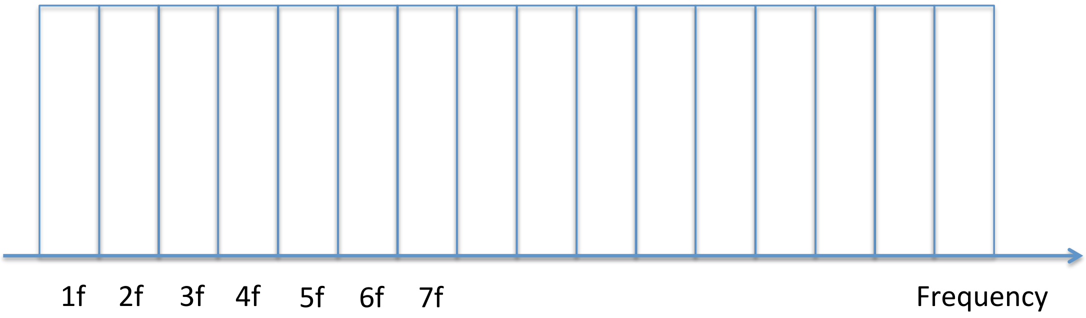


Figure 3-6 FFB Frequency bin spacing

Filipe et al. introduced a Bounded Q Fast Filter Bank (BQFFB) to improve the spectral analysis of the FFB for the musical context (Filipe, Diniz, Luiz, Biscainho, & Netto, 2006). Instead of calculating all bands linearly, the BQFFB logarithmically spaced the octaves, but inside each octave the channels were linearly distributed.

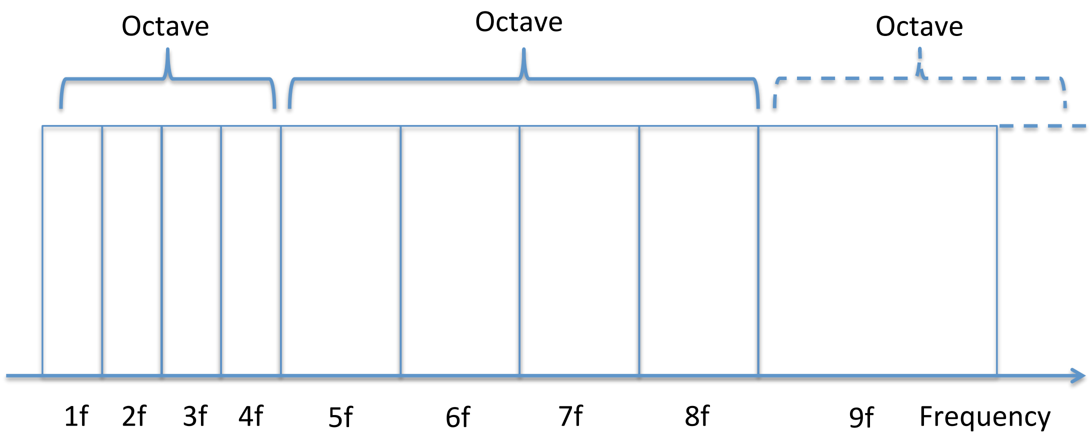


Figure 3-7 BQFFB Bin spacing

The BQFFB improved the performance of the FFB for analyzing music signals, but at the time the inefficiency meant a truly constant Q implementation wasn't practical.

The CQFFB demonstrated a computationally expensive, although still practical way of distributing the bands of the FFB geometrically across the entire spectrum, not just the octave bands.

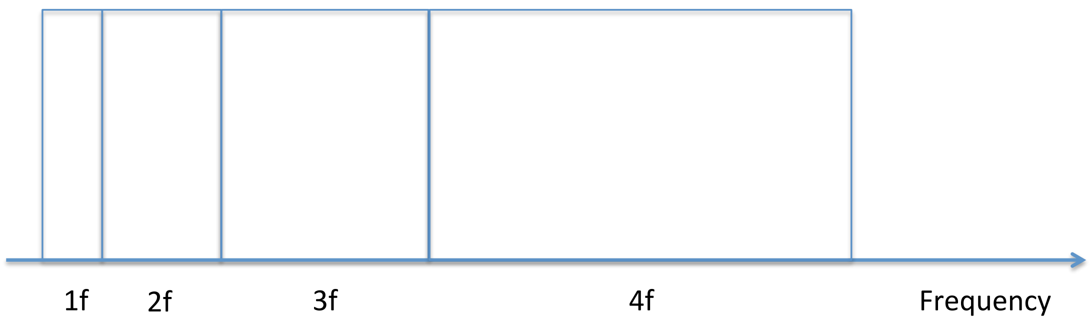


Figure 3-8 CQFFB Bin spacing

The implementation of the FFB combined with the constant Q spacing improved the distinction of maxima in the produced spectrum on tests with sinusoidal inputs. This property suggests the CQFFB is a useful tool for automatic music transcription, but the current implementation is relatively computationally expensive. Also several approaches for automatic music transcription require a signal to be transformed back to the time domain from the frequency domain, which the FFT and FFB are capable of doing but the CQFFB and BQFFB are not. This is possibly the reason why this method has so far not been adopted for automatic music transcription.

3.6 Other FFT and Filter Based Multiresolution Techniques

Zhou's Resonant Time Frequency Image (RTFI) (Zhou R. , 2006) method for frequency analysis uses down sampling in the fast multiresolution implementation of the RTFI (Zhou, Reiss, Mattavelli, & Zoia, 2009), implementing a cascading filter bank similar to Goto (Goto, 2002). The RTFI is becoming more popular due to its constant Q properties and flexibility (Benetos & Dixon, 2011).

Cancela et al. used an Infinite Impulse Response (IIR) filter on the output of an FFT in a simple but effective algorithm for multiresolution analysis (Cancela, Rocamora, & Lopez, 2009), which was used on the best 'Overall Accuracy' algorithm for the Audio Melody Extraction exercise at Mirex 2008 (Durrieu, Gael, & Bertrand, 2008).

Smith presents an approach to designing and efficiently implementing non-linear FFT filter banks that approximately matches the constant Q form (Smith J. O., 2009). By performing smaller inverse FFTs on each band of an FFT output, Smith synthesized the down sampling of the time domain signals in each band, thus resulting in a non-linear time-frequency scale. The concept of down sampling is introduced in the following section.

3.7 Multirate Filter Banks

Multirate filter banks use different sample rates for different bands, which are matched to different filter bandwidths to generate varying time-frequency resolutions across the spectrum.

The process of down sampling is to retain every *Mth* sample of a signal *x(n)* relabeling the index axis accordingly. The compression of time explicit in this process is accompanied by a stretching in the frequency domain (Akansu & Haddad, 1992), hence a non-linear time frequency resolution.

The down sampling of a digital signal when combined with low pass filtering is known as *Decimation.* The decimation functions in stages where the top half of the frequency spectrum is output as an audio band, and the bottom half of the frequency spectrum is down-sampled. The process is then repeated on the decimated audio. The basic process is shown in Figure 3-9

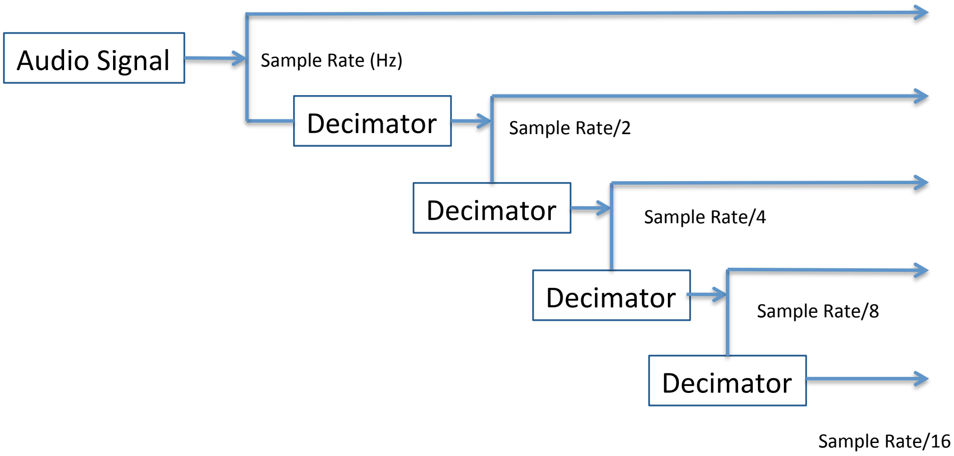


Figure 3-9 Decimation

It is not immediately obvious how this process varies the time frequency resolution, but applying the concept of reducing sample rates to the FFT decomposition, it is clear to see that varying the sample rate will affect the time resolution and the frequency resolution. Table 4 is an example of the effect of varying the sample rate for a 1024 sample FFT across different bands, resulting in a variable time frequency resolution.

Bottom of Frequency Band (Hz)	Top of frequency bands (Hz)	Sample Rate	Time Res of 1024 Window (s)	Frequency Resolution (Hz)
689.06	1378.13	2756.25	0.372	2.69
1378.13	2756.25	5512.5	0.186	5.38
2756.25	5512.50	11025	0.093	10.77
5512.50	11025.00	22050	0.046	21.53
11025.00	22050.00	44100	0.023	43.07

Table 4 FFT Multirate resolutions

Figure 3-10 below shows the time-frequency plane diagram for the first four bands of the multirate example in Table 4 to clearly show the varying time and frequency resolution across the different frequency bands of the original signal.

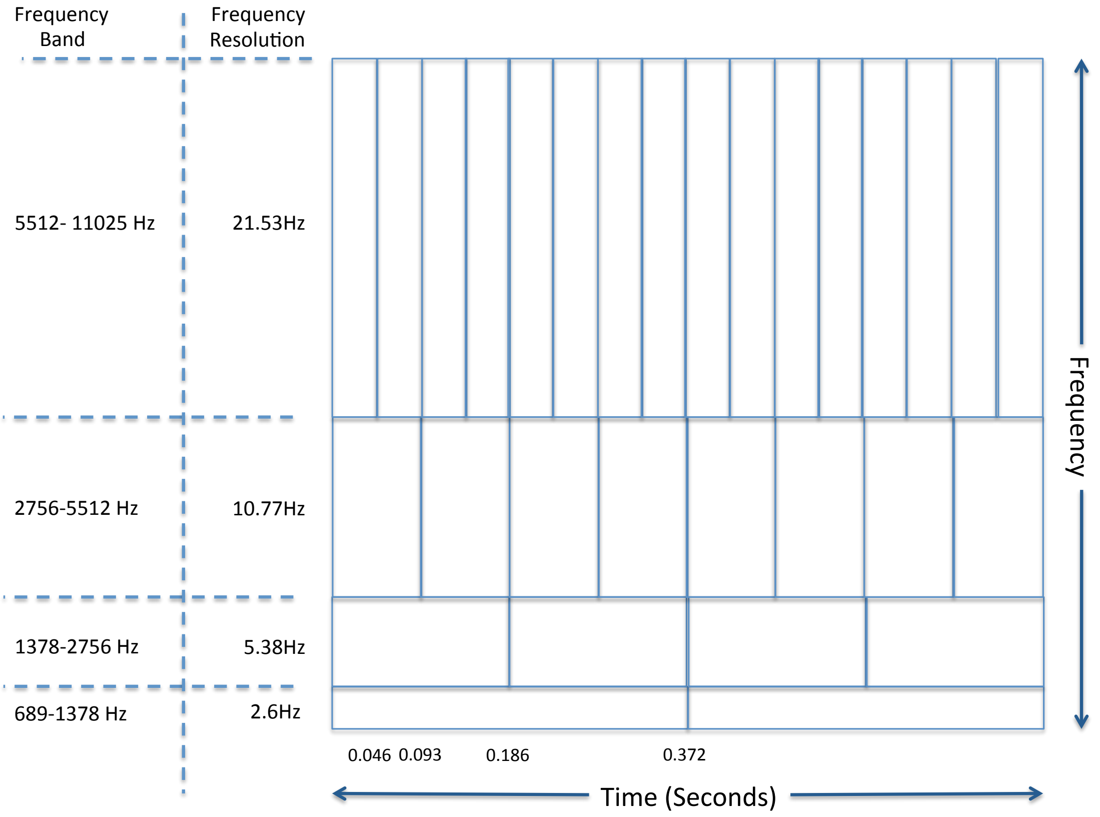


Figure 3-10 Multirate FFT plane diagram

Considering the frequency spacing of the equal tempered scale and the pattern of the time frequency resolution plane in Figure 3-10 it is clear that this method of multiresolution decomposition is suited to music transcription. The lower frequencies where note fundamentals are closer together are in a band that has a higher frequency resolution than the highest band where fundamentals are further apart. Also, high frequency notes tend to have faster rates of change than lower notes, which is reflected in the relevant time resolution of the frequency bands.

Goto implements a multirate filter bank as the low level processing in his PreFest algorithm (Goto, 2000). PreFest was the first algorithm to successfully prove the transcription of polyphonic music from a commercial CD is possible (Goto, 2000) by accurately estimating the melody and bass line note fundamentals.

Goto implements a multirate filter bank to obtain an adequate time frequency resolution. The multirate filter banks also allow for real time processing by keeping the computational load relatively low (Goto, Music Scene Description, 2006, p. 332).

Goto starts with a 16kHz sampled audio signal, which is decimated in 4 steps to a 1kHz sampled signal. The decimation stage consists of a low pass filter with a cut off frequency of 0.45 of the sampling frequency of that 'branch' of processing, and then half down sampled. An STFT is then performed on each frequency band. This process is shown in Figure 3-11.

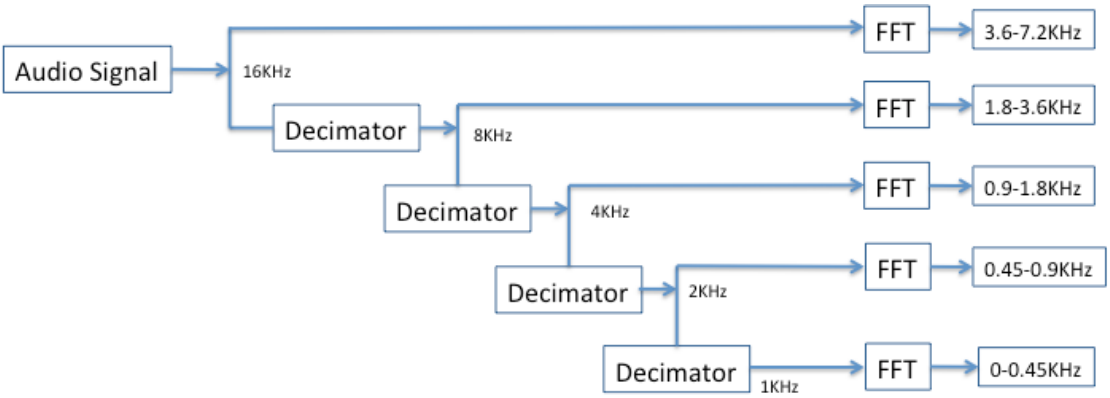


Figure 3-11 Goto implementation

The multirate filter as used by Goto is very similar in its construction and resulting time-frequency plain as the Discrete Wavelet Transform.

3.8 The Discrete Wavelet Transform

The discrete wavelet transform (DWT) is a constant Q transform that uses cascading pairs of high pass and low pass filters to decompose a signal to a time-frequency spectrum.

A single level of DWT consists of the signal to be analysed being filtered through a high pass and low pass filter simultaneously. A quadrature mirror filter is used for this process, which splits the signal in to two bands where each filter is subsampled by 2 at the output. (Mallat, 2009). This decomposition halves the time resolution, but as each output has either the high frequency band or low frequency band of the input signal, the frequency resolution has been doubled. This process is repeated in a cascading formation (Figure 3-12).

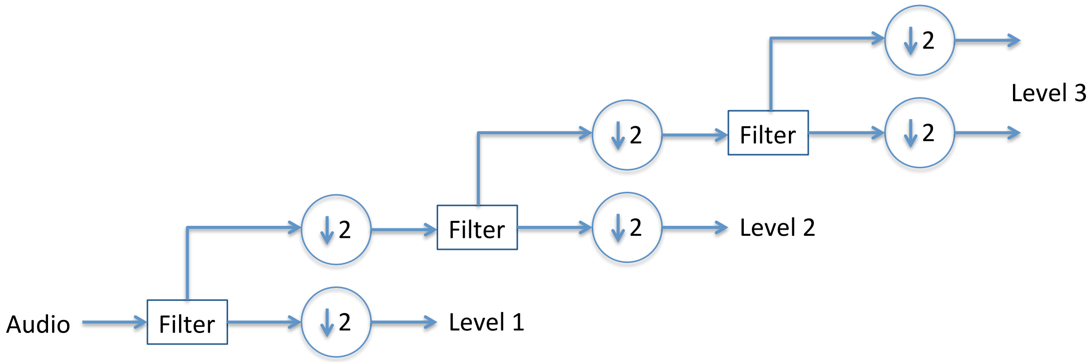


Figure 3-12 DWT down-sampling

The resultant frequency spectrum is shown in Figure 3-13.

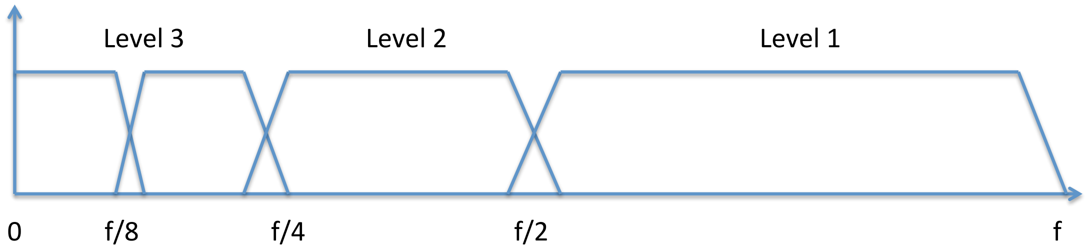


Figure 3-13 DWT frequency division

And the resultant time-frequency plane is shown in Figure 3-14.

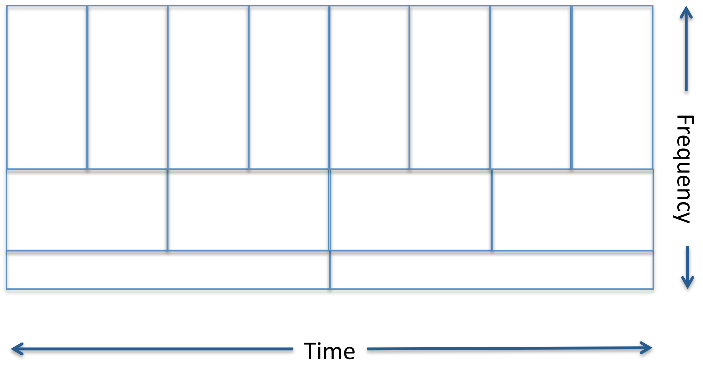


Figure 3-14 DWT plane diagram

Wavelet transforms differ from filter banks in that the half band filters always create a true pyramid structure in the time-frequency plane, whereas filter banks do not necessarily result in a pyramid structure (Humphrey, 2010). However, the similarity between the DWT and the multirate filterbank decomposition as used by Goto (Goto, Music Scene Description, 2006) is clear.

Wavelet analysis has been used for automatic music transcription and pitch analysis as the constant Q properties are ideal (Yegnanarayana & Murty, 2009), but it is not a popular choice in the fundamental frequency estimation discipline of music transcription. Analysis of the submissions to Mirex 2010 show only one competitor used the wavelet transform method for multipitch analysis (Yoshii & Goto, 2010). A significant reason for the unpopularity of the DWT is the Q factors currently required for multiple fundamental frequency estimation can be equivalent to up to 96 bands/bins per octave (Schorkhuber & Klapuri, 2010). To create this resolution using a wavelet transform requires filtering the input signal hundreds of times, thus making it highly inefficient and computationally expensive, particularly compared to the, albeit non constant Q, but very fast and efficient STFT (Schorkhuber & Klapuri, 2010).

3.9 Variable and Multiple Window STFT Representation

An alternative method to FFT filter based multiresolution analysis is the variable and multiple window technique. The size of the analysis widow used in an FFT directly affects the time-frequency resolution, so the adjusting in length and combining these windows results in a non-linear time-frequency response.

The process as implemented by Anderson (Anderson, 1996) and later by Tyagi and Bourland (Tyagi & Bourland, 2003) involves taking multiple sliding FFTs of varying window lengths of the same input data. The long windowed high frequency resolution FFT is used for low frequency analysis, the short windowed, low frequency resolution used for high frequency analysis. The use of multiple window sizes for multiresolution Fourier transforms was adopted by Brown and Puckette as part of a constant Q transform (Brown & Puckette, 1992), and Keren et al. also used varying window lengths in their low level processing for transcribing piano music (Keren, Zeevi, & Chazan, 1998).

Godwin developed the idea of variable window lengths by implementing a dynamically changing window length based upon the transient activity of the signal, providing greater time resolution in areas of high energy activity (Godwin, 1997).

Djurovic and Stankovic further developed an adaptive window for multiresolution analysis by calculating the optimal window width for an STFT reliant upon the Bias-Variance of the FFT and the Mean Square Error (MSE) (Djurovic & Stankovic, 2003).

The MSE is a method of quantifying the difference between implied values generated by an estimator (θ*) and the actual values being estimated (θ). The MSE is calculated as:

$$MSE = Var(\theta *) + Bias(\theta *)^2$$

Equation iii

Although the lack of bias of a system is attractive based on the above formula, if the bias can be increased to minimize the variance in the system, then the error rate can be reduced and accuracy improved.

In the context of the FFT for fundamental frequency estimation, the MSE of the system is a measure of the FFT's accuracy of measuring a frequency within an audio signal (Djurovic & Stankovic, 2003). The variance of the system is the probabilistic distribution of the frequencies within the signal (see section 5.1). The optimal window width for the FFT is derived by Papoulis (Papoulis, 1977), but cannot be used practically as the bias of the FFT relies on unknown behaviors dependent on the input signal (Djurovic & Stankovic, 2003).

Djurovic and Stankovic develop a method to calculate the optimal window width that can be implemented based on the intersection of the confidence intervals (ICI) rule (Djurovic & Stankovic, 2003), as introduced by Goldenshluger and Nemirovski (Goldenshluger & Nemirovski, 1997).

Although the ICI is mathematically involved, Katkovnik et al. explain the rule in the context of window sizes (Katkovnik, Egiazarian, & Shmulevich, 2001). For a finite set of window sizes, the bias is proportional to the window size. A confidence interval is calculated for the bias resulting from each window size, forming a sequence of confidence intervals. Considering the sequence of confidence intervals, there will be a common point of intersection of the intervals, from which the optimal adaptive window is calculated (Katkovnik, Egiazarian, & Shmulevich, 2001).

Djurovic and Stankovic include a probability parameter in the calculation of the confidence interval sequence, which determines the algorithm accuracy. The results of FFT analysis on a mixture of 3 sine waves show their optimised adaptive window out performs the minimum static window and maximum static window, generating stronger magnitudes in the FFT output with reduced noise (Djurovic & Stankovic, 2003).

Duxbury et al. suggested variable window analysis to be a redundant technique (Duxbury, Bello, Davies, & Sandler, A Comparison Between Fixed and Multiresolution Analysis for Onset Detection in Musical Signals, 2004), but research and methods are still being developed using variable windows. Benaroya et al implemented a tri-window multiresolution FFT (MRFFT) as a successful front end to a Bayesian high level process, improving results compared to a single window analysis (Benaroya, Blouet, Fevotte, & Cohen, 2006). Benaroya's et al. method implements cascading FFTs of different window lengths, each becoming shorter on each iteration.

Keren et al. Proposed a multi windowed FFT algorithm for polyphonic music transcription which demonstrated the usefulness of varying the frequency resolution to detect harmonics of piano notes (Keren, Zeevi, & Chazan, 1998) However, the process was computationally demanding and impractical for most real world applications.

Dressler described a very efficient implementation of a 4 windowed multiresolution STFT (Dressler, 2006), which is used as a benchmark for other FFT based multiresolution algorithms (Cancela, Rocamora, & Lopez, 2009).

3.10 Sinusoidal Extraction Using a Multiresolution FFT (MRFFT)

Dressler's MRFFT generates varying time and frequency resolutions by altering the data frame size only, leaving the hop size and window length constant. Four data lengths are used - 2048, 1024, 512 and 256 – all powers of 2 in length, resulting in a four layer MRFFT, but zero padding is used to maintain a constant window length of 2048.

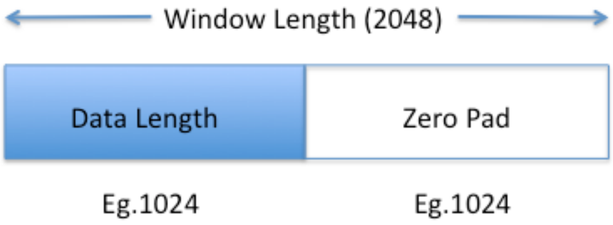


Figure 3-15 FFT data vs. window length

The time-frequency plane diagram of the MRFFT is dependent on sample rate and window length so remains constant in Dressler's implementation. However, as Dressler points out, the time-frequency resolution of the transform is not necessarily the time-frequency resolution of the resulting calculated as *sample rate/window size*, but actually, the true resolution of the transform is calculated as the *sample rate/data size*. This is discussed further in section 6.10. The sample rate/data size time frequency plane diagram as generated by Dressler takes on a familiar form (Figure 3-16)

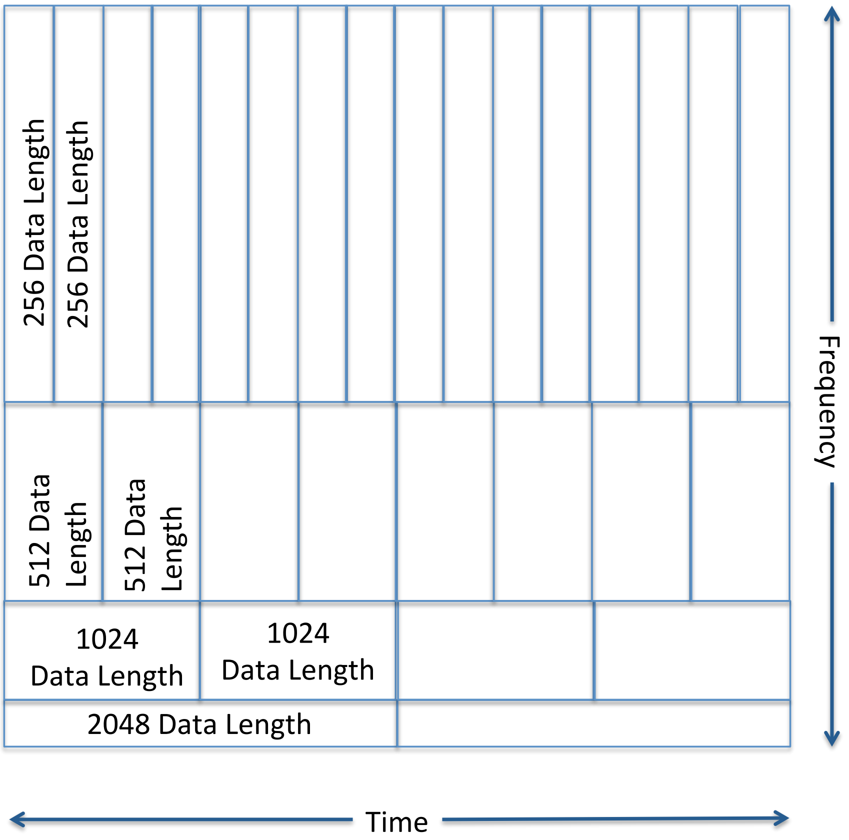


Figure 3-16 Dressler's plane diagram

As all transforms are the same length due to zero padding, the spectra of the 4 STFT windows can simply be summed. The magnitudes (resultant of the constant window size) of the summed spectra are valid for peak picking at this point, which demonstrates the simplicity of this process.

Wen and Sandler look to further improve the efficiency of MRFFT implementations by optimising a radix-2 FFT for multiresolution calculations by reducing the number of calculations required in the FFT decomposition by half by reusing internal results (Wen & Sandler, 2007).

Dresslers’ Mirex 2009 submission implemented the MRFFT as the low level processing stage, and performed with the highest overall accuracy for the Audio Melody Extraction test (Dressler, 2009).

4 Peak Picking and Spectral Processing

This section introduces methods for extracting frequencies from the frequency spectrum to present as note candidates to the high level processing.

Peak picking is the term given to the process of extracting the frequency associated with a maxima from a spectra representation such as a spectrogram, or more typically the output of a Fourier transform. Peak picking has an important role to play in spectral analysis for audio as it aims to select only peaks corresponding to genuine resonant components present in a signal (Nunes, Esquef, & Biscainho, 2007).

4.1 Threshold Based Peak Picking

The most basic peak picking method is to set a static threshold, and when the threshold is crossed by the magnitude of a frequency component, it is determined to be a note candidate. Using sine waves it is possible to gain acceptable results by using a very simple static threshold method, but in reality it is ineffective when dealing with the spectral complexities of 'real' audio.

Collins implemented a basic peak picker imitating how a human would visually 'peak pick' by comparing peaks to their nearby 'terrain'. Collins' peak picker scores the most salient peaks relative to their local 'terrain' – the current frame and 3 analysis frames either side (Collins, 2005). The spectral energy in the 7 frames is normalized to be between 0 and 1 and is then analysed. An empirically determined threshold of 0.34 is manually set to then extract any peaks. The manual setting of the threshold is typical for this kind of application (Duxbury, Bello, Davies, & Sandler, Complex Domain Onset Detection for Musical Signals, 2003).

Duxbury et al however, do state that manual setting of thresholds is not acceptable in all cases, for example a commercial product where the user should not be expected to set a threshold for each source. Thresholds are therefore set either globally, which is a computationally efficient method but more prone to errors by missing candidates in quiet passages, or over detecting in louder passages of music, or thresholds can be set locally by analyzing the spectral content on a frame by frame method to dynamically adjust the threshold. This, according to Duxbury et al is essential for effective onset detection.

Various methods of analyzing the frame content to determine the local threshold are used in peak picking algorithms (Nunes, Esquef, & Biscainho, 2007) but by its nature peak picking is prone to errors (Kumar, Jakhanwal, Bhowmick, & Chandra, 2011) and not robust enough to act as the only method to extract note candidates from a spectral analysis. The errors are due to threshold peak pickers relying solely on magnitude information, thus neglecting the detection of events without a strong energy increase e.g. low notes, transitions between harmonically related notes or onsets played by bowed instruments and due to the energy in the frequency domain attributed to features other than fundamental frequencies (Bello, Daudet, Abdallah, Duxbury, Davies, & Sandler, 2005), hence the need for further processing.

4.1.1 Phase Based Peak Pickers

Keiler and Marchand (Keiler & Marchand, 2002) suggest peak picking algorithms that also consider the phase of the signal are more accurate. By using phase changes in the spectral information, peak pickers can increase their accuracy of detecting low and high frequency tone changes regardless of their intensity. However, this approach is still not wholly robust as variations by the phases of noisy low energy spectra, and from phase distortions common in commercial post production effects and processes can cause errors (Bello, Daudet, Abdallah, Duxbury, Davies, & Sandler, 2005).

Betser et al. further discuss phase based frequency estimators for short time Fourier transforms, grouping algorithms in to 3 main types (Betser, Collen, Bertrand, & Gael, 2006) – Arccos estimator (Lagrange, 2004), Long Term Phase Vocoder (Puckette & Brown, 1998) and the Short Term Phase Vocoder (Flanagan & Golden, 1966) as used by Dressler (Dressler, 2006)

The accuracy of the peak picking algorithm can be improved by processing and enhancing the spectral information generated by the time-frequency transform.

4.2 Spectral Processing

The following section introduces some methods for processing the spectral results of a time to frequency transform to help improve the performance of a peak picking algorithm. These processing techniques aim to improve the 'quality' of the spectral representation by presenting 'strong', accurate spectral peaks to the peak picking algorithm.

4.2.1 Spectral autocorrelation

Autocorrelation can be used in the frequency domain as well as the time domain (discussed in 2.3) but suffers from the same limitations for multiple fundamental frequency estimation (Lahat, Niederjohn, & Krubsack, 1987). Spectral autocorrelation is the comparison of the spectrogram of a section of an audio signal with a spectrogram of an adjacent section. Frequency magnitudes reinforced by the addition of adjacent frame information are presented as 'stronger' note candidates. Although found to be quite accurate on single fundamental frequency estimation (Cheveigne & Kawahara, Comparative Evaluation of F0 Estimation Algorithms, 2001), spectral autocorrelation is not robust enough for the complexity of multiple fundamental frequency estimation (Klapuri, 2006b).

4.2.2 Spectral Compression

Harris developed early work on the identifying of fundamental frequencies based upon measuring the frequency intervals between potential harmonics (Harris & Weiss, 1963). Schroeder further developed this work by transposing spectral transients to lower frequencies to enhance potential fundamental frequencies (Schroeder, 1968). Spectral compression is used to generate the Schroeder histogram, which counts equally the contribution of each spectral peak to the related F0s that are common divisors of its frequency. Schroeder assigned the magnitude of higher frequency components to harmonically matching hypothetical fundamental frequencies on the spectrum This process focuses the energy of higher partials on distinct peaks, and the maximal peak determines the related F0 (Yeh, 2008, p. 12). Although this process is not robust against noise in the spectrum, Szczerba and Czyzewski successfully combined in part this method with prior music knowledge to help reduce errors made in pitch estimation (Szczerba & Czyzewski, 2005).

Klapuri developed the idea of spectral compression to create a computationally efficient fundamental frequency estimator for polyphonic music (Klapuri, 2006a). Klapuri's algorithm calculates the strength of a fundamental frequency candidate in the output of a Fourier transform as a weighted sum of the amplitudes of its harmonic partials. The accuracy of the system is improved through 'training' the algorithm with test data to increase accuracy when identifying harmonics, and also by a simple method of cancelling a confirmed fundamental frequency and associated harmonics from the mixture, thus simplifying the spectrum for further analysis. The utilisation of the information provided by the presence of partials in a polyphonic mixture is directly related to Harris's early work on spectral peak interspacing.

4.2.3 Harmonic Matching

Harmonic matching is the process of matching a known harmonic spectral pattern to an observed spectrum. This is performed by using either a specific spectral model or by a harmonic comb, which is a series of spectral pulses with equal spacing defined by a hypothetical fundamental frequency (Yeh, 2008). The purpose of the harmonic comb is to emphasize the energy in the observed spectrum at the expected ideal harmonic locations, thus making any harmonic energy clearer in the associated spectrogram.

Rao and Rao's submission to MIREX 2008 successfully developed the harmonic matching theory and used a 'two way mismatch' method (Rao & Rao, 2008). Two way mismatch minimizes a spectral mismatch error that is the result of a particular combination of energy at the partial and it's frequency deviation from the ideal harmonic location. Rao & Rao's algorithm, which uses an FFT for its low level processing, is particularly robust for sparse but strong harmonic interference in comparisons to other harmonic matching pitch detection algorithms. (Rao & Rao, 2008).

4.2.4 Spectral Tilt Compensation

Audio will typical exhibit a spectral pattern whose energy decreases with frequency (Grey & Gordon, 1978) and as a result low energy peaks in the high frequency range, which may actually correspond to a note, may be discarded (Nunes, Esquef, & Biscainho, 2007) (Figure 4-1)

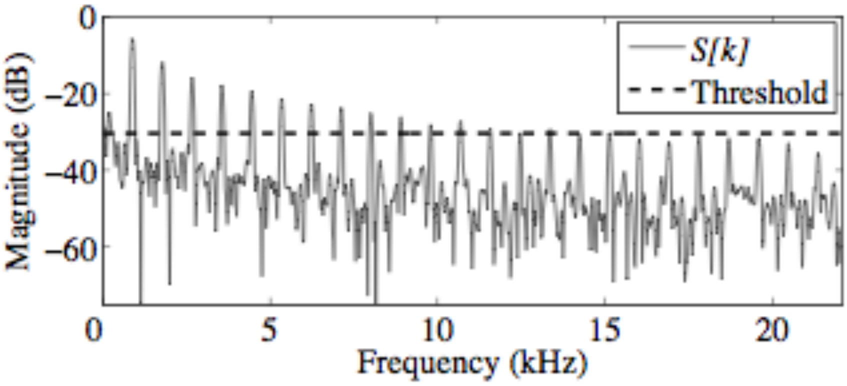


Figure 4-1 Spectral pattern with decreasing energy

Image from (Nunes, Esquef, & Biscainho, 2007, p. 3)

Nunes et al. evaluate methods of adjusting the spectral information to compensate for the changes in partial maxima based upon their frequency – *Spectral Tilt Compensation.*

In simplistic terms, a spectral tilt estimator calculates as accurately as possible the spectral profile of the signal and uses this to adjust the spectrum so a constant threshold can be used (Figure 4-2).

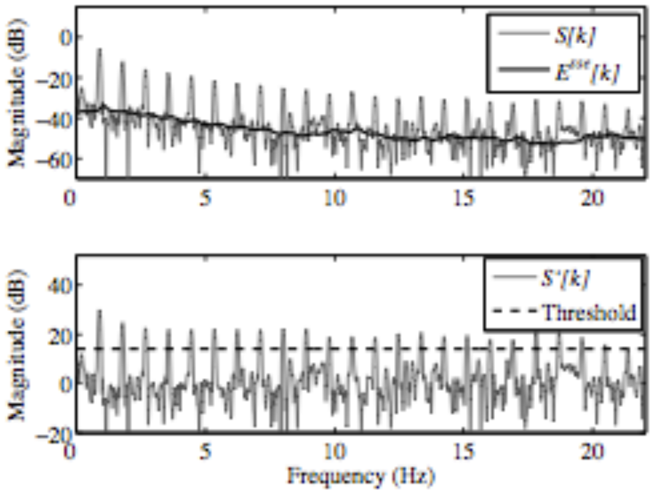


Figure 4-2 A comparison between the non tilted and tilted spectra

Image from (Nunes, Esquef, & Biscainho, 2007, p. 5)

One method suggested by Nunes to accomplish this is ‘Stochastic Spectrum Estimation’ (SSE), as introduced by Laurenti, Poli and Montagner (Laurenti, Poli, & Montagner, 2007). Laurenti et al are primarily concerned with modeling musical sounds and therefore separating the sinusoids, transients and noise that are the components of a musical instrument. To separate the ‘noise’ components Laurenti et al estimate the spectral envelope by calculating the energy of the signal in the frequency domain over successive sliding windows.

The magnitude output of a DFT of the signal is passed through a filter to remove any null magnitude samples. The reciprocal of this filtered spectrum is calculated and then smoothed. The estimated envelope is then calculated as being the reciprocal of the smoothed signal (Figure 4-3).

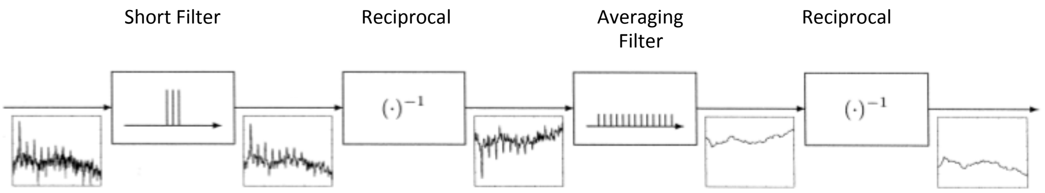


Figure 4-3 Diagram of the stochastic spectrum estimation method

In Nune's test system, the performance of the peak picker working on the SSE spectral tilted audio worked to an 88.5% success rate, compared to a 38.2% success rate for the same peak picker working on non processed spectral information. Although these results only relate to spectral tilt processing, it does demonstrate the significant improvement to peak picking performance spectral processing can make.

The output of a peak picking algorithm is a series of note candidates, which are presented to the high level processes to process. Although the scope of this thesis does not allow detailed discussion of high level processes, it is good to be aware of some of the more commonly used techniques used to put into context the low level processing discussed later.

5 High Level Processing Summary

The purpose of the high level processing in multiple fundamental frequency estimation systems is to survey the hypothetical notes presented by the low level processing and prune as accurately as possible the note candidates presented in error from those which are correct.

The following is an introduction to some of the common high level concepts and processes implemented in multiple fundamental frequency estimation systems.

5.1 Probability Density Function

High level processes often rely on prior information to determine the likelihood of an event occurring. The 'events' in F0 estimation will normally be note pitches and/or note intervals. The probability of these events occurring is characterized by the *probability distribution* selected for the algorithm (Roads, 1996, p. 896).

Probability density functions provide the *prior distribution* information for Bayesian statistical methods (Davy, An Introduction to Signal Processing, 2006a, p. 40)

5.2 The Bayesian Model

Rules of scales, pitches, intervals and harmonics etc. can be used to help understand, and extract information from a complex waveform (Davy, 2006b, p. 205).

As Davey explains "*This structure of tonal music can be utilised to build a Bayesian model, which is a mathematical model embedded into a probabilistic framework that leads to the simplest model that explains a waveform"* (Davy, 2006b, p. 205).

A Bayesian network is a graphical interpretation of probability that represents a series of random variables and their inter-dependency. The nodes of a Bayesian network correspond to random variables, such as note candidates, and the links between the nodes encode probabilistic dependencies between the corresponding random variables (Kashino, 2006, p. 313) i.e. the probability a note transition will occur based upon prior distribution. The direction of the arrow denotes the direction of probabilistic dependency from the origin of the arrow (the 'parent'), to the end point (the 'child').

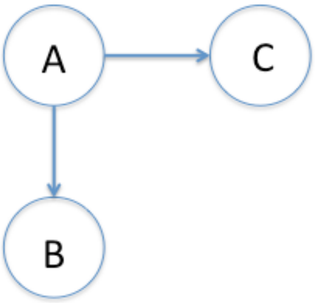


Figure 5-1 Bayesian nodes

In Figure 5-1the node labeled A is the parent to the child nodes labeled 'B' and 'C'. The arrows connecting the child nodes to the parent represent the probability of the variable B or C occurring when the current observed state is A.

Kashino implements a Bayesian network for music transcription in his Organised Processing Toward Integrated Music Scene Analysis (OPTIMA) system (Kashino, 2006, p. 318). Kashino performs both a 'bottom up' approached based upon analyzing the note candidates and applying them to a model, and also a top down approach where a chord hypothesis is applied and the note candidates are analysed based upon this paradigm.

Figure 5-2 shows the overview of the main process of the OPTIMA system.

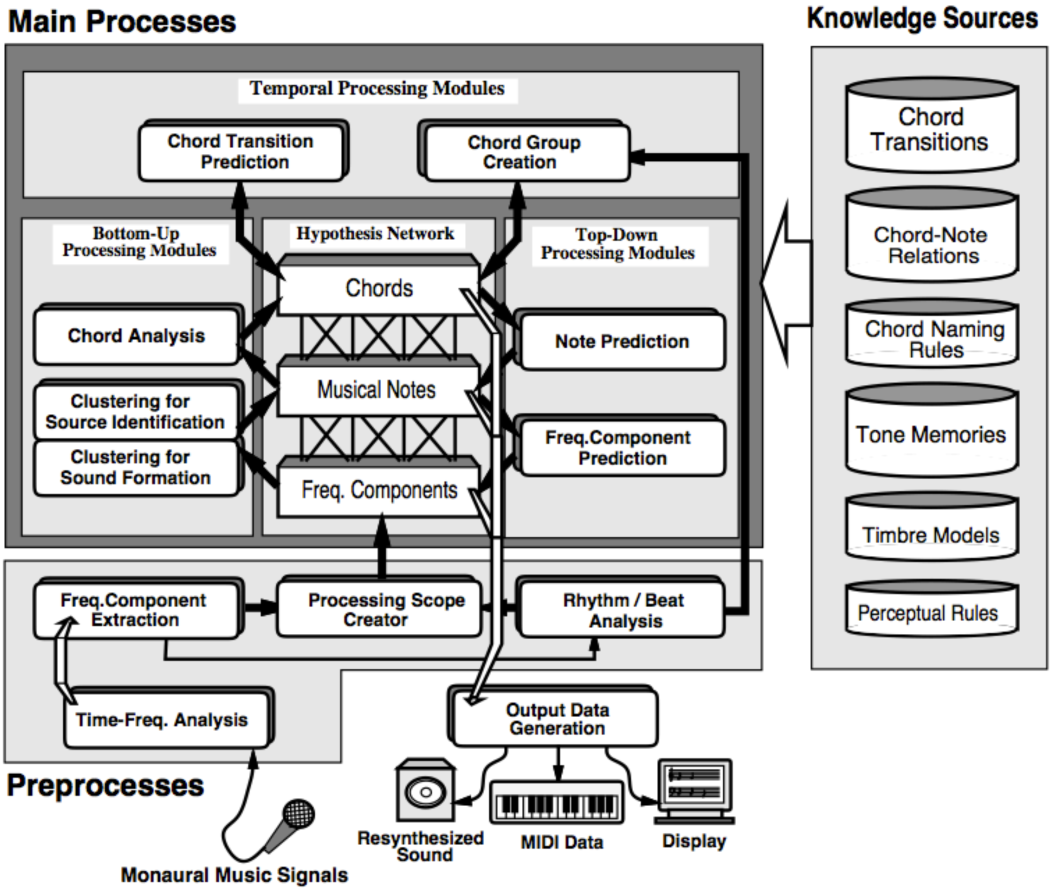


Figure 5-2 OPTIMA System overview

Image source: (Kashino, 2006, p. 318)

The 'preprocesses' stage is the initial time to frequency domain transform, spectral processing and peak picking procedures. The 'main processes' show the bottom up and top down approaches using a 'hypothesis network' based upon pdfs for chords, musical notes and frequency components. The 'Knowledge sources' are the data used to 'train the system' and generate the pdfs.

Models such as Bayesian are important as they have the potential to provide information about the source of the signal, without having the source available. A member of the Bayesian process family is the Hidden Markov Model.

5.3 Hidden Markov Model

Markov models consist of 'states', which are similar to the nodes of a Bayesian network. The states are what describe the signal. (Rabiner, 1989).

The Hidden Markov Model (HMM) has 2 defining features:

1) The HMM assumes the observation at time *t* was generated by a process whose state was hidden from the observer. For example, a frequency may be observed, but the instrument that generated it may be unknown.

2) The HMM assumes the state of the hidden process satisfies the Markov property.

(Ghahramani, 2001)

The Markov property is: given the value of the previous state, the current state is independent of all states before that and encapsulates all information about the history of the state to be able to predict the future of the process. (Ghahramani, 2001, p. 2)

For AMT, the 'left-right' HMM is particularly useful as it has the property that as time increases the state index increases (or stays the same). This is a desirable property to model signals that change over time (Rabiner, 1989).

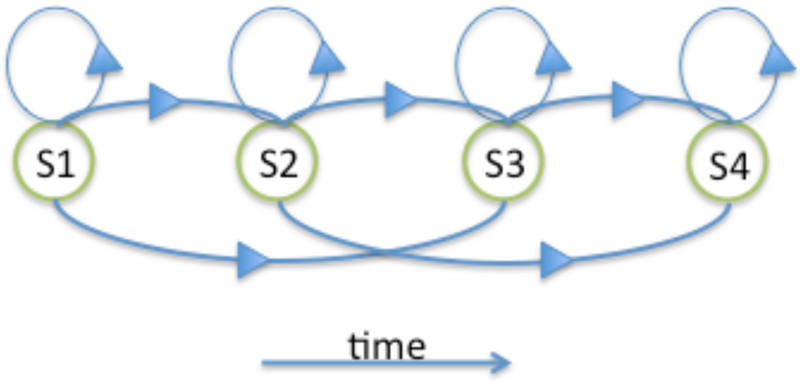


Figure 5-3 A left-right HMM with 4 states

Ryynanen uses two left-right HMM models, a note event model to indicate the probability of a note occurring, and a musicological model to determine the probability of transitions between the notes in a singing transcription system (Ryynanen & Klapuri, 2004). An over view of the system is shown in Figure 5-4.

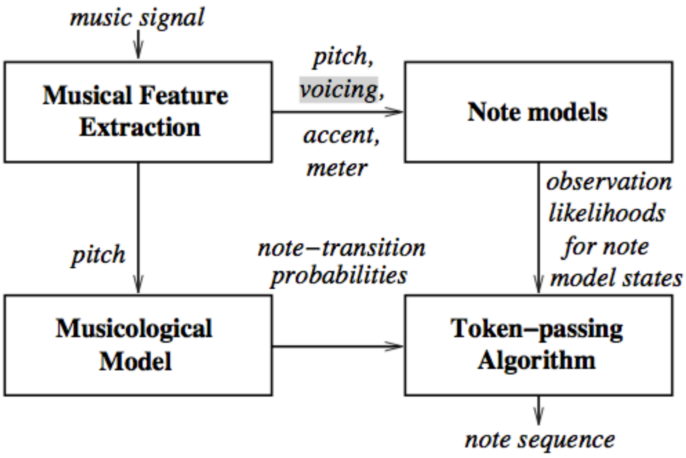


Figure 5-4 Block diagram of Ryynanen and Klapuri system.

Image taken from (Ryynanen & Klapuri, 2004)

The model uses 4 features extracted by the low level processing to determine a note candidate, these are: fundamental frequency estimates, voicing, accent and meter – these are the observed outputs of a *hidden* note source. These note events are described using a three-state left to right hidden markov model with each unique note represented with a separate HMM.

Each note HMM has 3 states, attack, sustain and silence/noise (Figure 5-5).

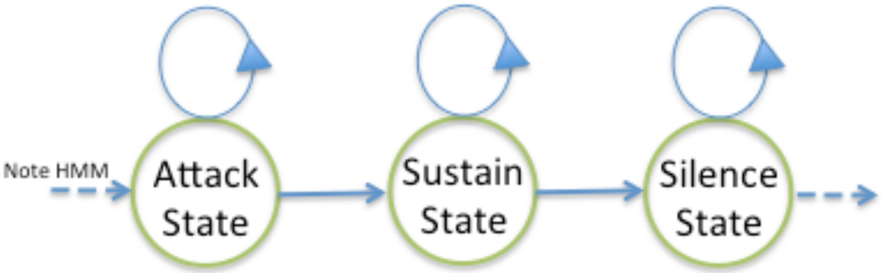


Figure 5-5 Ryynanen and Klapuri's 3 state note HM. The arrows show the possible state transitions

The HMM of different notes are joined into a system where the probabilities of transitions from one note to another is determined by a musicological model. In a similar manner to the Bayesian Network described in section [5.2] the melody is transcribed by finding the most probable path through the network based upon the probabilities given by the note HMMs and the musicological model.

Ryynanen and Klapuri's model is represented in Figure 5-6 Ryynanen and Klapuri's HMM Model.

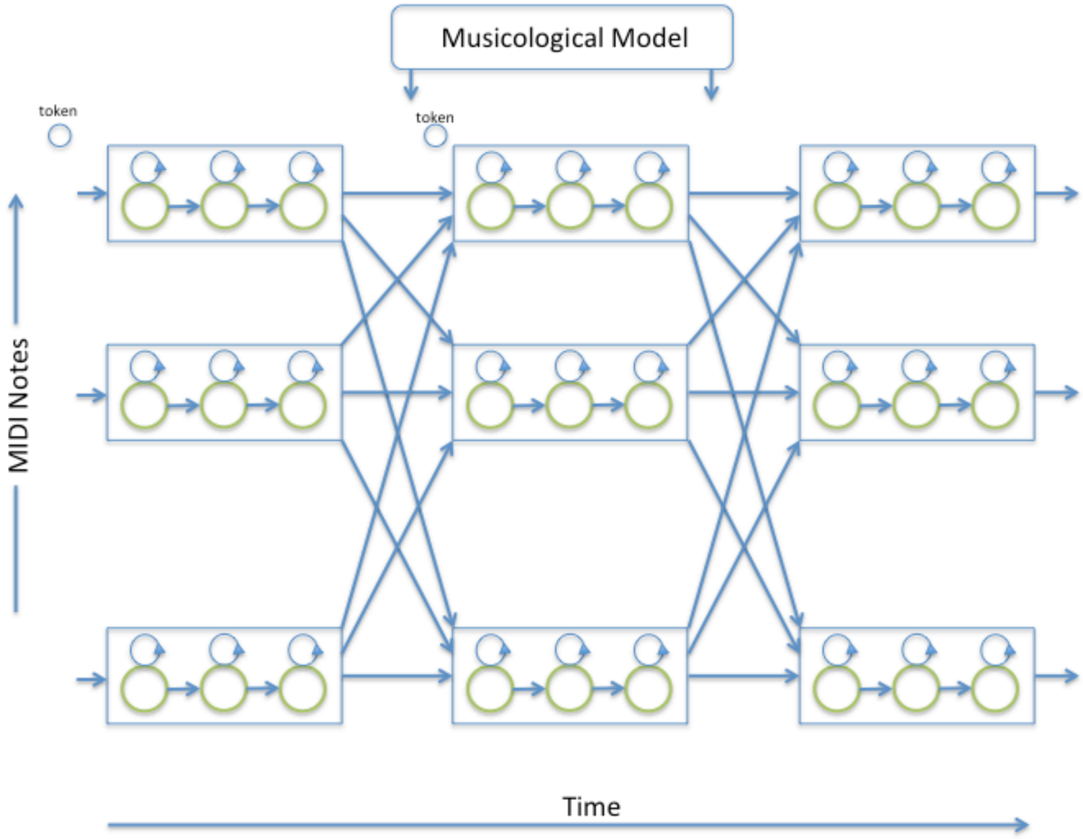


Figure 5-6 Ryynanen and Klapuri's HMM Model

5.4 Non Negative Matrix Factorization

Non negative Matrix Factorization (NMF) is a statistical process which doesn't rely on probability models, but functions on the principle of analyzing multiple variables at a single point in time (Lee & Seung, 1999).

A typical use of the NMF process is to decompose frequency magnitude spectra into two matrices, one to describe frequency information, one to describe time information (Smaragdis & Brown, 2003).

5.4.1 The NMF Process

Time domain signals aren't suitable for NMF as they contain both positive and negative values, but the magnitude of a spectrogram meets the non-negative requirement (Wang & Plumbley, 2005).

NMF decomposes an M by N matrix V in to two non negative matrices W and H. V is approximated by the product of the W and H (Equation iv).

$$V \approx W\ H$$

Equation iv

W is an M by R basis matrix, and H is an N by R coefficient matrix (Wang & Plumbley, 2005). For example where V = 28, M = 7, N = 4, R = 5 the following matrices are generated.

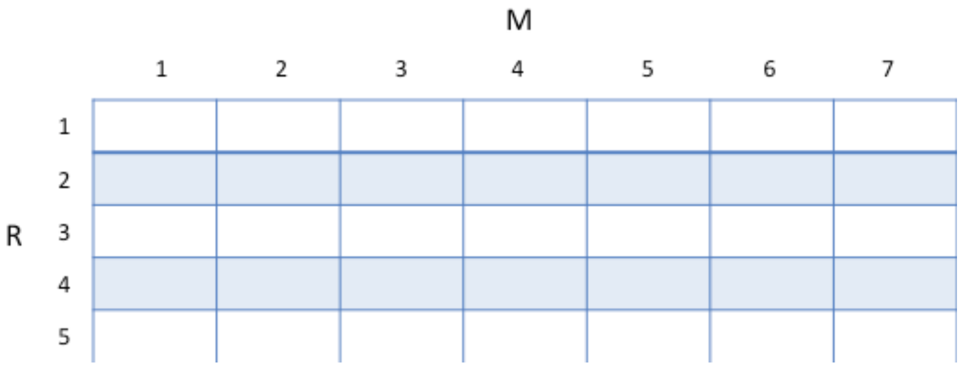


Figure 5-7 Matrix W

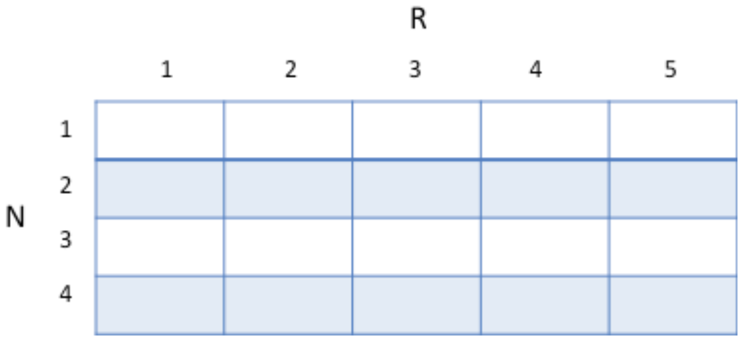


Figure 5-8 Matrix H

In simple terms the NMF summarises the profiles of the rows of V in the rows of H, and likewise for the columns of V in the columns of W (Smaragdis & Brown, 2003).

5.4.2 NMF for Automatic Music Transcription

A desirable characteristic of the NMF is its robustness to deal with multiple overlapping notes, which is highlighted in Smaragdis and Brown's polyphonic music example (Smaragdis & Brown, 2003).

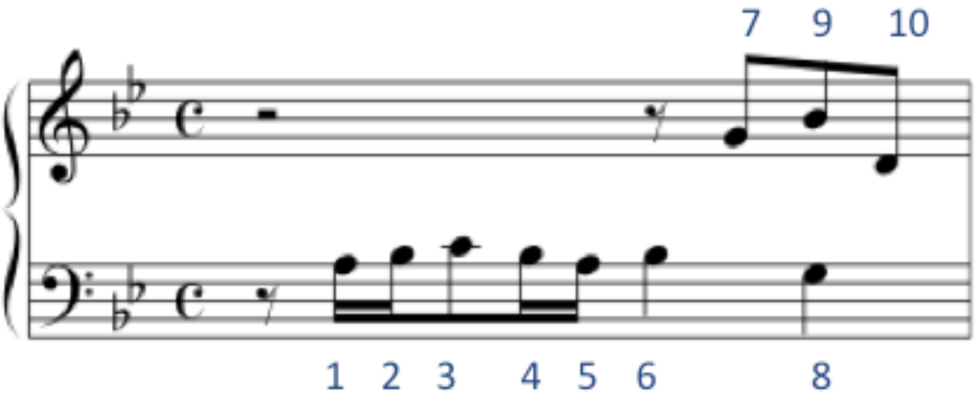


Figure 5-9 Score decomposed by NMF

The above piece of music (Figure 5-9 Score decomposed by NMF) is decomposed to W and H matrices where R=7 - the number of unique note frequencies. W stores frequency information, H stores time information. Analysis of the resultant matrices shows that the two simultaneous notes (numbered 8 and 9 on the stave) towards the end of the sample are consolidated as a single component (Figure 5-10). This is not ideal for polyphonic music transcription.

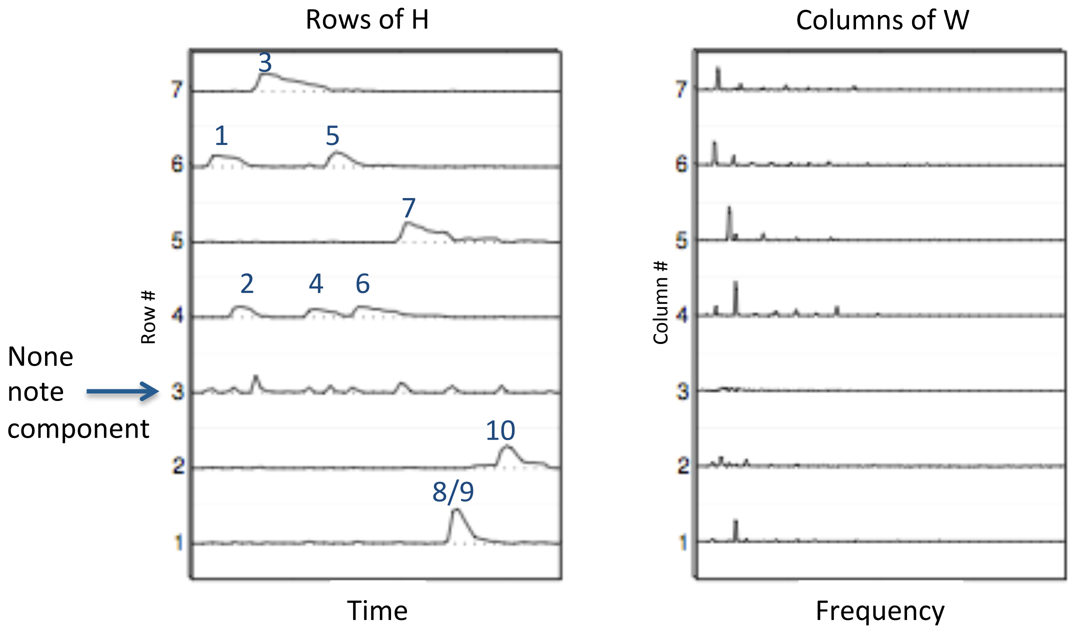


Figure 5-10 NMF Decomposition

Diagram showing the H and W matrices. Notes 8 and 9 from figure [5-9] appear as a single component.

Two notes have been transcribed as a single event because the only time the two notes occur is in unison, so the system only recognizes that combination of notes as a single event. If the system is 'taught' by showing it the note frequencies in isolation, or in different polyphonic groups their individuality will be highlighted and therefore decomposed as separate components.

Teaching a system with audio examples has been successful, and can occur in an 'offline' mode before the decomposition of the musical audio. Dessein et al. first populate a NMF model by using the spectral magnitude from note and instrument samples. The polyphonic signal to be decomposed is then projected on to the **W** matrix containing the learnt prior information during the real-time decomposition phase to accurately transcribed notes sounded in unison as separate events (Dessein, Cont, & Lemaitre, 2010). Figure 5-11 describes Dessein's system.

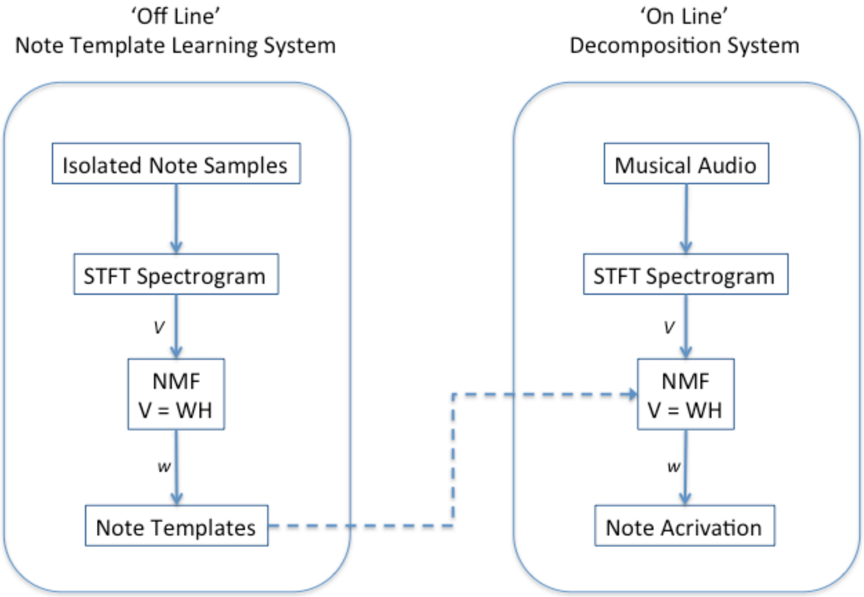


Figure 5-11 Dessein's NMF

A weakness common to systems that utilise prior information is the data the system is trained on is not necessarily spectrally identical to the source being analysed, and therefore induces errors (Dessein, Cont, & Lemaitre, 2010).

Although only overviews of the popular Bayesian, HMM and NMF high level processing techniques have been presented, it is intended to demonstrate that there are advantages to be gained by providing accurate frequency and energy components from the low level processing.

The following chapters investigate the parameters of the FFT for music transcription with the aim of providing quality spectral analysis to the high level processing.

6 Discussion of FFT Parameters for Automatic Music Transcription Algorithms

The Fourier transform and the STFT implementation is the initial process in many modern automatic music transcription algorithms to gain a time-frequency representation of the signal. The aim of the project is to evaluate the characteristics of the STFT as a low level process for multiple fundamental frequency estimation and investigate the optimisation of MRFFT parameters to improve the accuracy of the note candidates presented to the high level processing. For this reason, observations, testing and analysis are performed in the context of music.

The real world nature of the problem is reflected in the preferred methodology of a practical testing approach to determine characteristics and parameters rather than a theoretical or mathematical one.

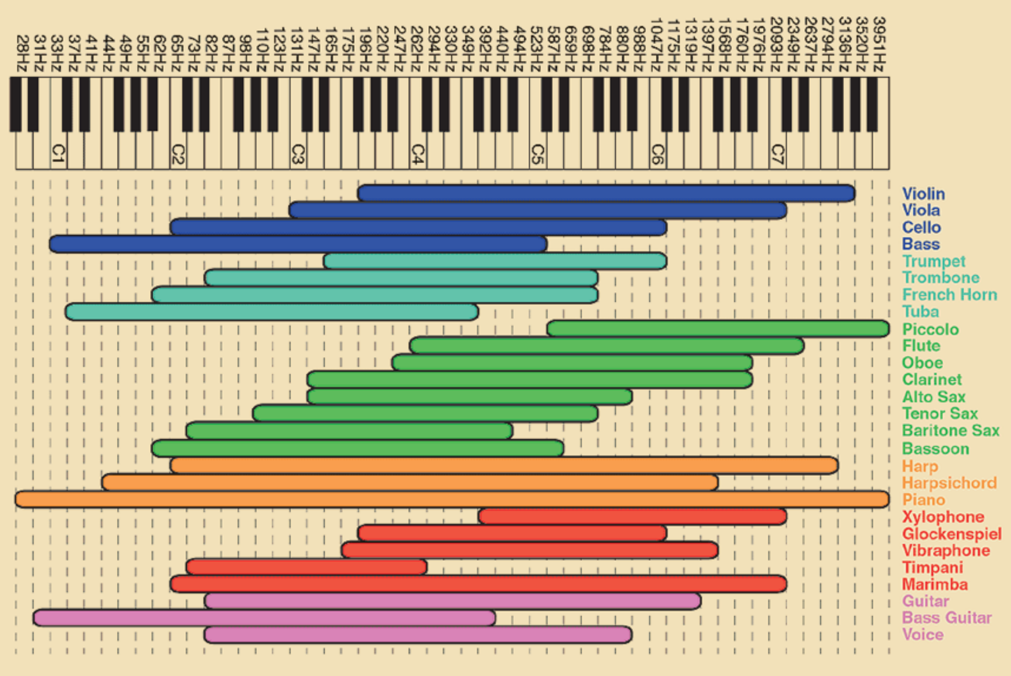
This chapter begins with an introduction to instruments and frequency ranges in the context of a STFT.

The parameters of the STFT are then introduced with discussion of their effect on the STFT output in the context of music transcription algorithms.

All figures, tables and results generated assume a sample rate of 44.1KHz where relevant.

6.1 Frequency Resolution - Instrument Frequencies and MIDI notes

Figure 6-1 shows the frequency range of a variety of musical instruments. The standard piano has 88 keys that stretch to just over 7 octaves. The lowest note on a piano is tuned to approximately 27.5Hz based on equal tempered tuning, with the highest note tuning being 4186.0Hz. Generally instrument frequency scales fall in to this range, although electronic synthesized instruments will generate tones even below 20Hz (the threshold of human hearing).



(White, 2001)

Figure 6-1 Instrument frequency ranges

Computer software music sequencers often represent notes in terms of Musical Instrument Digital Interface (MIDI) notes. MIDI notes are numbered from 0 to

127 and represent a note frequency. There is 1 MIDI note per semitone extending from 8.175Hz (MIDI note 0) to 12542.63Hz (MIDI note 127), the piano range falls between MIDI notes 21 and 108 inclusive.

For automatic music transcription and STFT analysis, it is important to consider the spacing between musical notes. The frequency resolution of the STFT refers to its ability to discern 2 adjacent note frequencies (Dressler, 2006). The lowest two notes on a piano are 27.5Hz and 29.135Hz, resulting in a frequency spacing of 1.635Hz (delta frequency) (Figure 6-2). To successfully identify these two fundamental frequencies as separate frequency events a resolution of 1.635 Hz is required.

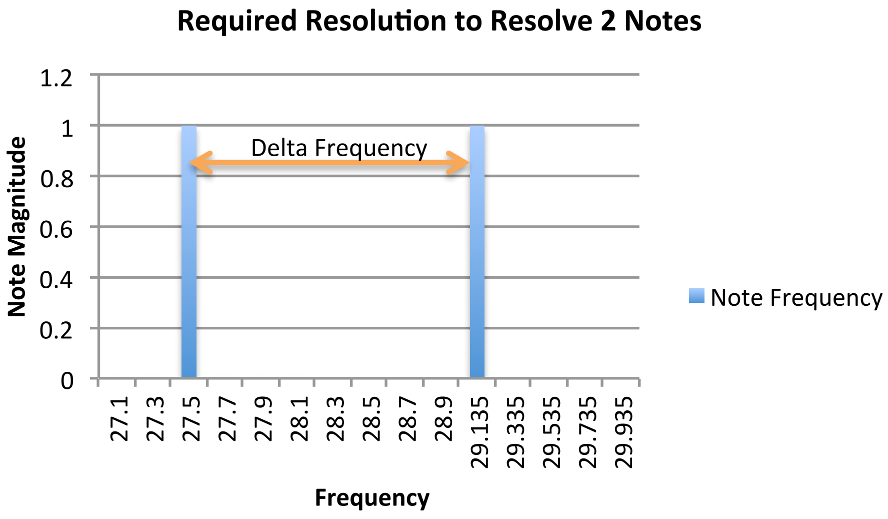


Figure 6-2 Delta frequency

In contrast, due to the logarithmic nature of equal tempered tuning, the top two notes have a delta frequency of 234.9Hz. Figure 6-3 plots the delta frequencies against MIDI note numbers to show the frequency resolution range required for automatic music transcription.

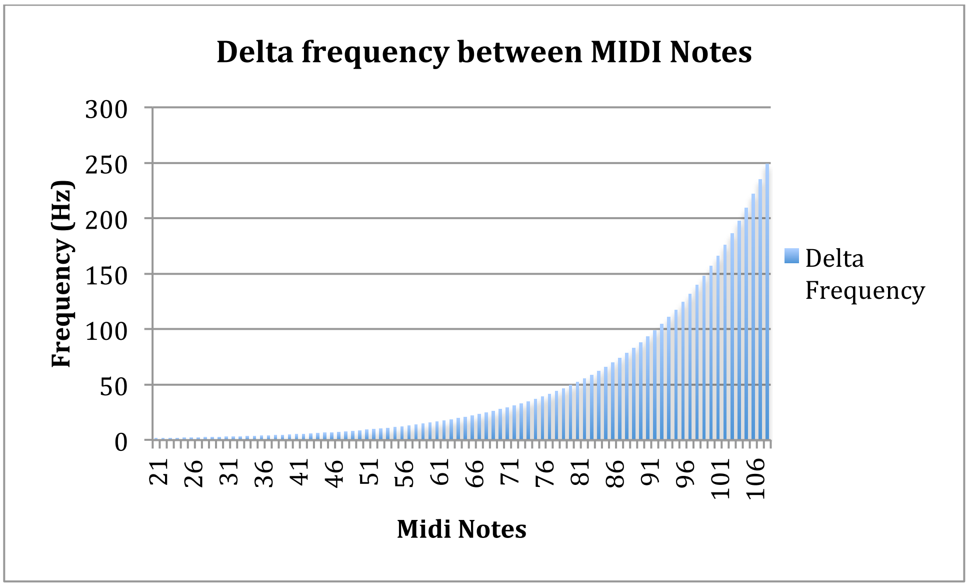


Figure 6-3 Note delta frequencies

The delta frequency provides a measure for the required frequency resolution of an FFT to accurately represent individual notes. However, the corresponding time resolution also must be considered, particularly in relation to transcribing short notes.

6.2 Time Resolution – Note Lengths and BPM

Musical notes have a frequency value (pitch) but also a length measured in beats. A beat is a relative measurement to the Beats per Minute (BPM), which is the number of integer beats the music features within 60 seconds. BPM is directly related to the meter and pace of the music. The name of the most common note lengths and beat value are listed in the Table 5.

Note Name	Beat Value
Semibreve	4
Minim	2
Crotchet	1
Quaver	0.5
Semi Quaver	0.25
Semi Demi Quaver	0.125
Semi Demi Hemi Quaver	0.0625

Table 5 Note names and values

Moelants collated the BPM of 74042 pieces of popular music and plotted the distribution. The BPM of the majority of popular music is in the range between 115 and 127BPM (Moelants, 2002). The mean is 121 BPM, which is not far from the popular assumption of 120BPM being the average for popular music. The plot of BPM that Moelants generates shows a steep drop in the use of BPMs over 150, with relatively few popular music compositions created with a BPM of over 200. 200BPM will therefore be considered the maximum realistic BPM a system would have to deal with. However, it should be noted that in part BPM is a perceptual measurement. For example, a composition at 200BPM that uses notes twice as log as a composition at 100BPM, the 200BPM will be perceived at 100BPM.

Table 6 shows the length in seconds of the different notes for different BPMs.

Note Name	Beat Value	100BPM (secs)	120BPM (secs)	200BPM (secs)
Semibreve	4	2.4	2	1.2
Minim	2	1.2	1	0.6
Crotchet	1	0.6	0.5	0.3
Quaver	0.5	0.3	0.25	0.15
Semi Quaver	0.25	0.15	0.125	0.075
Demi Semi Quaver	0.125	0.075	0.0625	0.0375
Hemi Demi Semi Quaver	0.0625	0.0375	0.03125	0.01875

Table 6 Note lengths at varying BPMs

If the STFT time resolution is unsuitable, shorter notes will be ‘seen’ as longer notes, and onset times will be less accurate. The FFT window length must be suitable to accurately measure the length of a note.

6.3 FFT Window Length and Note Length

The parameter of the STFT that determines the time resolution is the data length (or FFT length for non-zero padded transforms – see section 6.10), which is measured in samples. The longer the analysis window, the more samples it contains, therefore a longer period of time is analysed. This results in a coarser time resolution than a shorter window.

In the context of music analysis, the window length determines the shortest note that can be transcribed.

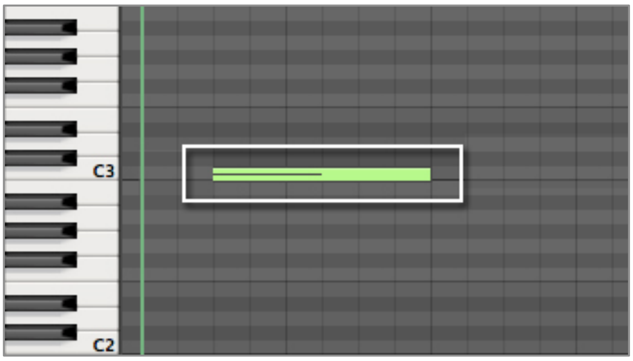


Figure 6-4 Note positioning within time domain window

Figure 6-4 is a diagramatic representation of a note positioned within a time domain wondow. The note is represented as the green bar in a piano roll format (a screen grab from Apple's Logic Sequencer) where the virtical axis is frequency and the horizontal axis is time. The horizontal scale is not of consequence as it is the relative position of the frequency component to the time domain window that is of interest. The white box is drawn on top of the screen grab and is representative of the FFT window length. The only time information available is the time at the start of the window and the time at the end of the window. Therefore, even though the note is shorter than the FFT window, the note transcribed as a result of the FFT output is quantised to an onset and end time equal to that of the start and end points of the FFT analysis window.

This forced quantisation causes onset and note length errors equal to the distance between the start of the analysis window and the note onset, and the end of the note and the end of the analysis window. The red blocks at the start and the end of the green piano roll note in Figure 6-5 show the error resulting from the analysis in Figure 6-4.

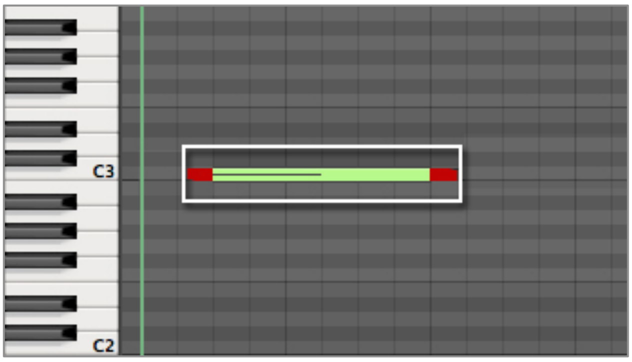


Figure 6-5 Note length error

As shown in chapter 2, the length of the FFT in samples determines both the time and frequency resolution of the FFT. It is the interplay of these two characteristics generated by the data length that results in the real world trade off of the STFT.

6.4 STFT Data Length – Time and Frequency Resolution

The data length is the number of samples of a signal that are entered in to the STFT. Zero Padding is discussed in section [6.10] but for this section assume the data length is equal to the STFT length.

In accordance with the decomposition described in section [2.6.2], as the FFT length increases (and therefore the time period it represents), the number of bins increases resulting in a finer frequency resolution. If the FFT length is shorter to better localize a frequency in the time domain, the number of bins is reduced and the frequency resolution becomes coarser. This is in line with Heisenberg's uncertainty principle that states the exact position (time) and momentum (frequency) of a particle (signal) cannot be known simultaneously (Smith S. W., 1997).

The corresponding time resolution of an FFT length is calculated as:

$$Time\ Resolution\ = \frac{L}{Fs}$$

Equation v

Where:

L is FFT Length

Fs is Sample Rate

The frequency resolution of an FFT length is calculated as;

$$Frequency\ Resolution\ = \frac{Fs}{L}$$

Equation vi

Where:

L is FFT Length

Fs is Sample Rate

Table 7below shows the interaction of the time and frequency resolution for different power of 2 lengths of window analyzing audio sampled at 44100Hz.

FFT Length	Time Resolution (s)	Frequency Resolution (Hz)
16	0.00036	2756.25
32	0.00073	1378.13
64	0.00145	689.06
128	0.00290	344.53
256	0.00580	172.27
512	0.01161	86.13
1024	0.02322	43.07
2048	0.04644	21.53
4096	0.09288	10.77
8192	0.18576	5.38

Table 7 FFT Time and frequency resolution

The inverse relationship between time and frequency resolution of the FFT is demonstrated in the above table, as one increases the other decreases. Figure 6-6 shows a plot of the data in Table 7 but range corrected so all values are between 1 and 0.

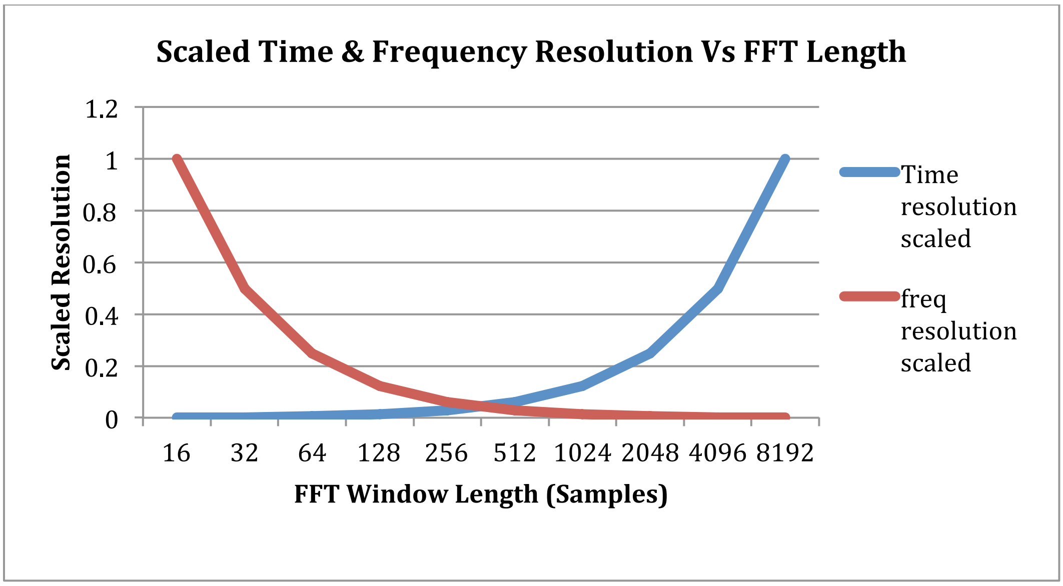


Figure 6-6 Scaled time and frequency resolution

Although the scale has been modified so both properties can be plotted on the axis, the visualization of the proportional interplay between the time and frequency resolution relative to the window length of the FFT provides a clear account of the resolution trade off. The curve of the graph indicates the necessity of a longer FFT to generate a resolution capable of differentiating two low notes with a small delta. Equally, the curve indicates the need for a short FFT length to accurately transcribe a note length and onset time.

Dressler uses 4 different data lengths, 256, 512, 1024, 2048 for a MRFFT (Dressler, 2006). An FFT length of 2048 samples will result in a frequency resolution of 21.53Hz. Therefore, due to the logarithmic nature of the equal tempered scale, the magnitude in the FFT output (spectral magnitude) generated by any note frequency lower than 369.9Hz will be represented by a frequency bin that also represents at least one other note. Figure 6-7 A 2048 FFT decomposition of a 27.5Hz and 29.1Hz sine wave mixture shows the FFT spectral magnitude of two sine waves tuned to 27.5Hz and 29.1Hz representing the two lowest notes on a standard piano analysed by a single FFT length of 2048 samples.

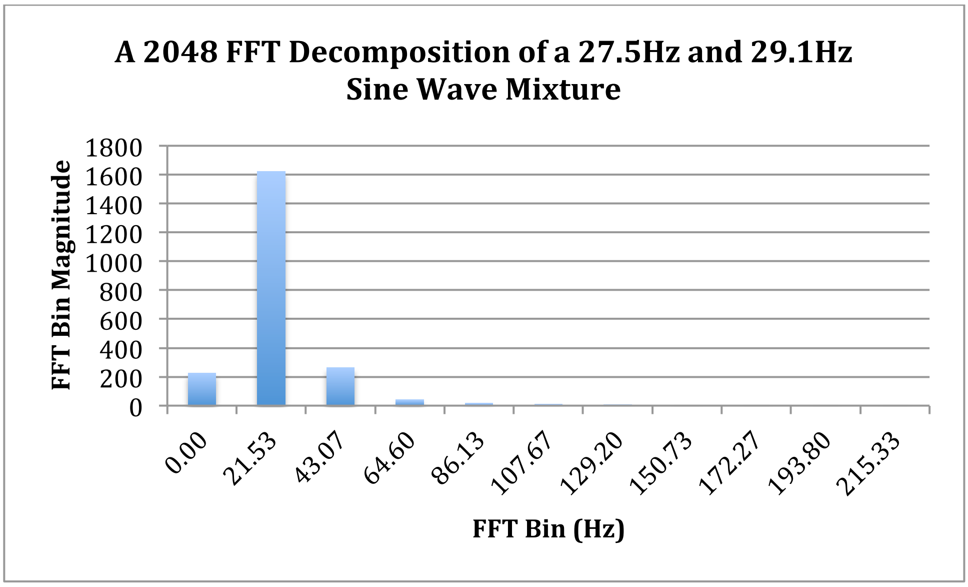


Figure 6-7 A 2048 FFT decomposition of a 27.5Hz and 29.1Hz sine wave mixture

There is only 1 distinct peak in the spectra, making it impossible to detect the two discrete frequencies. The magnitudes in adjacent bins are not considered to be peaks due to their low energy level. A threshold based peak picker would determine such low levels to be noise and not present the frequencies as note candidates. An FFT length of 26972 samples would be required for the two frequencies 27.5Hz and 29.1Hz to be represented by dedicated FFT bins (Figure 6-8 A 26972 FFT decomposition of a 27.5Hz and 29.1Hz sine wave mixture).

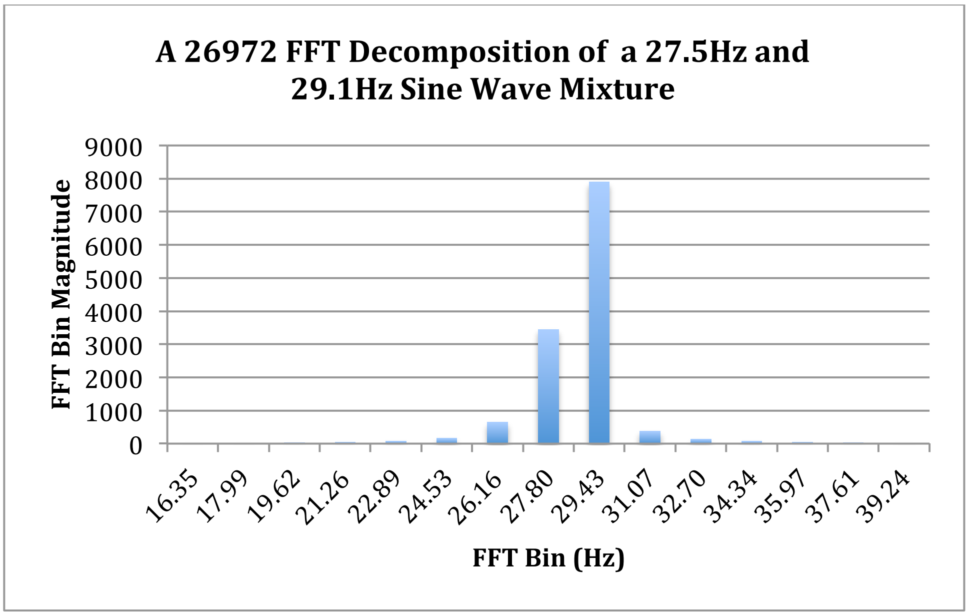


Figure 6-8 A 26972 FFT decomposition of a 27.5Hz and 29.1Hz sine wave mixture

Figure 6-9 shows the FFT spectral magnitude of two sine waves tuned to 369.9Hz and 391.9Hz, the first pair of notes with a delta larger than the 21.53Hz resolution of the 2048 FFT. Despite some cross channel interference between the bins of the FFT (discussed in section [6.5]) two distinct magnitudes are visible, which would result in a suitably configured threshold based peak picking algorithm correctly detecting the 2 frequencies.

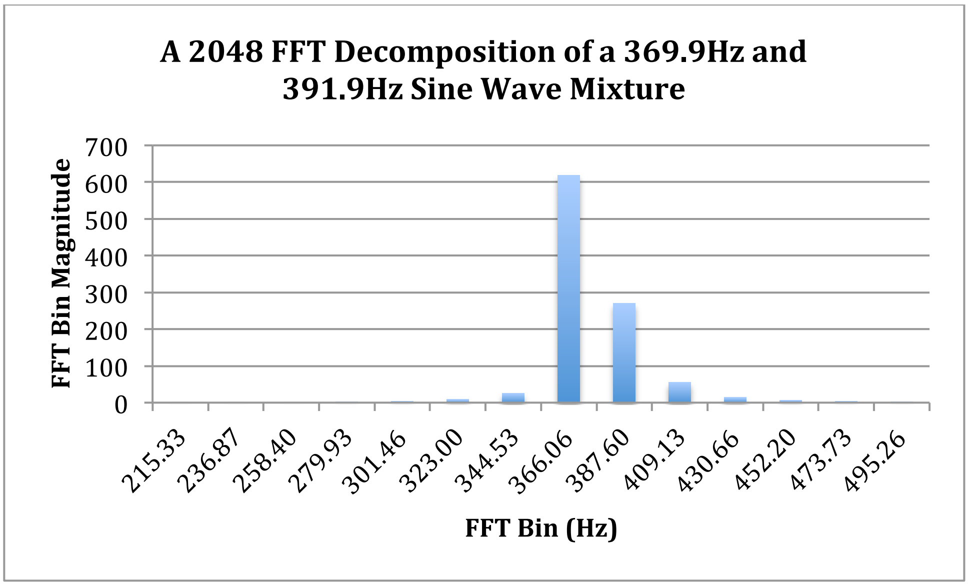


Figure 6-9 Suitable delta frequency

The energy present in the bins either side of the peak values is a result of the FFT decomposition method. This 'cross channel interference' needs to be considered, as it is undesirable 'noise' for the purpose of peak picking note candidates, but also contain essential information for a successful inverse FFT

6.5 Cross Channel Interference

Figure 6-9 is the spectral output of a 2048 FFT with a corresponding frequency resolution of 21.53Hz. The signal analysed was a mixture of a 369.9Hz and 391.9Hz sine waves of equal amplitude. The delta between these notes is 22Hz, so the resolution of the FFT is suitable. Therefore, it might be expected that as the two frequencies are represented by separate bins there would be two strong maxima in the relevant bins and no energy in the FFT bins not representing those two frequencies. However, as can be seen in Figure 6-9 this is not the case. This behavior is often referred to as *cross channel interference*.

It is important to remember the bins of an FFT represent a single sine wave frequency, not a range of frequencies. Imagining the FFT as a bank of very narrow pass band filters with overlapping stop and start bands, with the bin frequency values as the center frequency of each filter it is easy to imagine a frequency not represented by a center frequency would appear in adjacent filters. This is the case with frequencies decomposed by the FFT in Figure 6-9.

If a signal frequency is not represented by a bin value, then the energy associated with that frequency in the signal will be distributed across several bin values as the FFT attempts to represent that sinusoidal frequency with the frequency representations it does have. Figure 6-10 shows the FFT spectral magnitude of 376.83Hz sine wave analysed by a 2048 sample FFT.

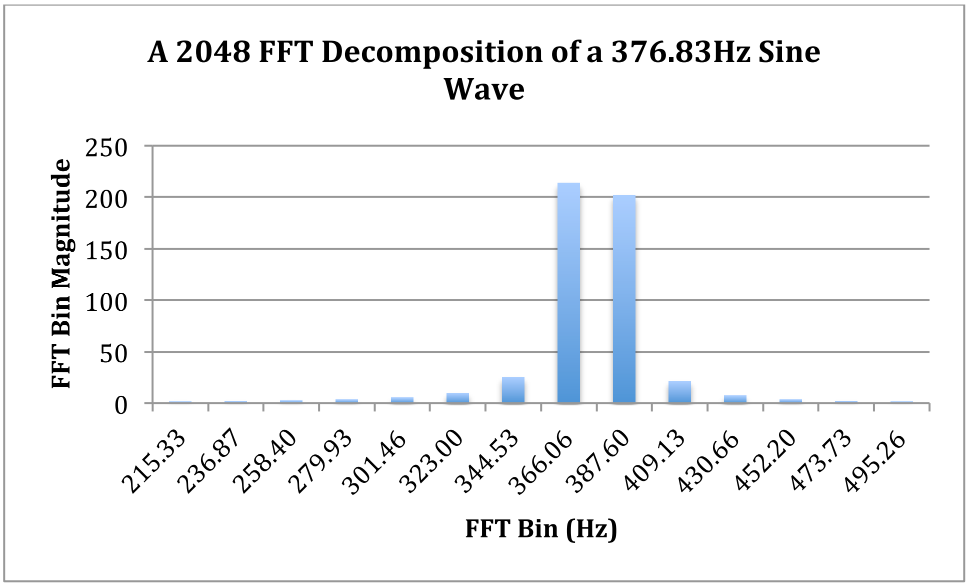


Figure 6-10 Maxima due to cross channel interference

376.83Hz is exactly half way between two bins of a 2048 FFT (using 44.1kHz as the sample rate), and the cross channel interference generated results in 2 strong magnitudes in the bins either side and also a decreasing spread of energies into all frequency bins away from the fundamental frequency bin (Roads, 1996, pp. 561-562).

Figure 6-11 shows the FFT spectral magnitude of a 366.06Hz sine wave, which matches an FFT bin value exactly, analysed by a 2048 sample FFT.

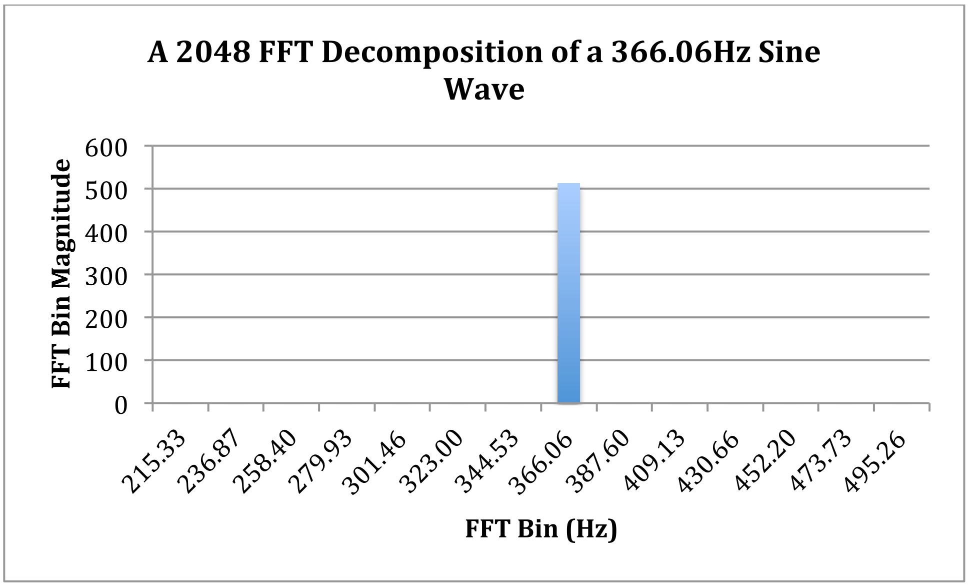


Figure 6-11 Bin value matches frequency

As expected a single strong magnitude is present as the signal frequency matches the FFT bin frequency.

As well as looking at cross channel interference from the perspective of frequency bins, it can also be considered in terms of the complete (or incomplete) cycles of a wave represented by a time domain window.

6.6 Cross channel interference – Cycles

Figure 6-12 shows the first 2048 samples of a 366.06Hz sine wave as analysed by the FFT in Figure 6-11.

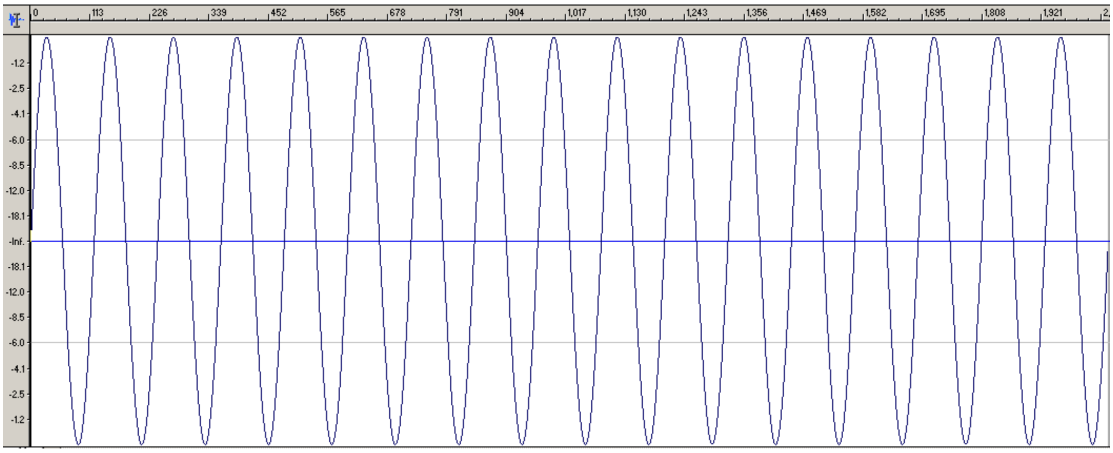


Figure 6-12 Complete number of cycles

Figure 6-12 shows that a signal with a frequency equal to a bin value of the FFT will contain a complete number of cycles within the analysis window.

Figure 6-13 shows the first 2048 samples of the 376.8Hz sine as transformed in Figure 6-10

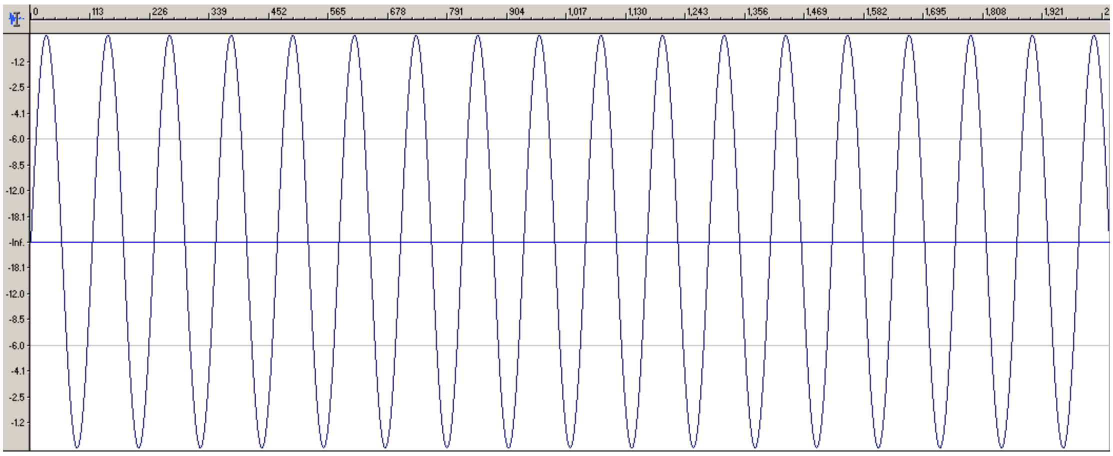


Figure 6-13 Incomplete cycles

As 376.8Hz is exactly between the values of the FFT bins there is not an integer value of cycles. The first 2048 samples of the wave contain 17.5 cycles of the signals.

The first 2048 samples of the 369.9Hz sine wave analysed in Figure 6-10 that suffered from significant cross channel interference is show in Figure 6-14.

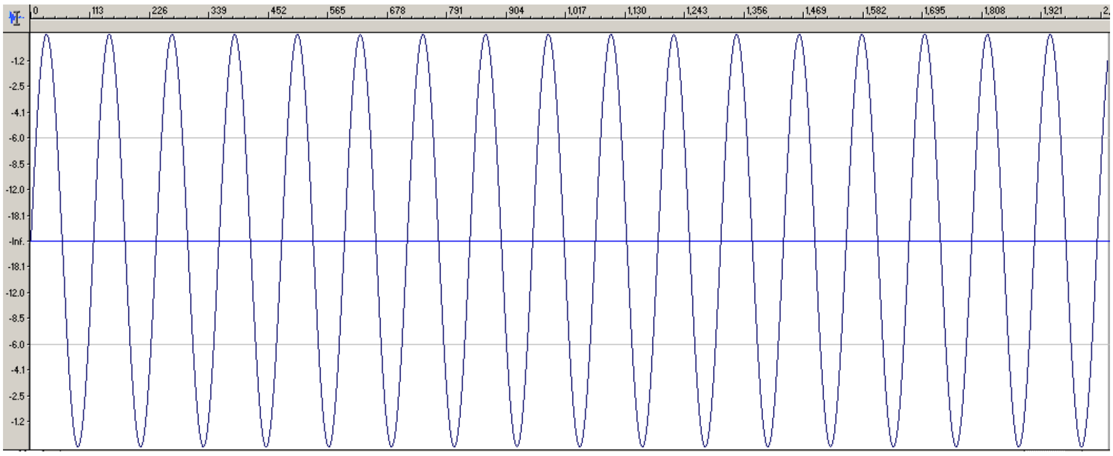


Figure 6-14 Incomplete cycle of 369Hz

The FFT window length contains 17 complete cycles and a fraction of a cycle.

The discontinuities in the waveforms of frequencies that do not match the bin values generate cross channel interference. Considering the FFT output as a collection of sine waves of set frequencies and varying magnitudes that when combined recreate the original time domain signal, extra sine waves are required to construct the non regular discontinuities at the ends of the windowed waveform (Roads, 1996, p. 1098).

By applying a window shape to the rectangular data length the discontinuities are compressed and therefore the magnitudes of the sine waves required to represent the discontinuities are also compressed, thus attenuating cross channel interference. Shaping the time domain samples to be transformed can result in both positive, and negative outcomes.

6.7 Analysis Frame Window Shape

A rectangular window is the result of sampling the signal to be analysed and not shaping the signal in anyway. An alternative is to apply an envelope to the sampled signal to smooth the edges of a rectangular window and suppress the discontinuities associated with windowing (discussed in section 6.7). Figure 6-15 shows the process of applying an envelope to a STFT analysis window.

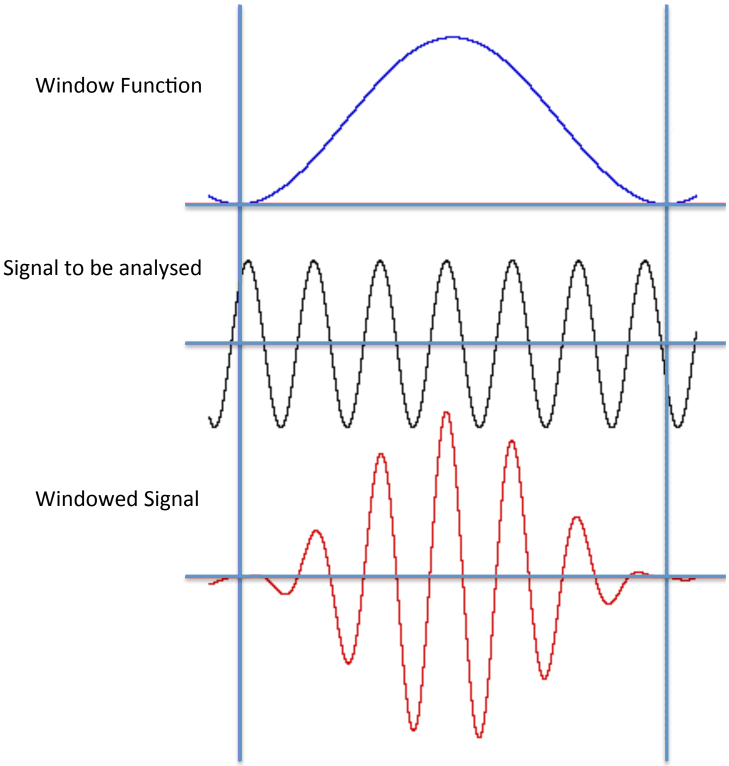
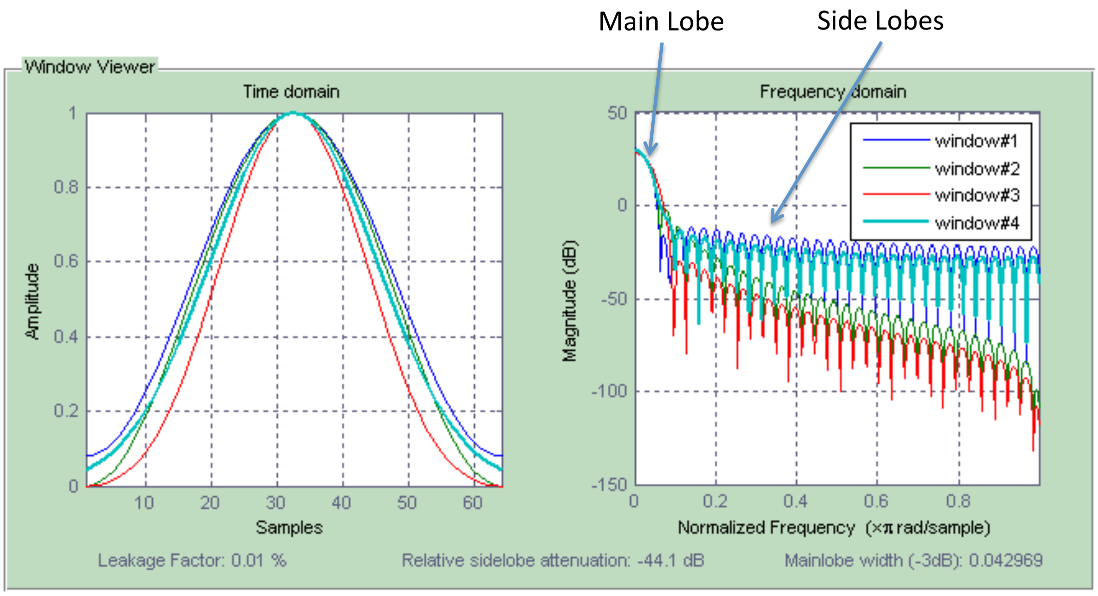


Figure 6-15 Windowing Process

The resulting waveform shape in Figure 6-15 shows the minimization of the discontinuity of waveform by reducing the time domain amplitude of the signal at the edges of the window.

Applying an envelope to the signal in the time domain also has an effect on the signal in the frequency domain, which can be positive or negative dependent upon the envelope shape used (Roads, 1996, p. 1099). An envelope shape is characterized by *lobes* in the frequency domain – a main lobe and a series of side lobes on either side.



Leakage Factor: 0.01 %

Relative sidelobe attenuation: -44.1 dB

Mainlobe width (-3dB): 0.042969

Figure 6-16 Plots of Hamming (Blue), Hann (Green), Blackman (Red) and Gaussian (Turquoise) window shapes and corresponding lobes.

The height of the side lobes indicates the effect they will have in frequency bins that they 'land' on. High side lobes will increase any components present in the corresponding bins. Low side lobes reduce the magnitudes of bins adjacent to the main lobe, but will increase the bandwidth of the main lobe, which itself can lead to cross channel interference.

The ideal envelope shape for automatic music transcription implementations would generate a tall thin main lobe, and no side lobes. Unfortunately, the ideal is not possible.

In a similar scenario to the time-frequency trade off of the FFT, a main lobe-side lobe trade off compromises window shapes. Windows with close to ideal main lobe behavior exhibit poor side lobe behavior. Windows with good side lobe behavior such as the Blackman window or Kaiser Bessel, are compromised by their main lobe behavior (Harris F. J., 1978).

Harris presents extensive analysis and reviews of window shapes for Fourier transforms. The choice of window shape is dependent on application, but whatever that may be, the window shape only changes the shape of the leakage but doesn't eradicate it. Therefore, there is no 'universally best' window choice (Roads, 1996, p. 1103).

Window shapes can be important in automatic music transcription algorithms but need to be evaluated and chosen carefully to ensure they enhance the spectral representation. The following work does not consider the evaluation of this parameter, but there is no reason why future work could not be enhanced by evaluating and using alternative window shapes.

As the context for this thesis is automatic music transcription, it is useful to consider the suitability of the FFT to handle 'real world' music, and to compare the FFT performance with the human auditory system's ability to determine frequency.

6.8 Note Length, Frequency and Cycle

Longer FFTs are used for low frequency notes to accurately represent the frequency content, but time domain factors also need to be considered.

Note lengths in terms of BPM and seconds are shown in Table 5 and Table 6 but these should be considered in terms of FFT window length and cycles per window also to determine the suitability of FFT length to accurately place a note event in time.

Hsieh and Saberi present measured statistics of the number of cycles of a waveform a human requires to accurately identify its pitch. A minimum of 4 cycles is required before human pitch identification is considered to be above chance. For the frequencies between 65.4Hz and 262Hz, the lowest 2 octaves Hseieh and Saberi tested frequency identification was much less accurate than the higher octaves, and required a greater number of cycles of the waveform to be identified. A sample 5 cycles long for a pitch in the lowest 2 octaves would yield only a 10% success rate, 10 cycles yielded a much higher 80% success rate – which was deemed to be an acceptable success rate (Hsieh & Saberi, 2007).

Hsieh and Saberi's experiments were performed by playing a sine wave signal to the listener through headphones in an anechoic room, so did not take into consideration the effects of reverb. In the 'real world' recordings of music will feature reverb, whether naturally occurring or added in the production process.

The effects of reverb on music include a 'blurring' of pitch as a note frequency and its harmonics will overlap with the next note. The extent to which this happens depends on the length of the reverb used, but the result is an increase in difficulty for an automatic music transcription tool to determine and extract fundamental frequencies (Wilmering, Fazekas, & Sandler, 2010).

Using Hsieh and Saberi's research, assumptions can be made about note duration in the lower frequencies, which can be a guide to suitable time resolutions. If a human requires 10 cycles of a sound to determine the pitch, it is unlikely a composer will write a note that is shorter than 10 cycles and therefore undetectable.

If a note is significantly sorter than the analysis window, the timing and length of the transcribed note will be compromised, but if the FFT analysis window is approximately the length of 10 cycles of the low frequencies, then it can be assumed the time resolution is sufficient to accurately transcribe the shortest possible note that would be used in the lower octaves

Table 8 shows the time duration for 1 and 10 cycles of the 2 octaves of an equal tempered piano. It also displays the number of samples required to sample 10 cycles of each frequency at 44.1kHz sampling rate, and what the frequency resolution of an FFT is with a data length equal to the number of samples required to represent 10 cycles. The penultimate column displays the delta frequency between the current note and the one above, and the final column displays the required FFT to ensure each note is represented by a discrete bin value.

Note Frequency	Time for 10 cycles	No. of samples for 10 cycles when Fs=44.1KHz	FFT Freq resolution when Data length = 10 cycles samples	Frequency difference to note above in pitch (Delta Freq)	Required FFT Length to resolve delta frequency	Time difference between 10 cycles and required FFT Length
27.50	0.36	16036.36	2.75	1.64	26969	0.248
29.14	0.34	15136.31	2.91	1.73	25455	0.234
30.87	0.32	14286.78	3.09	1.84	24026	0.221
32.70	0.31	13484.92	3.27	1.94	22678	0.208
34.65	0.29	12728.07	3.46	2.06	21405	0.197
36.71	0.27	12013.70	3.67	2.18	20204	0.186
38.89	0.26	11339.42	3.89	2.31	19070	0.175
41.20	0.24	10702.99	4.12	2.45	17999	0.165
43.65	0.23	10102.28	4.37	2.60	16989	0.156
46.25	0.22	9535.28	4.62	2.75	16036	0.147
49.00	0.20	9000.10	4.90	2.91	15136	0.139
51.91	0.19	8494.97	5.19	3.09	14286	0.131
55.00	0.18	8018.18	5.50	3.27	13484	0.124
58.27	0.17	7568.16	5.83	3.46	12727	0.117
61.74	0.16	7143.39	6.17	3.67	12013	0.110
65.41	0.15	6742.46	6.54	3.89	11339	0.104
69.30	0.14	6364.04	6.93	4.12	10702	0.098
73.42	0.14	6006.85	7.34	4.37	10102	0.093
77.78	0.13	5669.71	7.78	4.63	9535	0.088
82.41	0.12	5351.49	8.24	4.90	9000	0.083
87.31	0.11	5051.14	8.73	5.19	8495	0.078
92.50	0.11	4767.64	9.25	5.50	8018	0.074

Table 8 Note lengths in cycles and FFT lengths

The fourth and final columns show the discrepancy between a suitable data length to accurately represent the length of a note 10 cycles long, and the frequency resolution required to resolve the delta frequency components. The FFT length required to resolve the delta frequencies is consistently larger than the length of 10 cycles, which will lead to a transcription timing error.

The error between the shortest note (10 cycles) and the shortest FFT length to resolve the delta frequency is shown in real terms in Table 10 where the error is calculated as seconds. The lengths of a crotchet, quaver and semi quaver beat at 120bpm is shown as a reference in Table 9.

Crotchet @ 120bpm (seconds)	Quaver @ 120bpm (seconds)	Semi Quaver @ 120bpm (seconds)
0.5	0.25	0.125

Table 9 Note references

Note Frequency	Time for 10 cycles (Seconds)	Time difference between 10 cycles and required FFT Length (Seconds)	10 cycle note transcription length
27.50	0.36	0.248	0.608
29.14	0.34	0.234	0.574
30.87	0.32	0.221	0.541
32.70	0.31	0.208	0.518
34.65	0.29	0.197	0.487
36.71	0.27	0.186	0.456
38.89	0.26	0.175	0.435
41.20	0.24	0.165	0.405
43.65	0.23	0.156	0.386
46.25	0.22	0.147	0.367
49.00	0.20	0.139	0.339
51.91	0.19	0.131	0.321
55.00	0.18	0.124	0.304
58.27	0.17	0.117	0.287
61.74	0.16	0.110	0.270
65.41	0.15	0.104	0.254
69.30	0.14	0.098	0.238
73.42	0.14	0.093	0.233
77.78	0.13	0.088	0.218
82.41	0.12	0.083	0.203
87.31	0.11	0.078	0.188

Table 10 10 cycle note errors

Table 10 above shows that the longest window required to transcribe the smallest delta frequency (27.50 Hz Note frequency) will increase the 10 cycle note to a length of 0.608 seconds. This is the equivalent of a dotted quaver being transcribed as a note longer than a crotchet at 120BPM (Figure 6-17).

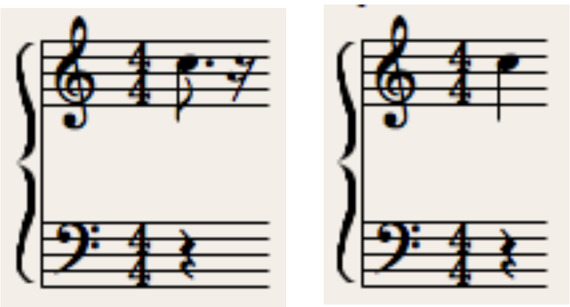


Figure 6-17 Real term transcription note error – original note (left) transcribed note (right)

The percentage increase of transcribed note length compared to the length of 10 cycles for the lowest 2 octaves is approximately 68%, so the minimum note is length (10 cycles) will always be transcribed as an event of 1.68 times longer.

This demonstrates in real terms the trade off between the required frequency resolution and the required time resolution.

Note Frequency	Cycles in a 256 FFT Window	Cycles in a 512 FFT Window	Cycles in a 1024 FFT Window	Cycles in a 2048 FFT Window	Cycles in a 4096 FFT Window
27.50	0.16	0.32	0.64	1.28	2.55
29.14	0.17	0.34	0.68	1.35	2.71
30.87	0.18	0.36	0.72	1.43	2.87
32.70	0.19	0.38	0.76	1.52	3.04
34.65	0.20	0.40	0.80	1.61	3.22
36.71	0.21	0.43	0.85	1.70	3.41
38.89	0.23	0.45	0.90	1.81	3.61
41.20	0.24	0.48	0.96	1.91	3.83
43.65	0.25	0.51	1.01	2.03	4.05
46.25	0.27	0.54	1.07	2.15	4.30
49.00	0.28	0.57	1.14	2.28	4.55
51.91	0.30	0.60	1.21	2.41	4.82
55.00	0.32	0.64	1.28	2.55	5.11
58.27	0.34	0.68	1.35	2.71	5.41
61.74	0.36	0.72	1.43	2.87	5.73
65.41	0.38	0.76	1.52	3.04	6.07
69.30	0.40	0.80	1.61	3.22	6.44
73.42	0.43	0.85	1.70	3.41	6.82
77.78	0.45	0.90	1.81	3.61	7.22
82.41	0.48	0.96	1.91	3.83	7.65
87.31	0.51	1.01	2.03	4.05	8.11
92.50	0.54	1.07	2.15	4.30	8.59
98.00	0.57	1.14	2.28	4.55	9.10
103.83	0.60	1.21	2.41	4.82	9.64
110.00	0.64	1.28	2.55	5.11	10.22
116.54	0.68	1.35	2.71	5.41	10.82
123.47	0.72	1.43	2.87	5.73	11.47

Table 11 Note cycles vs. FFT length

Table 11 shows the number of cycles of different note frequencies contained in 4 typical FFT window lengths. It is interesting to note that an FFT length of 4096 contains fewer than 10 cycles of frequencies below 110Hz. An FFT of 4096 samples can determine a frequency below 103.83 Hz successfully, but by doing so with fewer than 10 cycles it is actually outperforming the human auditory system based upon Hsieh and Saberi's research.

The ability of an FFT to accurately represent a frequency component is important, but for a peak picker to determine that frequency as a note candidate, the magnitude of the frequency in the FFT output is important.

6.9 Magnitude of the FFT

The magnitude of the output of the FFT is dependent on the amplitude of the signal in the time domain, the position of the signal frequency relative to the bin frequencies as discussed above, and also the FFT length. Figure 6-18 and Figure 6-19 show the FFT spectral outputs for a 1076.66Hz sine wave transformed with a sample rate of 44.1kHz. One window length is 8192, and the other is 4096. The frequency value of the input signal is equal to a bin value for each window length. The sine wave peak amplitude in the time domain is 1.

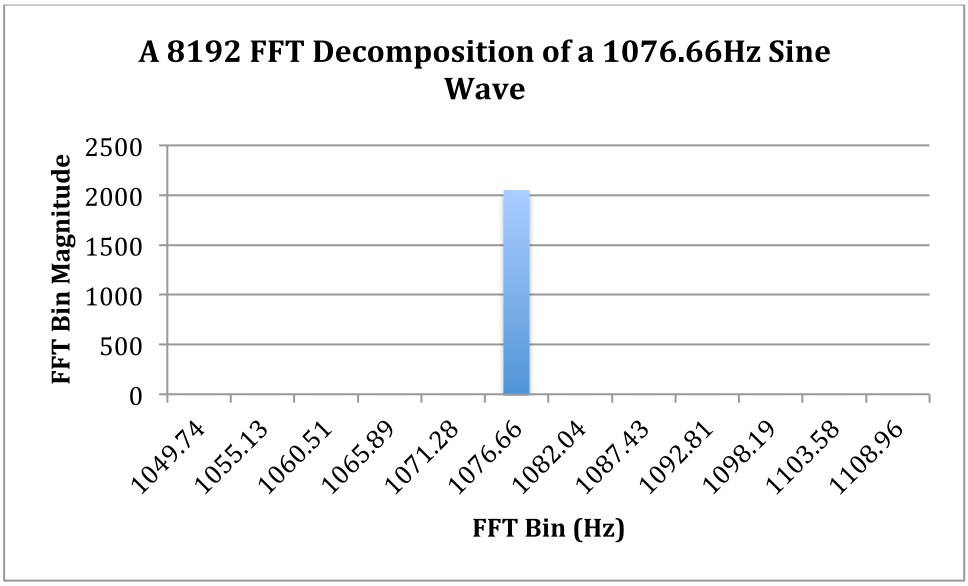


Figure 6-18 8192 FFT magnitude

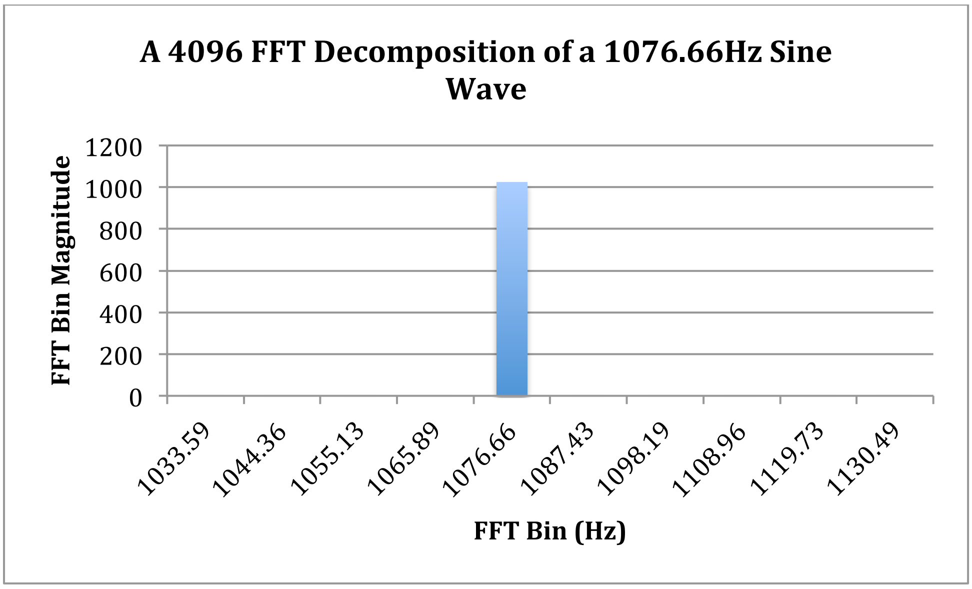


Figure 6-19 4096 FFT magnitude

The magnitude measured for the 8192 FFT is 2048; the magnitude measured for the 4096 FFT is 1024. Each value is a quarter of the FFT length used to transform the signal. This holds true for the power of 2 Fastest Fourier Transform in the West (FFTW) when the input signal is a sine wave equal to a bin frequency value and the amplitude of the input signal is 1. However, it cannot be considered as the ‘maximum’ value. If cross channel interference occurs then the summing of spectral components means there is no ‘maximum’ spectral magnitude.

However, what can be concluded is that longer FFT data lengths generate larger valued spectral magnitudes. The alternative way to consider this is that longer FFT lengths create a finer bin resolution. This means frequencies can be more accurately represented with less spectral leakage, resulting in more defined spectral magnitudes. Large magnitudes in the frequency domain are ideal as it increases the likelihood of the magnitude being chosen by the peak picker as a note candidate.

So far, the discussion of FFT parameters and characteristics has been based upon the data length being equal to the FFT window length, but this doesn’t have to be the case.

6.10 Zero Padding – Data Length and Window Length

Zero padding is the process of adding a series of zeros to the end of a sampled signal usually with the intention of increasing the frequency resolution of the FFT while maintaining a small time domain resolution.

An example is taking a sample of a signal that is 512 samples long (data length). If the sample rate is 44.1kHz, the 512 samples represent a period of time equal to 0.011 seconds. If a further 512 samples of values equaling zero are applied to the end of the data length, then the total number of samples is 1024, allowing a 1024 sample FFT to be performed instead of a 512 sample FFT which would result in a coarser frequency resolution.

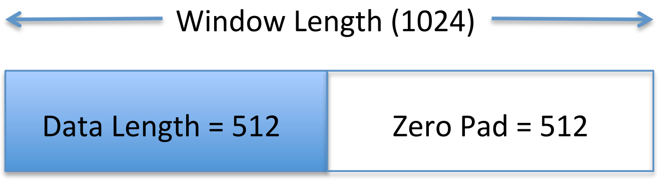


Figure 6-20 Zero padding

When discussing zero padding in relation to FFT window lengths, it is good to consider the FFT as having two frequency resolutions. For the purpose of this thesis they are termed the *native* resolution and the *grid* resolution.

The native resolution depends upon the data length and refers to the ability of the FFT to distinguish 2 closely spaced frequencies (Dressler, 2006). The native resolution is equal to:

$$Native\ Resolution = \frac{Fs}{D}$$

Equation vii

Where:

Fs	is Sample Rate
D	is Data Length

The grid resolution depends upon the window length and refers to the scale on which the FFT outputs are plotted – these are the bin values for the generated frequency spectrum representation (Dressler, 2006). The grid resolution (or bin spacing) is equal to:

$$Grid\ resolution\ = \frac{Fs}{W}$$

Equation viii

Where:

Fs is Sample Rate

W is Window Length

Appending zeros to the end of a time domain signal to generate a finer grid resolution is interpolating the FFT outputs. There is not any actual new data as that is dependent on the native resolution and the sampling frequency (Dressler, 2006).

Zero-padding has the effect of interpolating the points in between the points of the non padded analysis. Only by adding more data samples can the actual frequency resolution be increased, but interpolating extra points, can aide the visualization of curves in the spectrum (Roads, 1996, p. 1104). The interpolation can result in frequencies that fall between bin values of an unpadded FFT being visualized in a zero padded FFT.

The use of zero padding and the interpolation of the FFT can be quite successful. Figure 6-21 and Figure 6-22 show the FFT frequency spectra for a signal containing a 107.6Hz sine and an 118.4Hz sine. One FFT used a data window of 1024 zero padded to an FFT length of 4096, the other used a data length of 1024 and a FFT length of 1024.

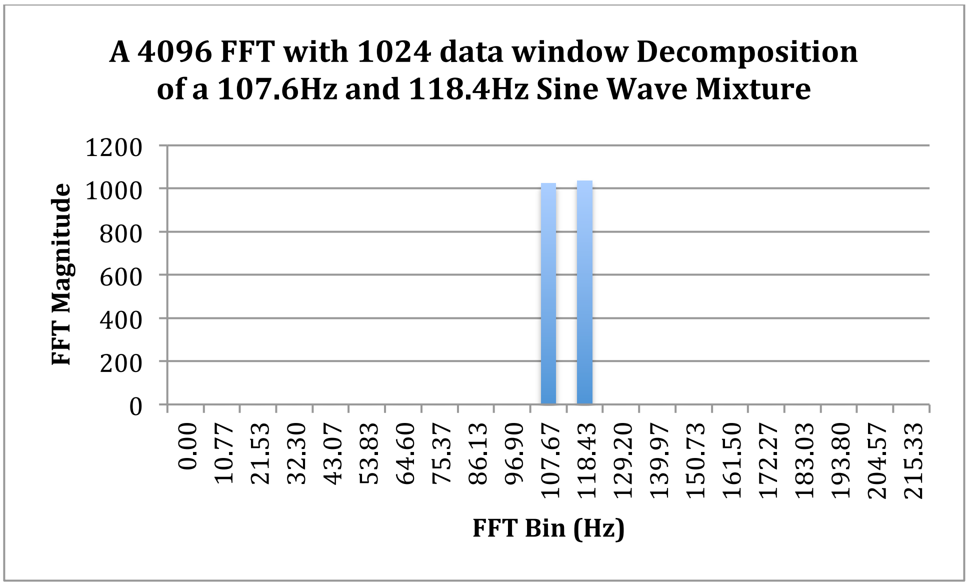


Figure 6-21 Zero padded analysis

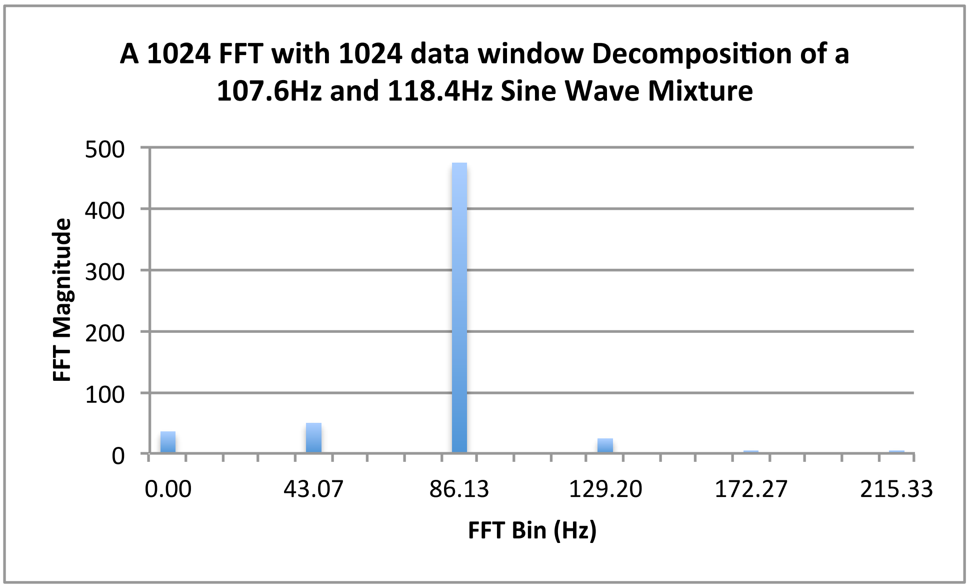


Figure 6-22 Non zero padded analysis

The zero padded version shows 2 distinct magnitudes, which accurately represent the 2 frequencies being decomposed. In the non zero padded decomposition the 2 signals are represented by a single significant magnitude in the FFT output spectrum, but if no new information is added when a signal is zero-padded, why are the 2 signals visible in the zero padded FFT?

The information to construct the 2 frequency components was present in both transforms. This is true because if the inverse FFT were performed the time domain signal would be perfectly reconstructed. However, in the decomposition of the non-padded FFT the data is hidden and not displayed (Quach, 2008). The non-padded version can't display the 2 peaks whereas the padded version can, but it only as a function of interpolation of the data points generated from the data length. As it is the data length that provides actual samples of the time domain signal rather than interpolated points, the native resolution is still of importance (Quach, 2008) (National Instruments, 2006).

Zero padding is a useful technique, but can also cause problems. The examples above use a frequency that matches a bin frequency value. Figure 6-23 and Figure 6-24 shows the FFT spectral outputs for a 91Hz sine wave whose frequency does not match the bin values of the native or padded FFT frequency resolution.

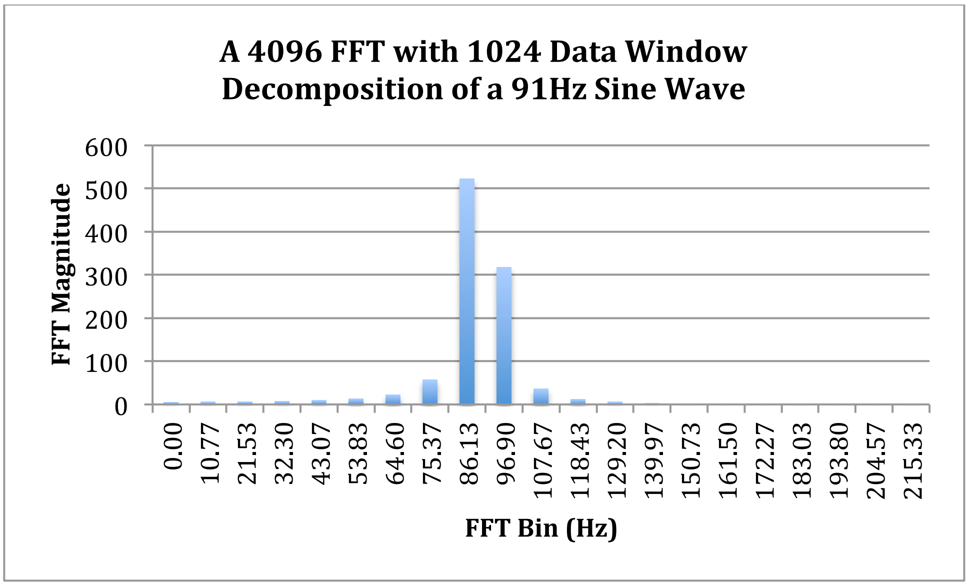


Figure 6-23 not useful zero padding

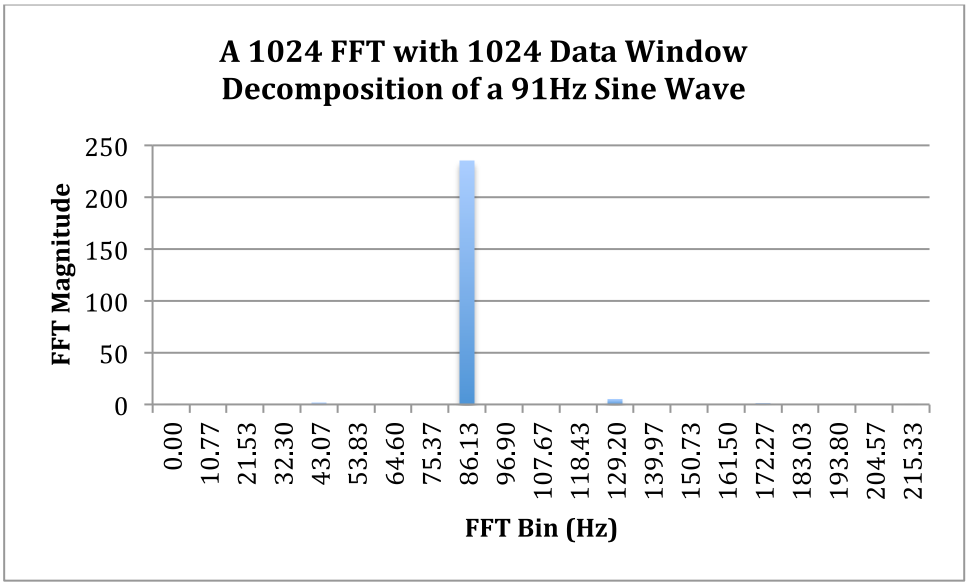


Figure 6-24 Accurate decomposition

Although from the zero padded FFT it can be determined that the frequency is between 86.13Hz and 96.90Hz, it is at the cost of generating increased cross channel interference without providing any further information regarding the actual frequency of the signal. Equally, the same spectral output could be interpreted as representing 2 frequency components, one at 86.13Hz and another at 96.9Hz.

Reading the non-padded FFT, the actual frequency value of the signal has been misrepresented, due to the frequency resolution not allowing a more accurate representation. Although the non padded transform is still inaccurate it features less channel interference and correctly displays only 1 magnitude.

If 91Hz was a note frequency, and the two FFTs were for the purposes of automatic music transcription, the results of the Fourier analysis would enable a 'closest fit' function. So, although the dominant frequency is 86.13Hz, if it is known the note value is at 91Hz, the result can be altered. This process can be applied to both signals but the noise of the padded FFT is still undesirable as it may interfere or contribute to other note fundamental frequency magnitudes.

Zero padding, although useful should not be used as an alternative to using the longest data length possible - the more data points available for the transform, the more accurate and defined the FFT spectrum output will appear (Quach, 2008). There are arguably benefits to zero padding, but as discussed, there are also associated problems. As Quach states, it is the data length that is of primary importance for an FFT, therefore the investigation in Chapter 7 does not consider zero padding in the evaluation of FFT parameters. It may be possible to include the evaluation of zero padding in future work.

Another method of manipulating the output of the FFT is to alter the window alignment of successive frames of an STFT.

6.11 STFT Hop

The STFT analysis window moves along the waveform after the Fourier transform has been performed on that window of the waveform.

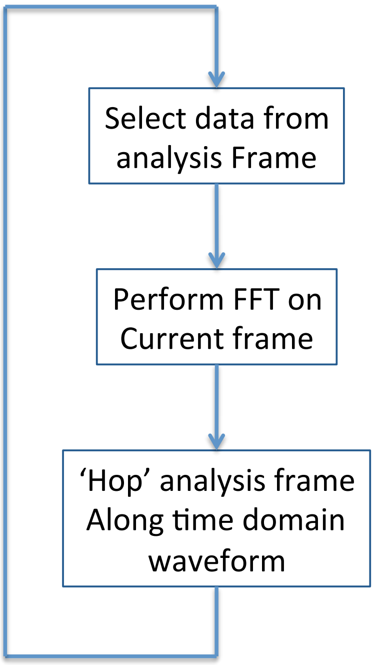

Select data from analysis Frame
Perform FFT on Current frame
'Hop' analysis frame Along time domain waveform

Figure 6-25 Hop Process

The *hop size* refers to the distance the STFT analysis window moves after each STFT. If the hop size is equal to the data length, there is no overlap (Figure 6-25).

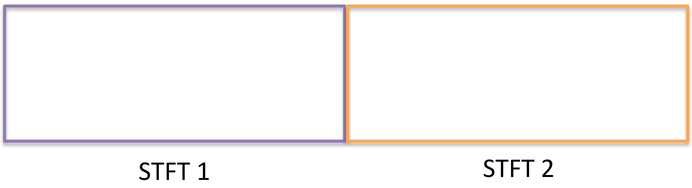


Figure 6-26 100% hop size

If the hop size is smaller than the data length as in Figure 6-27 the overlap does not result in an increased time resolution. In a similar way to zero padding, the overlapping of windows does not provide any additional information. However, interpolation between windows can result in the ability to detect finer details in the time domain, which could be missed without overlapping windows

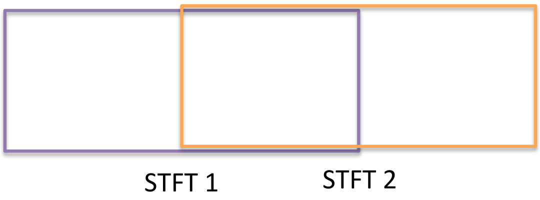


Figure 6-27 50% hop size

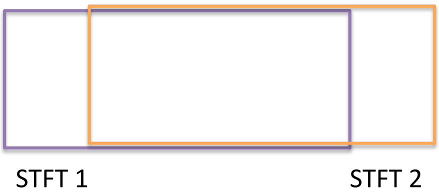


Figure 6-28 25% hop size

The following examples are hypothesised based upon a basic energy based FFT peak picking method where there is no further processing of the FFT data following the initial transform. A basic peak picker will assume the presence of a note if the magnitude of energy within a frequency bin exceeds a threshold.

Figure 6-29 shows a scenario where a hop size equal to the length of the analysis window leads to a larger note length error than using a smaller hop size as shown in Figure 3.1b.

The presented figures are diagramtic representations of hypothesised window alignment and associated energy content. The green blocks represent the original played notes, the red shows the onset time error of the transcribed note, and the blue shows the note length error of the transcribed note. The white boxes represent the STFT analysis windows.

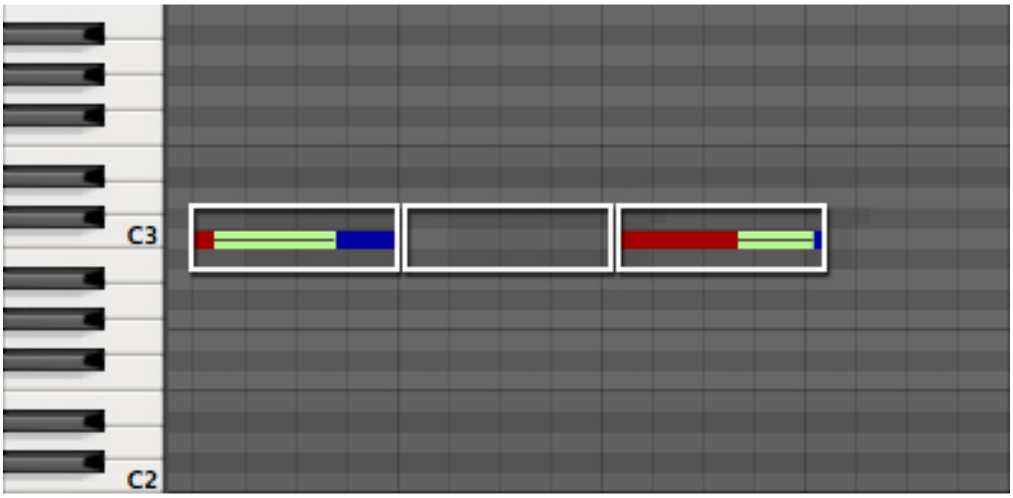


Figure 6-29 – non overlap

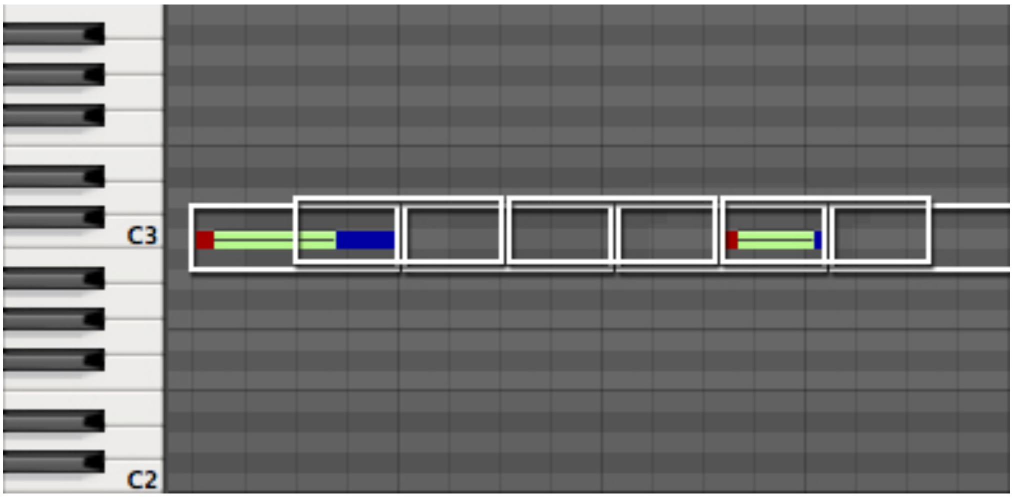


Figure 6-30 overlap

Figure 6-30 shows STFT with a hop size of FFT Length divided by 2, or an over lap of 50%. In this example the first note is transcribed with the same errors using 0% overlap and 50% overlap. The second note would suffer from much larger quanisation errors using the 0% overlap method (Figure 6-29) than the 50% overlap method as the length of the note would be extended to a full frame length rather than 50%.

Figure 6-31 and Figure 6-32 and Table 12 further clarify the process of the 50% overlap shown in Figure 6-30.

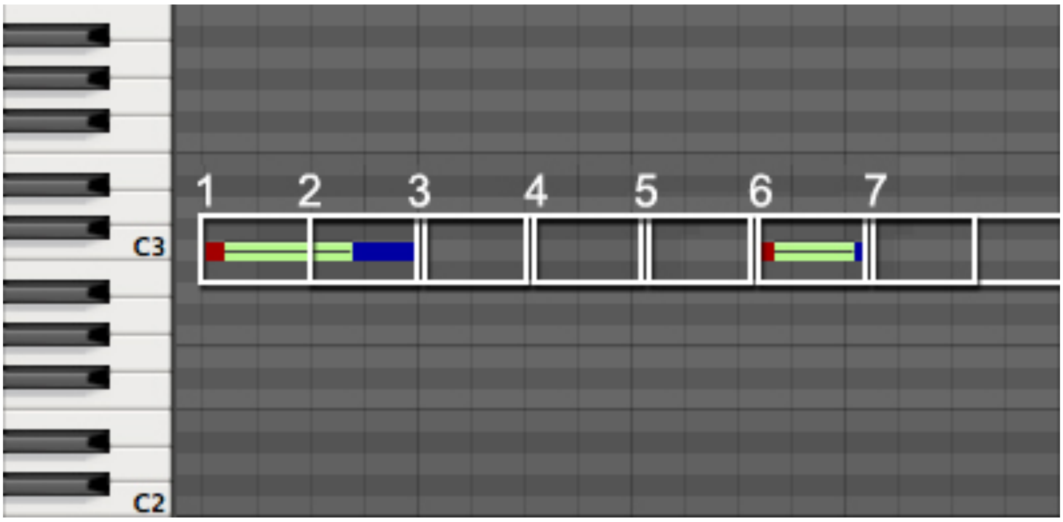
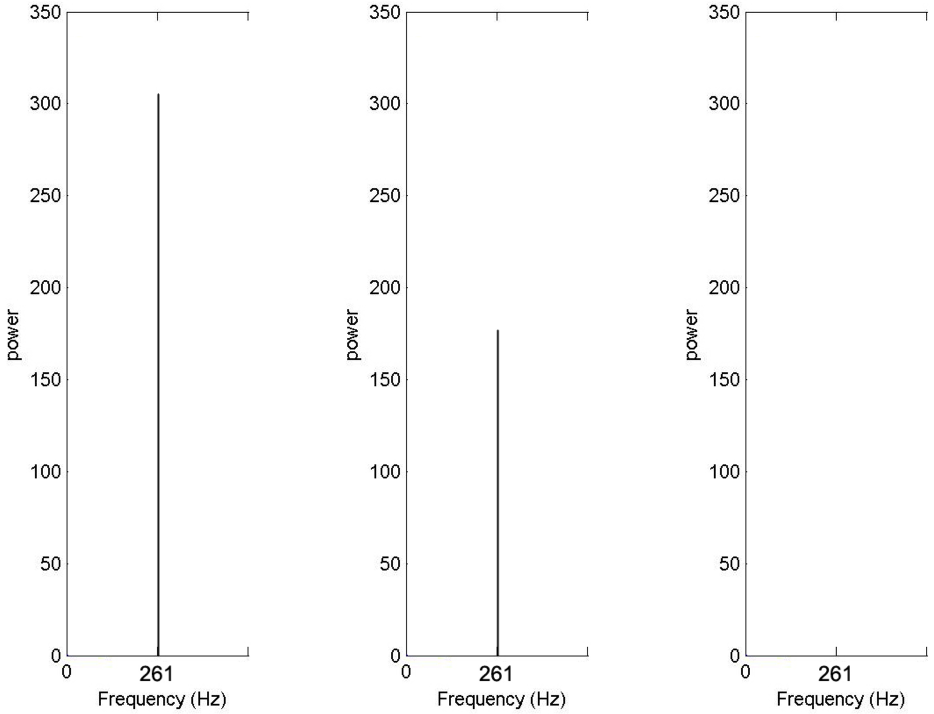


Figure 6-31 Numbered overlapping frames



Frame 1 Frame 2 Frame 3

Figure 6-32 magnitudes present in frames

Frame Number	Note Frequency (Green) Present?
STFT 1	Yes
STFT 2	Yes
STFT 3	No
STFT 4	No
STFT 5	No
STFT 6	Yes
STFT 7	No

Table 12 STFT note presence

Figure 6-32 shows the FFT ouput for frames 1,2 and 3. The note frequency is strongly present in frames 1 and 2, but no frequency compontents are detected in frame 3, therefore, using a basic peak picker it is assumed the note ended at the start time of frame 3. Note that because the note is a smaller proportion of frame 2 than frame 1, the relative power magnitude of that frequency component is smaller also.

It is important to note that utilising an overlapping STFT method does not provide more accurate note timing information for all scenarios. If a single note is longer than the hop size, the timing accuracy is not increased using an overlapping STFT. The transcription of the first green note in Figure 6-29 and Figure 6-30 demonstrates this point.

For the simple case of a single note in a frame, if a note is equal in length to a multiple of the hop size and aligned with the window boundaries, then an accurate note length transcription will be generated. When multiple notes occur in a single frame it becomes more complicated.

6.11.1 Hop Size - Multiple Notes in a Single Frame

If there are multiple notes of the same frequency within a single analysis frame, the Fourier transform will not provide any information to distinguish the two notes as separate events (Figure 6-33).

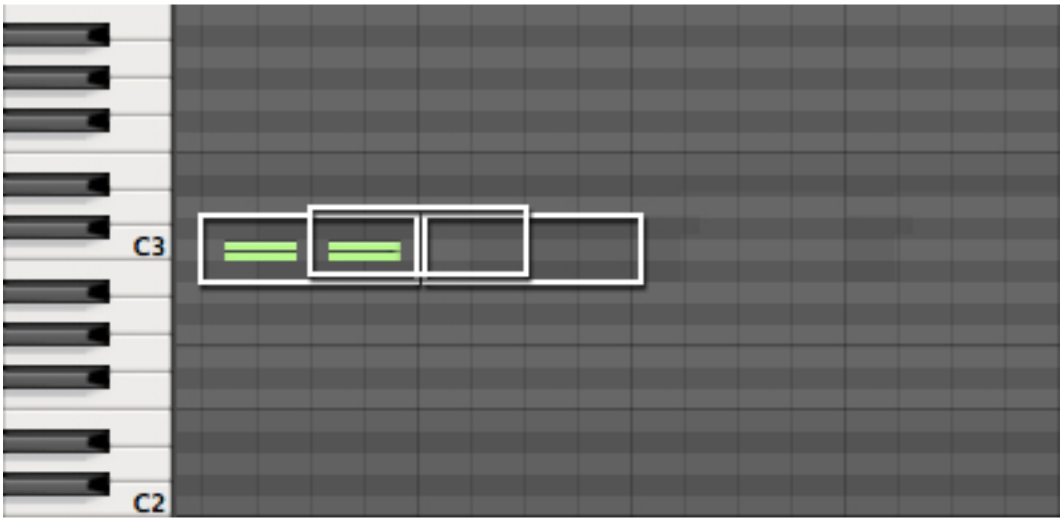


Figure 6-33 Multiple notes of the same frequency

Frame Number	Note Frequency (Green) Present?	Note Start/End
STFT 1	Yes	Start
STFT 2	Yes	
STFT 3	No	End

Table 13 STFT Note presence

The same note frequency is present in both frames 1 and 2 of the STFT, the resulting transcribed note is shown in Figure 6-34.

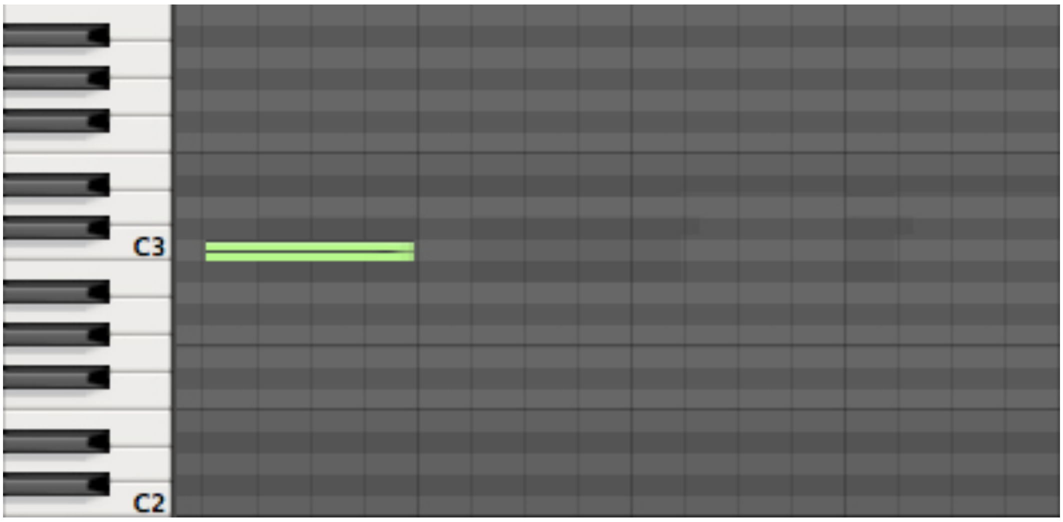


Figure 6-34 Transcription error

To be able to differentiate between 2 notes of the same frequency, the FFT length must be shorter than the gap between the 2 notes. For the purpose of distinguishing 2 notes of the same frequency it is the silence inbetween the notes which is important to detect (Figure 6-35).

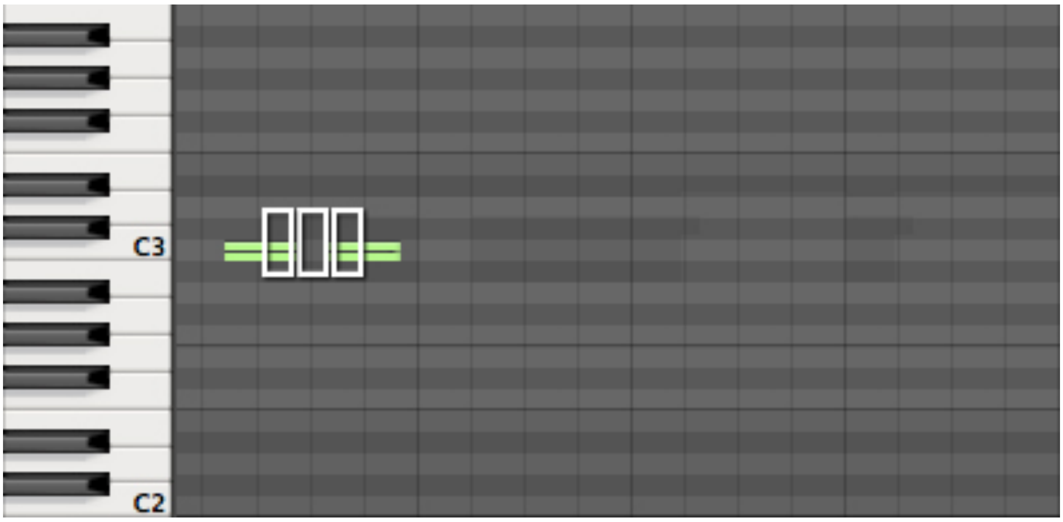
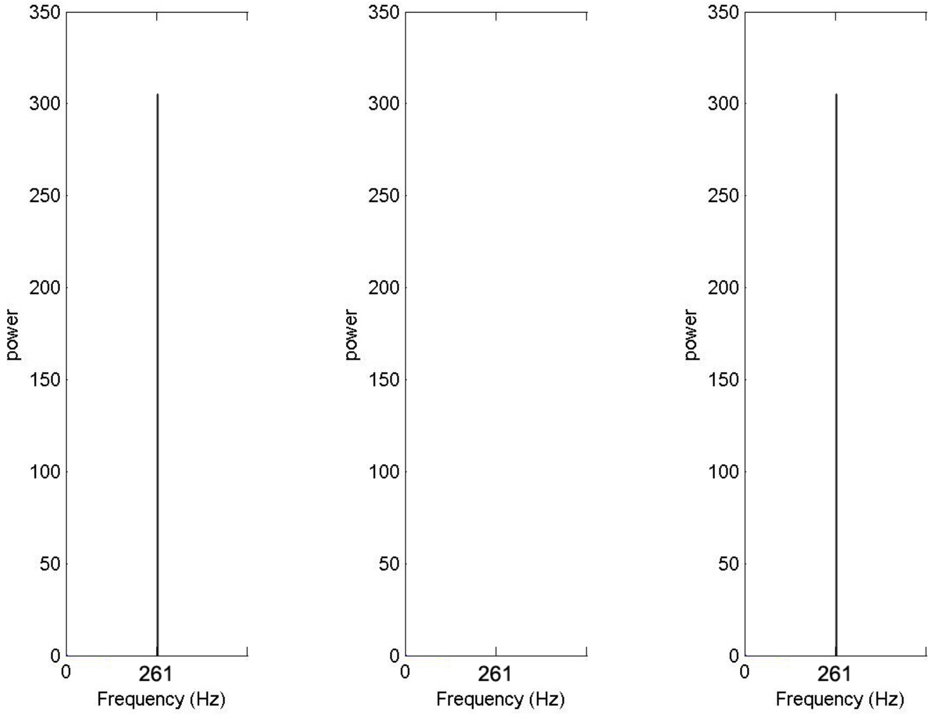


Figure 6-35 Silence detection



Frame 1 Frame 2 Frame 3

Figure 6-36 Silence detection magnitudes

Using a hop size of 50%, if the two notes separated by the gap in Figure 6-33 were of different frequencies, both notes would succesfully be transcribed. However the transcribed notes would be longer than the originals as the silence between the two would not be detected.

The shortest detectable note length is equal to the hop size of the STFT. If the hop size is equal to the FFT length, then the minimum note length is determined by the number of samples in the FFT analysis window. If overlapping windows are utilised the hop size allows for an increase in accuracy of onset times and note lengths, (Figure 6-29 and Figure 6-30).

6.11.2 Minimum Note Length Detectable

The success of overlapping windows to detect short notes depends largely on the position of the note within analysis frames. If the note is positioned so it features in a single window, then an accurate transcription will be possible. Figure 6-37 shows an effective window length (due to the overlap) equal to the length of the note. This will provide a highly accurate transcription as the window length is equal to the note length. However, if the note is positioned so it features in 2 FFT analysis windows, then the transcribed note will be extended in length (Figure 6-38), even though the analysis window is equal to the length of the note.

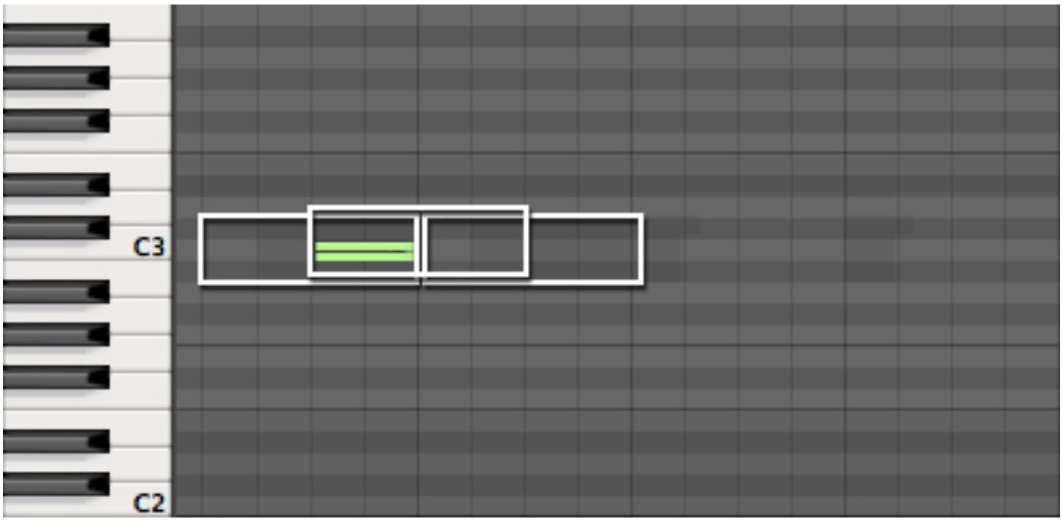


Figure 6-37 An effective window length

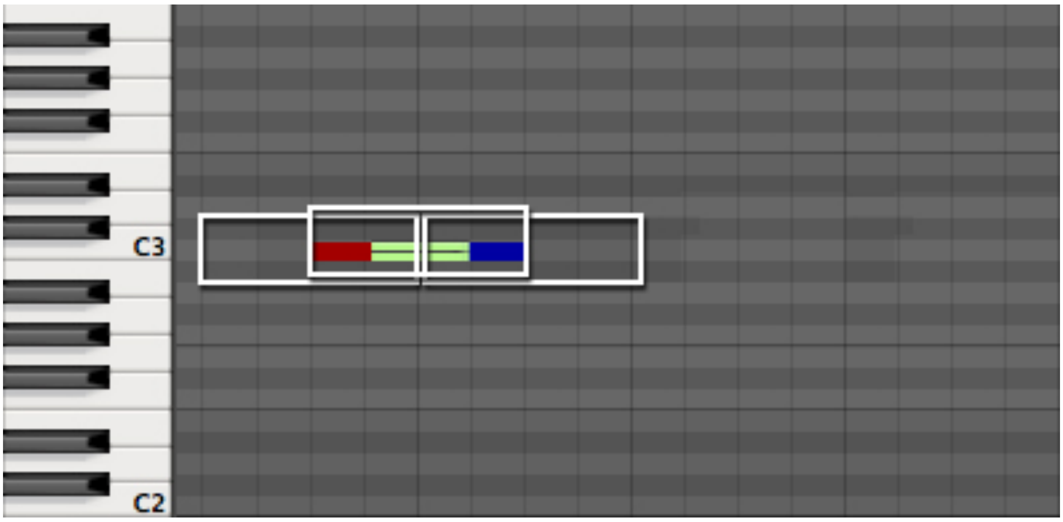


Figure 6-38 Ineffective window length

So while the hop size does determine the shortest note possible to transcribe accurately, the position of the note in the analysis window will determine the actual accuracy. If a note of length equal to hop is position over two frames, the transcribed note will have a length of twice the hop size.

This same scenario is true when considering the smallest length of silence detectable. If a silence between two notes is positioned over two frames, the silence will not be detected (Figure 6-39).

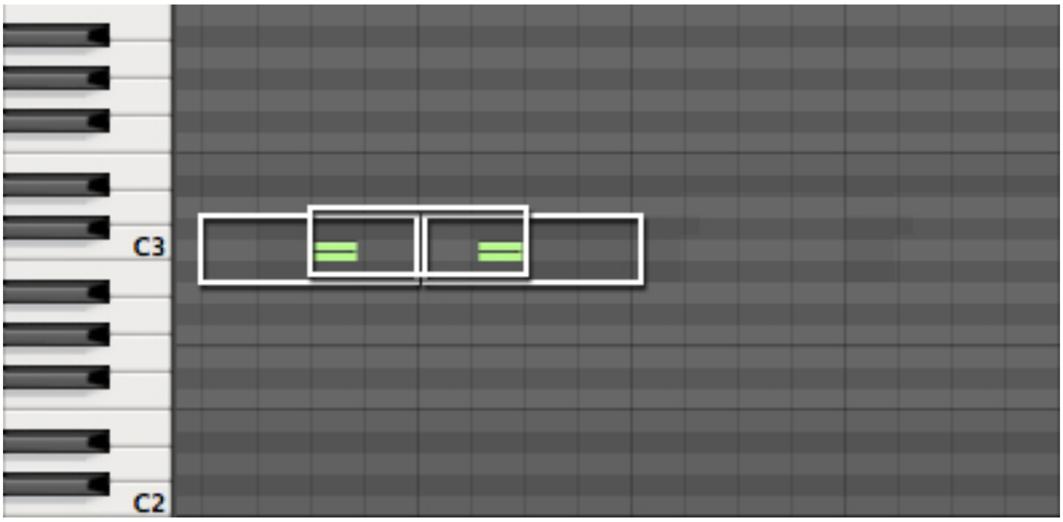


Figure 6-39 Undetected silence

Although the hop size is shorter than the length of silence between the two notes, because the notes are present in the same analysis window the silence will be removed from the transcribed version (Figure 6-40).

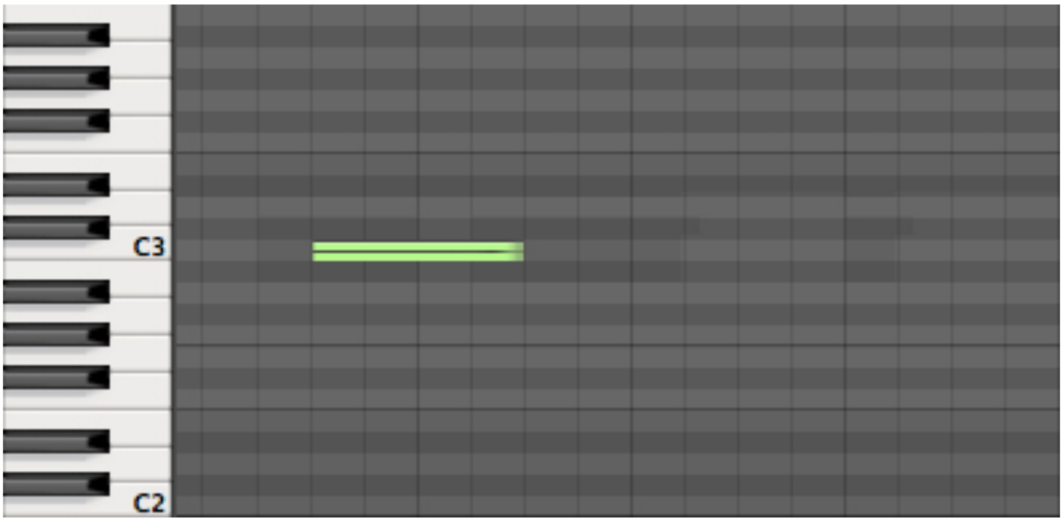


Figure 6-40 2 notes transcribed as one

If the analysis windows are realigned, it is possible to correctly transcribe the silence, although as the notes are shorter than the hop size, they would still be transcribed with timing errors. This is only true if the notes are of different frequencies (Figure 6-41)..

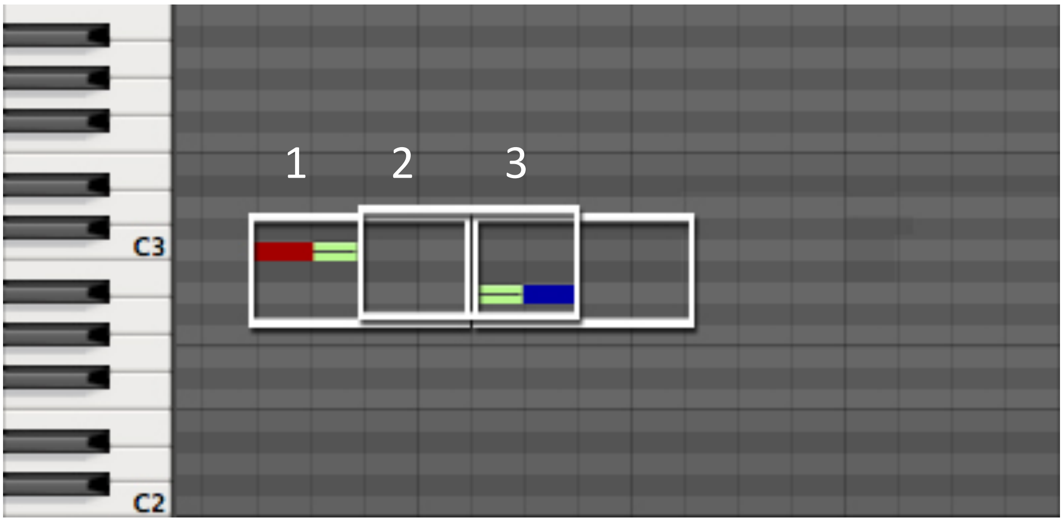


Figure 6-41 Silence maintained

Although frame 2 in Figure 6-41 contains energy from the second note (in frame 3), because that frequency is not present in frame 1 it can be deduced that the 2nd frequency onset does not occur until frame 3. As the first note frequency is not present in frame 2, it is deduced that the first note ends before the start of frame 2. This means the silence between the two notes is maintained.

If the notes are the same frequencies as each other, then the same frequency component will be present in frame 1, 2 and 3 leading to a transcription error where the silence is removed (Shown as the orange band in Figure 6-42).

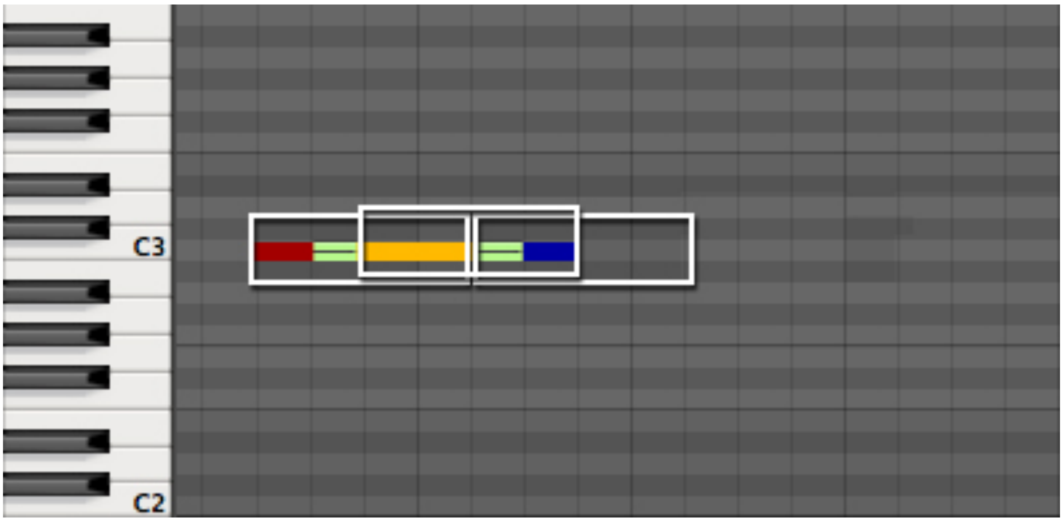


Figure 6-42 Silence is removed

6.11.3 Note Length Errors in Real Terms

The best case scenario for note positioning using an overlapping STFT is a note equal to the hop length, which is positioned in a single analysis frame (Figure 6-43).

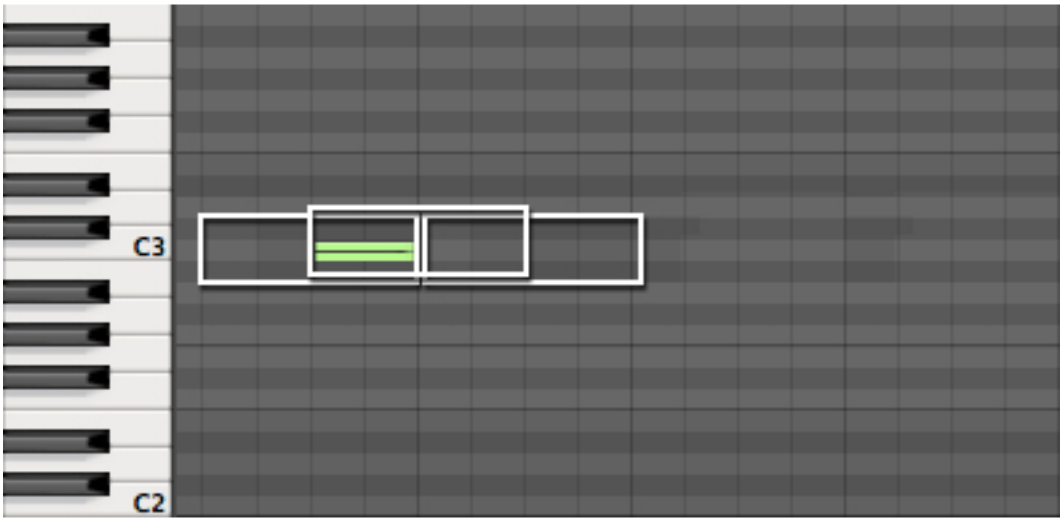


Figure 6-43 Best case scenario

The worst case scenario is a note positioned across the boundaries of analysis windows (Figure 6-44).

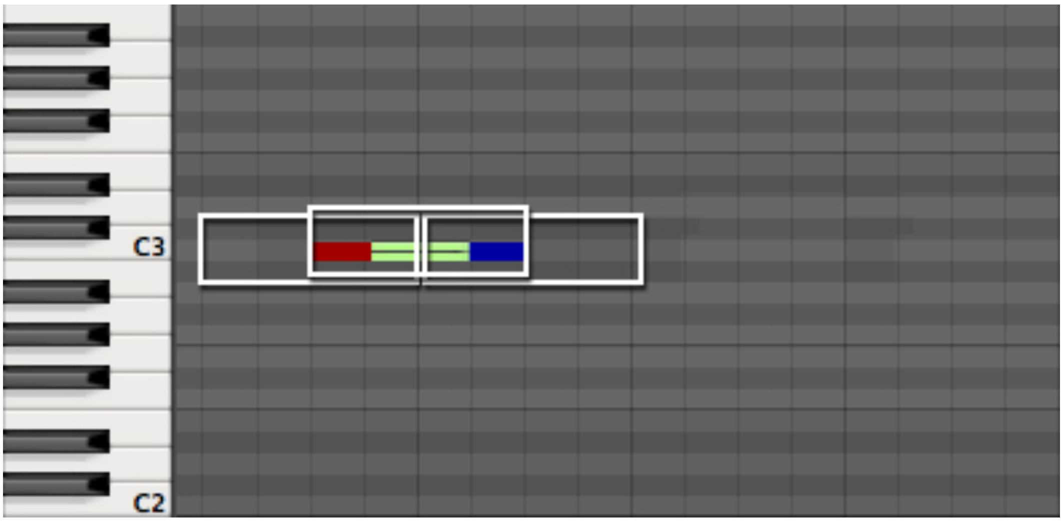


Figure 6-44 Worst case scenario

From this it can be stated that the shortest transcribed note possible will be quantised to be 1 hop length.

If a note is longer than 1 hop length and therefore positioned across *N* analysis window boundaries, then an error *of N* hop lengths minus the original note length can be generated

This conclusion is used to generate real case timing errors. The graphs below show the calculated timing errors of best and worst case transcriptions of quavers and semi quavers at 120 and 200 BPM, based upon an FFT length of 8192 and window overlap of 0,%, 25% and 50%.

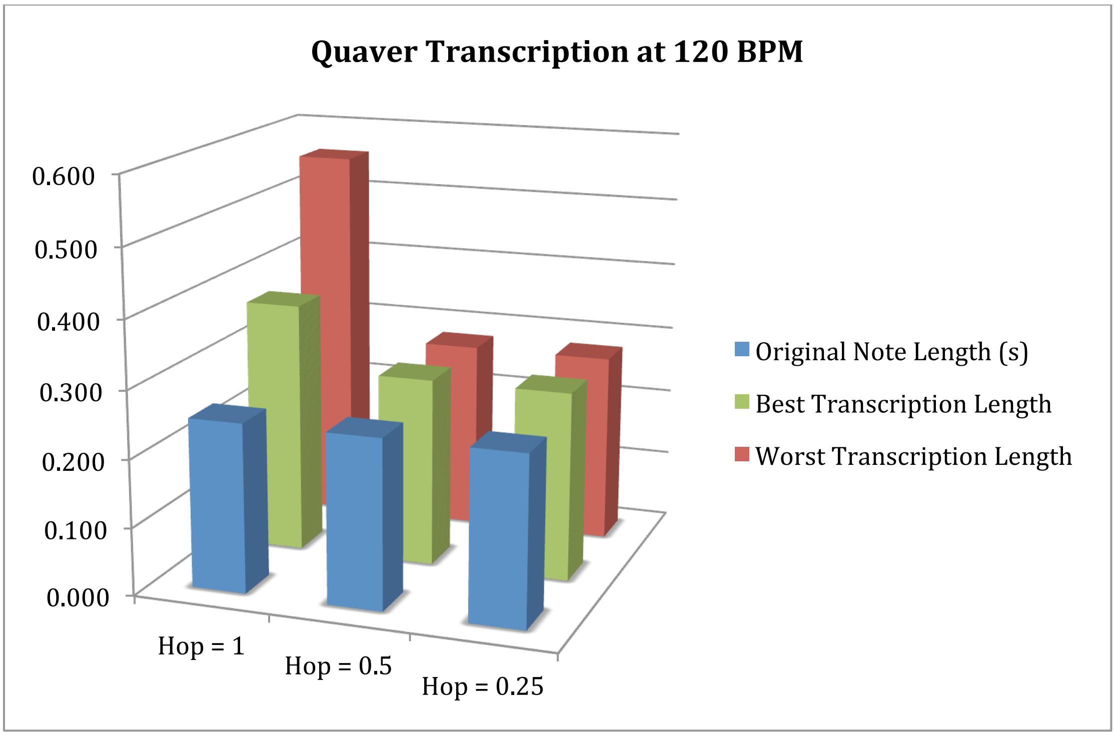


Figure 6-45 Quaver transcription errors

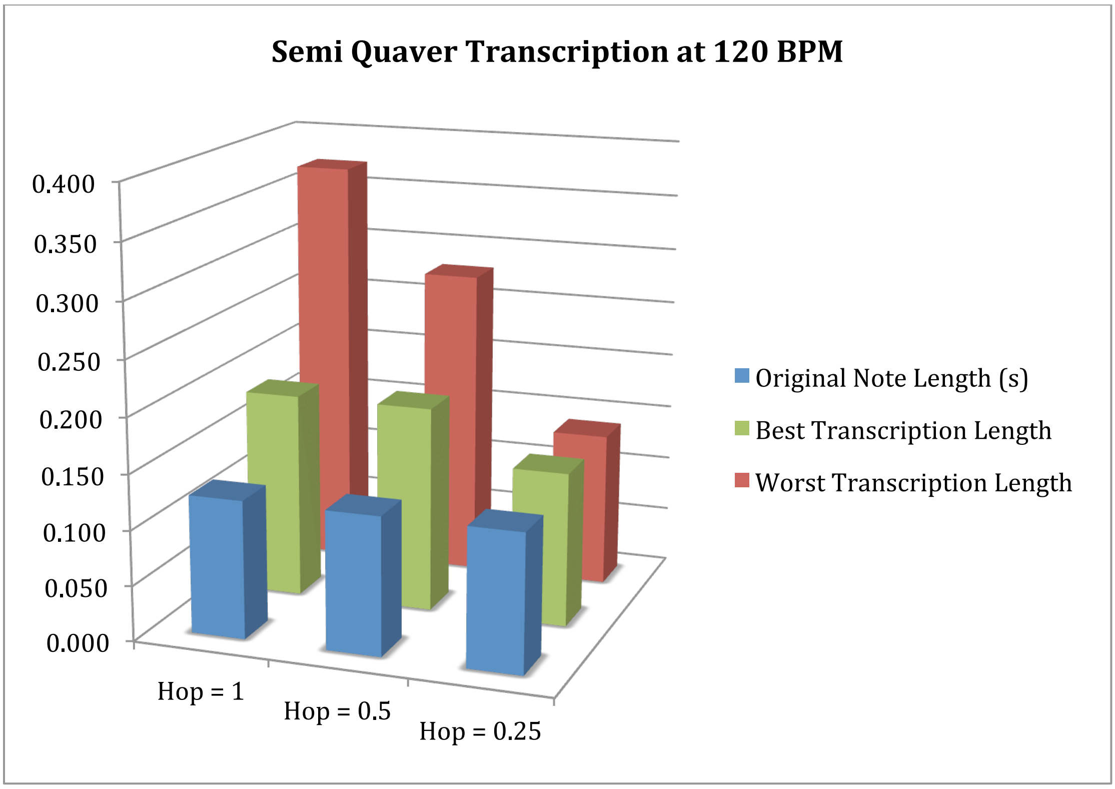


Figure 6-46 Semi quaver transcription errors

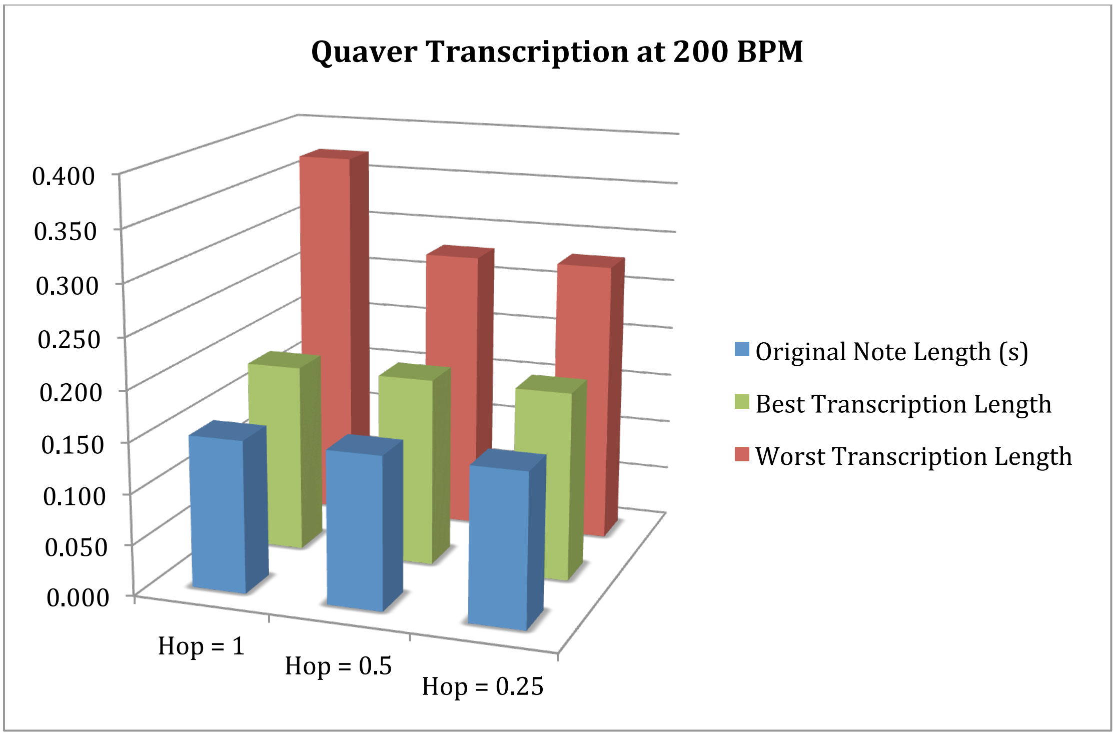


Figure 6-47 Quaver transcription errors at 200bpm

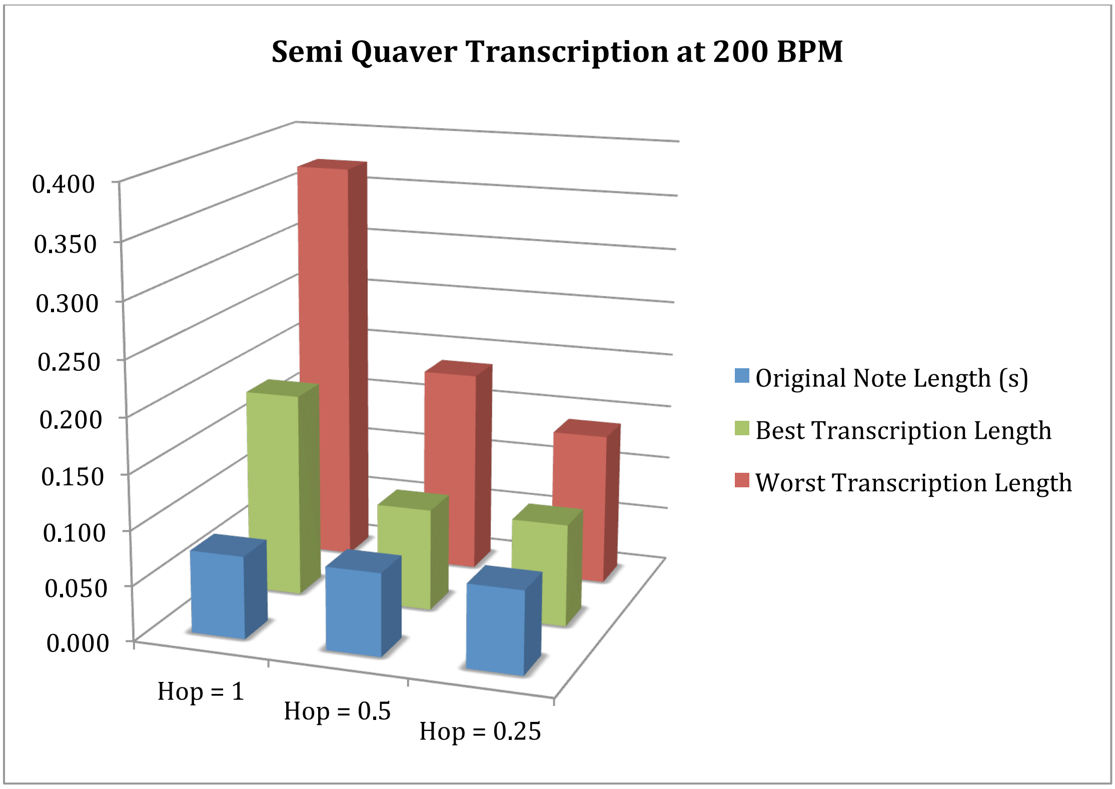


Figure 6-48 Semi quaver transcription errors at 200bpm

The graphs show a decreased hop length generally increases note length transcription accuracy. However, these conclusions are drawn from a simplified theory and not practical experiments. Also they are based on only a single note per anlysis frame.

6.11.4 STFT Hop Summary

Hop length can have positive benefits. While not increasing the actual time resolution of the transform, the interpolated information generated by overlapping the windows can increase the accuracy of note length and onset transcription. This is certainly true for the simple scenarios discussed but potential gains become much harder to calculate and quantify when considering multiple notes in a single window, the variations of spaces between notes, note frequency and note length.

The extent of the positive outcome of altering hop size appears to depend on how the musical content of the signal being analysed aligns with the boundaries of the overlapping windows, and also how the length of the musical notes and silences relate to length and overlap of the analysis windows. As the gains of varying hop size relies in part on the characteristics of the signal being analysed, hop size was not evaluated as part of the investigation in chapter 7. Future work could consider developing a method for evaluating and optimising the hop parameter for music transcription purposes.

6.12 FFT Parameters used by automatic music transcription algorithms

This chapter has discussed the control parameters of the STFT that determine the characteristics and behavior of the STFT output.

As Table 3 demonstrates, the parameters for the FFT used in music transcription algorithms vary. This was a motivating factor to optimise the parameters of the FFT for music transcription.

Having discussed the STFT parameters and observed their impact for practical use in music transcription, a novel method of scoring the performance of STFTs based upon the transform output and suitability for music analysis is introduced in the following chapter.

A set of optimised multiresolution FFT (MRFFT) parameters for use in automatic music transcriptions are generated and presented and tested on sinusoidal extraction tasks. Results are compared with other multiresolution approaches and discussed.

7 Optimisation of FFT Parameters for Automatic Music Transcription

Table 14 is a summary of the low level processes discussed in Chapter 4 that are typically implemented in automatic music transcription algorithms. Table 3 shows that the FFT is the favoured process for frequency domain transformations – despite it's time-frequency resolution trade off and linear frequency response.

	Advantages	**Disadvantages**
Filter Banks	Flexible configuration Simple implementation Flexible filter shapes	Potentially computationally expensive for suitable frequency resolution
STFT	Fast Computationally efficient	Linear frequency response Time-frequency trade off
CQFFB	Fast Constant Q frequency response	Constant Q based on octave divisions Computationally expensive Non reversible
Multirate Filter Banks	Fast Computationally efficient Constant Q Frequency response	Computationally expensive to generate suitable frequency resolution
Wavelets	Multiresolution frequency resolution	Relatively slow Computationally expensive to generate suitable frequency resolution
MRFFT	Multiresolution frequency response Fast Computationally efficient	Not Constant Q FFT Parameters potentially not optimised for transcribing music

Table 14 Low Level Processing comparisons

The motivation for the following investigation is to determine a set of FFT parameters that are optimised for automatic music transcription and evaluate them with commonly used parameters to observe if the optimised parameters offer any improvements.

The outline of this chapter is as follows:

An overview of the optimisation process is presented with justifications for parameters chosen. The search process used to find the optimised MRFFT settings is then presented, including the scoring process for note-bin alignment, frequency resolution, time resolution and overall MRFFT score.

Section 7.7 introduces the 6 'solutions' that are generated from the optimisation process. The results of this optimisation process are then presented and discussed.

A sinusoidal extraction test is performed by all 6 solutions to evaluate their optimised performance. The method and results of this test are presented and discussed, followed by a further investigation motivated by these results, including analysis of quality of note candidates generated by MRFFT solutions.

7.1 A 'tuned' Multiresolution FFT for Automatic Music Transcription

The presented system varied the cut off frequencies for dividing the frequency domain into sub-bands and varied the FFT length used in each sub-band. The algorithm scored the time resolution, frequency resolution and the alignment of equal tempered scale fundamental frequencies with the bin spacing to determine a set of optimised 'tuned' parameters for automatic music transcription.

Zero padding, window shape and hop overlap are not considered in this initial evaluation of parameters. Based upon the conclusions drawn in the sections of Chapter 6 these parameters and manipulations of the FFT output can be beneficial for music transcription algorithms. However, as discussed, there are also drawbacks to each method, so each must be used/selected carefully. The performance of each of these parameters also depends largely on the characteristics and content of the input audio. The methodology of this initial work does not account for the variations of input signals that would affect the performance of zero padding, window shape and hop overlap. Therefore, the system would not be able to evaluate these particular parameters accurately.

The focus of this work is on the essential parameters of multiresolution Fourier transforms that are independent of the audio input. The FFT data length, and the division of sub-bands are the basic parameters for all MRFFTs, so these are evaluated and optimised.

A rectangular window was used for the audio testing of the optimised MRFFT. The presented scoring method doesn't account for evaluating window shapes, and as there is no universally accepted 'best' window shape for audio analysis, the simplest shape was chosen. The rectangular window used provides a worse case scenario in terms of side lobe behavior and cross channel interference, so any improvements seen in resulting magnitudes, cross channel interference and bin alignment can be attributed to the optimised FFT length, rather than an optimised window shape.

The outcomes of this investigation are:

- A tuned FFT length for a single band FFT
- A set of 3 optimum sub-band divisions and an optimised FFT length for each sub-band of a multiresolution FFT
- A set of 4 optimum sub-band divisions and an optimised FFT length for each sub-band of a multiresolution FFT

By optimising the FFT length and sub-band division for automatic music transcription, it is hoped the spectra presented from the resulting FFT will yield more accurate results than parameters currently being used by providing a higher quality of note candidate to the higher level processing. The method for determining optimised FFT lengths and sub-band divisions is based upon an instrument tuned to the equal tempered scale, and therefore whose fundamental note frequencies are in the ratio or $2^{1/12}$.

7.2 The Searching and Scoring of FFT Parameters

A program was developed using Matlab software that cycled all possible combinations of sub-band cut offs (the position in the frequency domain where one band ends and another starts) for a 3 band and 4 band MRFFT.

Figure 7-1 shows the scoring and searching process. The diagram only includes cut off frequencies for simplicity and clarity.

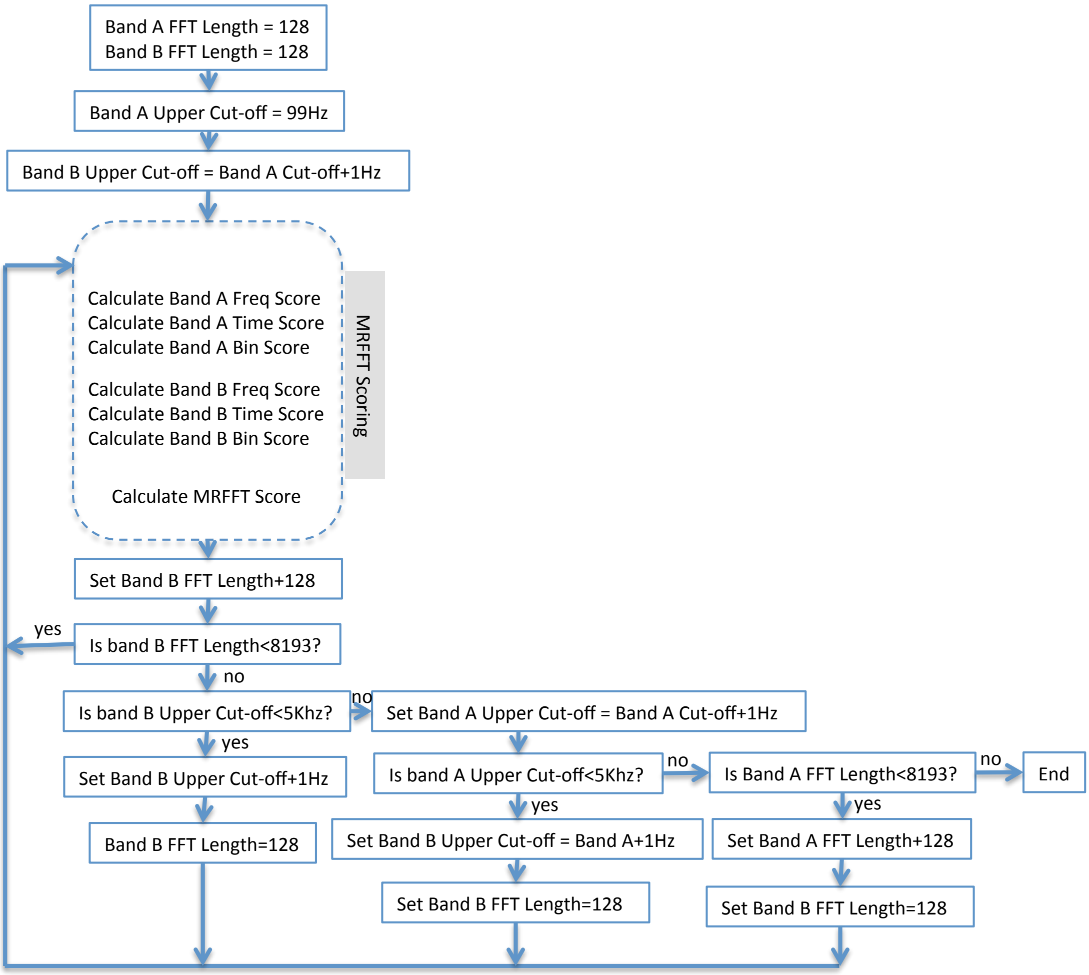

Band A FFT Length = 128
Band B FFT Length = 128
Band A Upper Cut-off = 99Hz
Band B Upper Cut-off = Band A Cut-off+1Hz
Calculate Band A Freq Score
Calculate Band A Time Score
Calculate Band A Bin Score
Calculate Band B Freq Score
Calculate Band B Time Score
Calculate Band B Bin Score
Calculate MRFFT Score
MRFFT Scoring
Set Band B FFT Length+128
yes
Is band B FFT Length<8193?
no
Is band B Upper Cut-off<5Khz?
no
Set Band A Upper Cut-off = Band A Cut-off+1Hz
yes
Set Band B Upper Cut-off+1Hz
Is band A Upper Cut-off<5Khz?
no
Is Band A FFT Length<8193?
no
End
Band B FFT Length=128
yes
Set Band B Upper Cut-off = Band A+1Hz
yes
Set Band A FFT Length+128
Set Band B FFT Length=128
Set Band B FFT Length=128

Figure 7-1 The scoring and search process for a 2 cut-off MRFFT.

The search method was an exhaustive search with the objective of minimising a combined ‘error score’ for frequency resolution, time resolution and note-to-bin alignment. The error score is calculated based upon the difference between the best possible frequency resolution, time resolution and note-to-bin alignment and those generated by the combination of FFT Length and sub band divisions in the MRFFT. The scoring method for these parameters is presented in sections 7.3 to 7.6.

The MRFFT was calculated for every combination of sub-bands. For every combination of sub-band division all FFT lengths from 256 to 8192 samples in increments of 128 samples were applied to each band in all combinations. The resulting note-to-bin-alignment, time resolution and frequency resolution were scored and summed to determine the optimum set of sub-band cut off frequencies and FFT lengths for each band.

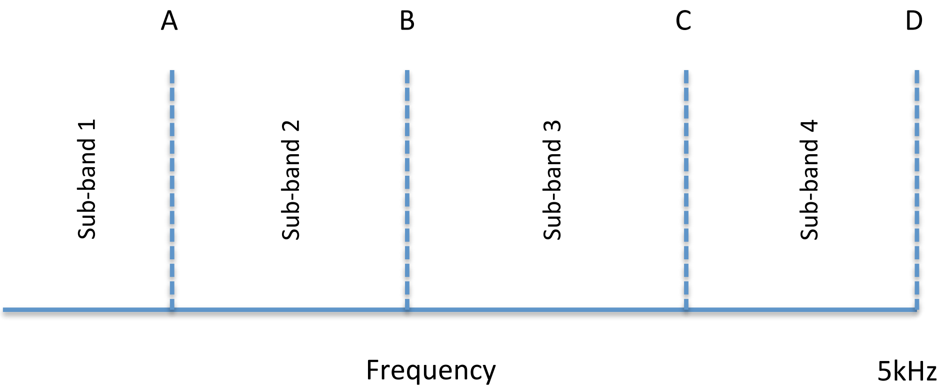


Figure 7-2 Sub-band cut off points

A 4 band MRFFT. The algorithm moves the cut off frequencies A, B and C through all combinations of positions. For each position, all FFT lengths between 256 and 8192 samples in increments of 128 are evaluated on each sub-band. All combinations of FFT lengths on all combinations of subbands are evaluated and scored.

Traditionally, FFT decomposition has been restricted to power of 2 lengths. This was due to the functionality and efficiency of the decomposition method relying upon the number of samples equaling a power of 2 (Duhamel & Vetterli, 1990). Modern computer processors and decomposition methods, such as the Fastest Fourier Transform in the West allow non-power of 2 transform lengths. This flexibility of window length can be utilised to better align note frequencies with bin values within each sub-band.

The following sections describe the parameter scoring methods.

7.3 Bin Scoring

For each set of cut off frequencies and FFT length within each band, the position of the fundamental frequencies of the equal tempered scale within the FFT bins is scored between 1 and 0. For every FFT length the bin values change, so the alignment of fundamental frequencies with bin values alters for every FFT length.

If the fundamental frequency of a note matches the frequency of a bin, it is ideal so the bin is scored with an error score of 0. If a note is positioned at the half way point between bins, the bin is scored 1 – this is the worst score possible. A fundamental frequency exactly between bins will cause significant cross channel interference and not be represented accurately in the FFT spectrum.

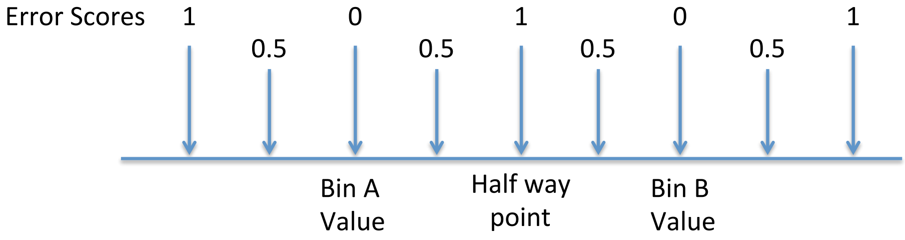


Figure 7-3 Bin scoring method

Error scores assigned to frequencies based upon their position relative to bin values of A and B

If the bin spacing is such that a single bin represents more than a single fundamental frequency, then the bin is scored with a penalty of 1000, thus eliminating that particular FFT length from being considered an optimal length. This penalty is imposed, as it is crucial that a bin will only represent a single fundamental frequency.

The following equations explain the bin scoring process

If Nb = 1

$$b_i = \sum_{j=1}^{N=1} | x_j - y_j |$$

Equation ix

else if Nb >1

$$b_i = 1000$$

Equation x

else if Nb = 0

$$b_i = 0$$

Equation xi

where:

Nb	number of notes in bin
b_i	is the score for the FFT bin
j	is the number of bins in the FFT
x	is the bin frequency
y	is the note frequency

When all notes have been positioned into corresponding bins and those bins have been individually scored between 0 and 1, the scores for all the bins in that transform are summed providing a total score for that FFT length in that sub-band – the *Sub-band FFT Bin Score*.

$$SBs = \sum_{b=1}^{B} x_b$$

Equation xii

where:

SBs is Sub band FFT Bin Score

B is total number of bins in sub band

x_n is bin score

7.3.1 Sub-Band FFT Bin Score – Weighting

The division of the frequency spectrum into subbands results in some subbands containing more note frequencies than others. To ensure a sub-band with a good score but only a single note, or a sub-band with a poor score but lots of notes for example doesn't 'skew' the overall score, the initial FFT bin scores are weighted by the number of notes in the current band relative to the total notes.

$$WSb = SBs\times \left(\frac{X}{Y}\right)$$

Equation xiii

where:

WSb	is Weighted Sub band FFT Bin Score
SBs	is sub-band FFT Bin Score
X	is Notes in Sub-band
Y	is total notes across all bands

The multiresolution FFT bin score is calculated by summing the Weighted *Sub-band FFT Bin Score* for each band in the multiresolution FFT.

$$MRb = \sum_{n=1}^{N} WSb_n$$

Equation xiv

where:

MRb	is the multiresolution FFT bin score
N	is total number of bands in MRFFT
WSb	is Weighted Sub-Band FFT Bin Score

7.3.2 MRFFT Bin Score – Range Correction

To allow the MRFFT bin score, frequency resolution score and time resolution score to be summed into a 'total score', their scores need to be adjusted so they are all within the same range. The MRFFT Bin score is range corrected to be between 1 and 0 by dividing the MRFFT bin score by the total number of notes within the MRFFT. This averaging works as a bin can only score between 0 and 1 for each note unless 2 notes are positioned within the same bin, in which case the FFT length is penalized so wouldn't feature as an optimised solution.

7.4 Frequency Resolution Score

For each FFT length checked in the algorithm, a different frequency resolution is generated. The initial frequency score is simply the FFT frequency resolution.

$$SBf = Fs/L$$

Equation xv

where:

SBf is Sub band Frequency Resolution

Fs is Sample Rate

L is FFT Data Length

7.4.1 Frequency Score - Weighting

The number of notes in the band then weights the frequency score of the FFT used in each band in the same manner as the bin score is weighted.

$$WSf = SBf \times \left(\frac{X}{Y}\right)$$

Equation xvi

where:

WSf	is Weighted Sub Band Frequency Score
SBf	is sub-band Frequency Score
X	is Notes in Sub-band
Y	is total notes across all bands

The MRFFT Frequency score is calculated by summing the Weighted Sub-band Frequency score for each band in the multiresolution FFT.

$$MRf = \sum_{n=1}^{N} WSf_n$$

Equation xvii

where:

MRf	is MRFFT Frequency Score
N	is total number of bands in MRFFT
WSf	is Weighted Sub-Band Frequency Score

7.4.2 MRFFT Frequency Score – Range Correction

As the FFT lengths are limited to be between 256 and 8192 samples long, the 'best' and 'worst' frequency resolutions are decided by these values. The MRFFT Frequency score is range corrected to be between 0 and 1 using the following formula.

$$RcMRf \;=\; 1 - \left(\frac{MRf - \left(\frac{Fs}{A}\right)}{\left(\frac{Fs}{B}\right) - \left(\frac{Fs}{A}\right)} \right)$$

Equation xviii

where:

RcMRf	is Range Corrected MRFFT Frequency Score
MRf	is MRFFT Frequency Score
Fs	is Sample Rate
A	is shortest FFT Data Length
B	is largest FFT Data Length

The '1 minus' at the start of the equation is to adjust high frequency resolution values to be closer to 0, as zero is ideal. In this case a frequency resolution generated by an 8192 FFT length would be assigned a score of 0, as that resolution is the best that can be produced in this model.

7.5 Time Resolution Score

For each FFT length checked in the algorithm, a different time resolution is generated. The initial time score is simply the FFT time resolution.

$$SBt = L/Fs$$

Equation xix

where:

SBt is sub band Time Resolution

Fs is Sample Rate

L is FFT Data Length

7.5.1 Time Score - Weighting

The number of notes in the band then weights the time score of the FFT used in each band in the same manner as the frequency score is weighted.

$$WSt = \mathrm{SBt}\times \left(\frac{X}{Y}\right)$$

Equation xx

Where:

WSt is weighted Sub band time score

SBt is sub band Time Resolution

X is Notes in Sub-band

Y is total notes across all bands

The MRFFT Time score is calculated by summing the Weighted Sub-band Time score for each band in the multiresolution FFT.

$$MRt = \sum_{n=1}^{N} WSt_n$$

Equation xxi

where:

MRt is MRFFT Time Score

N is total number of bands in MRFFT

WSt is Weighted Sub-Band Frequency Score

7.5.2 MRFFT Time Score – Range Correction

As the FFT lengths are limited to be between 256 and 8192 samples long, the 'best' and 'worst' time resolutions are decided by these values. The MRFFT time score is range correct to be between 0 and 1 using the following formula

$$RcMRt = 1 - \left(\frac{MRt - \left(\frac{A}{Fs}\right)}{\left(\frac{B}{Fs}\right) - \left(\frac{A}{Fs}\right)} \right)$$

Equation xxii

where:

RcMRt is Range Corrected MRFFT Time Score

MRt is MRFFT Time Score

Fs is Sample Rate

A is shortest FFT Data Length

B is largest FFT Data Length

7.5.3 The MRFFT Score

The MRFFT score is equal to:

$$Score = \frac{MRb + RcMRf + RcMRt}{3}$$

Equation xxiii

where:

MRb is the multiresolution FFT bin score

RcMRf is Range Corrected MRFFT Frequency Score

RcMRt is Range Corrected MRFFT Time Score

The MRFFT Score is a value between 0 and 1, 0 being best and 1 being worst. The MRFFT Score is calculated for every combination of sub-band divisions with every combination of FFT Lengths.

The algorithm presents a 'best' MRFFT for each set of sub-bands. These results provide the best FFT length to use in each sub-band created by each set of cut-off frequencies. The combination of cut off frequencies and FFT length that generate the lowest MRFFT Score is selected as the optimal MRFFT.

7.6 Scoring Restrictions

The FFT lengths considered were limited to be between the range of 256 and 8192 samples. This reflects the range of FFT lengths commonly used for music transcription algorithms.

As the FFT lengths considered were limited, the frequency range of notes used to generate scores was also limited. This was to ensure the frequency resolution of the longest transform didn't exceed the smallest delta frequency. Applying this restriction ensured it was possible for every note to be positioned within a bin without sharing it with another fundamental frequency. It is crucial a bin does not represent more than 1 fundamental frequency.

The frequency resolution of a 8192 FFT at 44.1kHz sample rate accommodates the delta frequency between a note at 98Hz and the next note at 103Hz. Therefore the range of notes was restricted be between 98Hz and 5kHz. 5kHz accounts for the full note range by exceeding the fundamental frequency range of virtually all popular western instruments.

7.7 Testing Parameters

Results for 6 different transforms are presented. They are:

1) **4 Band MRFFT 98Hz- 500Hz F Range** – a 4 band MRFFT optimised by scoring all sub-band divisions and FFT lengths between 256 and 8192 for note values between 98Hz and 5000Hz. This MRFFT was generated as being the optimised 4 band MRFFT.

2) **256-8192 3 Band MRFFT** – a 3 band MRFFT designed by scoring all sub-band divisions and FFT lengths between 256 and 8192 for note values between 98Hz and 5000Hz. This MRFFT was generated as being the optimised 3 band MRFFT.

3) **Actual Dressler Bands and FFT lengths** – based on the parameters presented by Dressler (Dressler, 2006). The note frequency range is limited to 369Hz – 5000Hz due to the 256-2048 FFT lengths used by Dressler. Dressler's work is highly referenced and is becoming a popular MRFFT. It is therefore a useful benchmark to compare against

4) **Dressler FFT Length, Variable Bands** – FFT values were limited to 2048, 1024, 512 and 256, but the division of the sub-bands was flexible. This is a variation of Dressler's method to determine if it can be further optimised by maintaining the power of 2 FFT lengths, but changing the cut-off frequencies.

5) **256 – 2048 FFT Limit** – FFT values were limited to be between 256 and 2048 and the division of sub-bands was flexible. The restriction of FFT length allows a direct comparison to Dressler's solution

6) **8192 1 Band FFT** – a single band 8192 FFT.

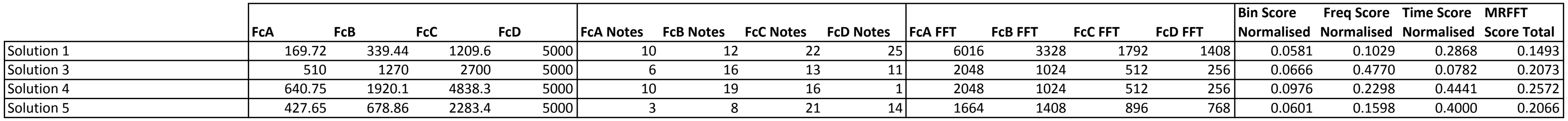
Solutions 1 and 2 are original. Solution 3 is an evaluation of Dressler's proposed MRFFT. Solutions 4 and 5 are variations of 3, and solution 6 is for the purpose of providing a single band comparison to the multi-resolution solutions.

7.8 Optimised MRFFT Results

The optimised MRFFT parameters resulting from the described scoring system are presented in the tables below.

Each MRFFT solution is listed in a table showing the cut off frequencies for the number of bands used (FcA, FcB, FcC, FcD), the number of notes present in each band (FcA Notes, FcB Notes, FcC Notes, FcD Notes),which is used for the weighting of results, the FFT length used in each of those bands (FcA FFT, FcB FFT, FcC FFT, FcD FFT), and then the normalized Bin Score, Frequency Score, Time Resolution Score, and finally the MRFFT score for the solution.

7.8.1 Solution 1,3-5 Optimisation Results

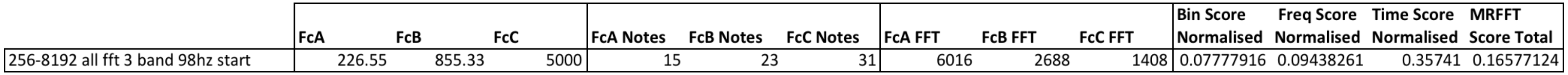


													Bin Score	Freq Score	Time Score	MRFFT
	FcA	FcB	FcC	FcD	FcA Notes	FcB Notes	FcC Notes	FcD Notes	FcA FFT	FcB FFT	FcC FFT	FcD FFT	Normalised	Normalised	Normalised	Score Total
Solution 1	169.72	339.44	1209.6	5000	10	12	22	25	6016	3328	1792	1408	0.0581	0.1029	0.2868	0.1493
Solution 3	510	1270	2700	5000	6	16	13	11	2048	1024	512	256	0.0666	0.4770	0.0782	0.2073
Solution 4	640.75	1920.1	4838.3	5000	10	19	16	1	2048	1024	512	256	0.0976	0.2298	0.4441	0.2572
Solution 5	427.65	678.86	2283.4	5000	3	8	21	14	1664	1408	896	768	0.0601	0.1598	0.4000	0.2066

Table 15 Solution 1,3-5 optimisation results

Solution 1 contains 69 notes in total compared to 46 for solutions 3,4 and 5. This is due to the restricted frequency range determined by the shorter maximum FFT length. Results are still comparable however as scores are biased based upon proportions of notes present in each band relative to the total.

7.8.2 Solution 2 Optimisation Results



									Bin Score	Freq Score	Time Score	MRFFT
	FcA	FcB	FcC	FcA Notes	FcB Notes	FcC Notes	FcA FFT	FcB FFT	FcC FFT	Normalised	Normalised	Normalised	Score Total
256-8192 all fft 3 band 98hz start	226.55	855.33	5000	15	23	31	6016	2688	1408	0.07777916	0.09438261	0.35741	0.16577124

Table 16 Solution 2 optimisation results

7.8.3 Solution 6 Optimisation Results

	FcA	FcA Notes	FcA FFT	Bin Score Normalised	Freq Score Normalised	Time Score Normalised	MRFFT Score Total
8192 1 Band FFT	5000	69	8192	0.259977	0	1	0.41999233

Table 17 Solution 6 optimisation results

Figure 7-4 shows the sub-band divisions as described by Dressler in solution 3, and the divisions generated for solutions 1,2,4 and 5. The areas labeled ‘Not Analysed’ are the note frequencies not considered for that solution. This is due to the maximum FFT length frequency resolution not being high enough to resolve the delta frequencies of the lower notes. For this investigation the ideal of not having multiple notes in a single bin is upheld, but two notes in a single bin could possibly resolved by high level processing.

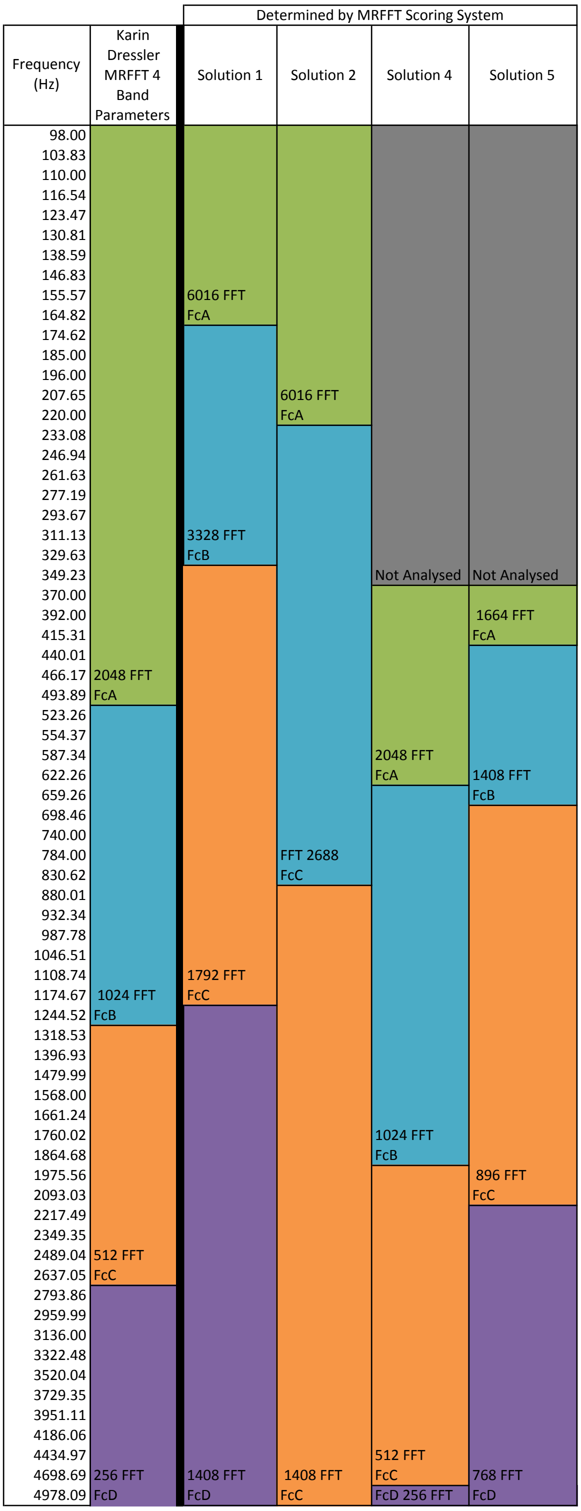


Figure 7-4 Sub-band divisions

Figure 7-5 shows the Bin Score, Time score, Frequency score, and the total MRFFT score for each set of parameters presented.

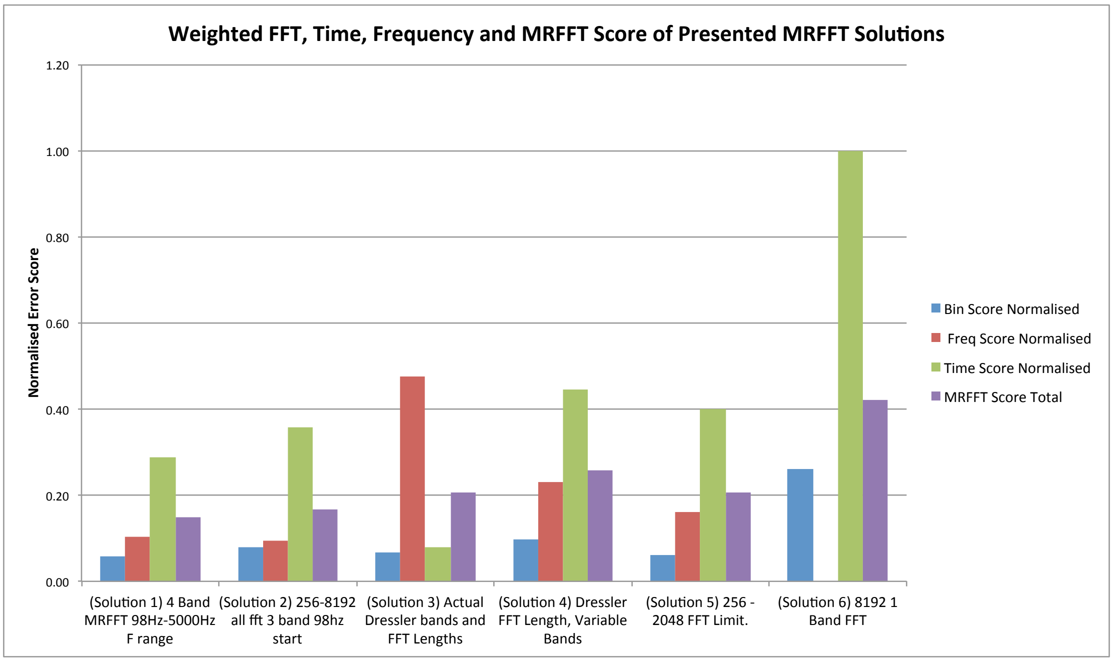


Figure 7-5 Optimisation results graph

A low bin score indicates that a MRFFT will generate strong note candidates with minimal cross channel interference.

A low frequency score indicates a good frequency resolution so frequency representations in the FFT output will be accurate.

A low time score indicates a good time resolution, which indicates the MRFFT will be able to accurately place a frequency event in the time domain by minimizing note length and onset/offset errors.

7.8.4 Optimisation Results Discussion

The interplay between FFT length and the time and frequency resolution is relatively simple to predict, as demonstrated by the extreme nature of the time and frequency scores for single band 8192 FFT that demonstrates the ultimate trade-off. The less predictable measure is the positioning of the fundamental frequencies relative to the bin frequencies generated by the FFT length.

Based upon the optimisation results presented, the optimised 4 Band MRFFT is the most 'tuned' MRFFT for music transcription by generating the lowest MRFFT error score (0.149). The 3 band MRFFT out performed the Dressler parameters also, but predictable faired worse than a 4 band MRFFT.

Dressler's MRFFT generates a low time resolution error score, which is in contrast to the other solutions presented. The time-frequency trade off is clear in the frequency error score however, as it is significantly larger than the other MRFFT solutions. Despite this the weighted bin score, which is related to the spacing of the bin frequencies, is comparable to the 4 band MRFFT that has a much lower frequency error score. The MRFFT Score of the Dressler transform is outperformed by solution 5 (256 – 2048 FFT range with optimised cut offs and FFT lengths)), although in reality there is very little difference with a delta MRFFT score of 0.001 between the Dressler MRFFT and solution 5. Solution 5 favours an improved frequency resolution than the Dressler MRFFT, but the time score suffers for it.

It is interesting to note that the 3 band MRFFT transform scored better in the optimisation evaluation than the 4 band Dressler MRFFT. This suggests the transcription results from the 3 band MRFFT could be an improvement on those generated by Dressler's at the reduction of a sub-band.

As expected, all solutions preferred a longer FFT length in the low frequency, and a shorter FFT length in the high frequency sub-bands.

Solution 3, the Dressler MRFFT actually fares well in the scoring system being only slightly worse than a 4 band MRFFT using different sub-band divisions and non power of 2 FFT lengths (solution 5). The increased maximum FFT length from 4096 to 8192 can be attributed to why the 3 band MRFFT generates a lower error score than any of the 256-4096 MRFFT variants presented.

Dressler's solution favours a stronger time resolution property over the frequency resolution. This suggests solution 3 will perform better than the other solutions with regards to accurately transcribing note onset times and note lengths.

The division of sub-bands (Figure 7-4) shows minimum variation, but there is a pattern of the highest frequency band of the optimised solutions being extended into lower frequencies than compared to Dressler in solution 3. This suggests the cut off frequencies Dressler uses are not optimised for the positioning of fundamental frequencies relative to bin frequencies. This is reflected in the slightly lower Bin Score given to solution 3.

7.8.5 Weighting Desirable Parameters

The optimised 4 Band MRFFT demonstrates a closer to ideal frequency resolution than time resolution. One method to further enhance and optimise the FFT scoring system is to weight desirable criteria. If a scenario required an improved time resolution property, the scoring method could be altered to prefer solutions with good time resolution properties to frequency and bin scores.

Figure 7-6 shows the original optimised 4 Band MRFFT, and a solution generated with a weighting on the time resolution score.

The range adjustment formula for the time resolution calculation was altered by a factor of 2 as in Equation xxivx.

$$Range\ Corrected\ MRFFT\ Time\ Score = \frac{1 - \left(\frac{W - \left(\frac{X}{Fs}\right)}{\left(\frac{Y}{Fs}\right) - \left(\frac{X}{Fs}\right)}\right)}{2}$$

Equation xxiv

Where:

W is MRFFT Time Score

Fs is Sample Rate

X is shortest FFT Data Length

Y is largest FFT Data Length

	FcA	FcB	FcC	FcD	FcA Notes	FcB Notes	FcC Notes	FcD Notes	FcA FFT	FcB FFT	FcC FFT	FcD FFT	Bin Score Normalised	Freq Score Normalised	Time Score Normalised	MRFFT Score Total
Solution 1 - Time weighted	170	339	960	5000	10	12	18	29	6016	3328	1664	1152	0.0702	0.1258	0.2662	0.1541
Solution 1	170	339	1210	5000	10	12	22	25	6016	3328	1792	1408	0.0581	0.1029	0.2868	0.1493

Table 18 Time Weighted 4 band MRFFT Solution

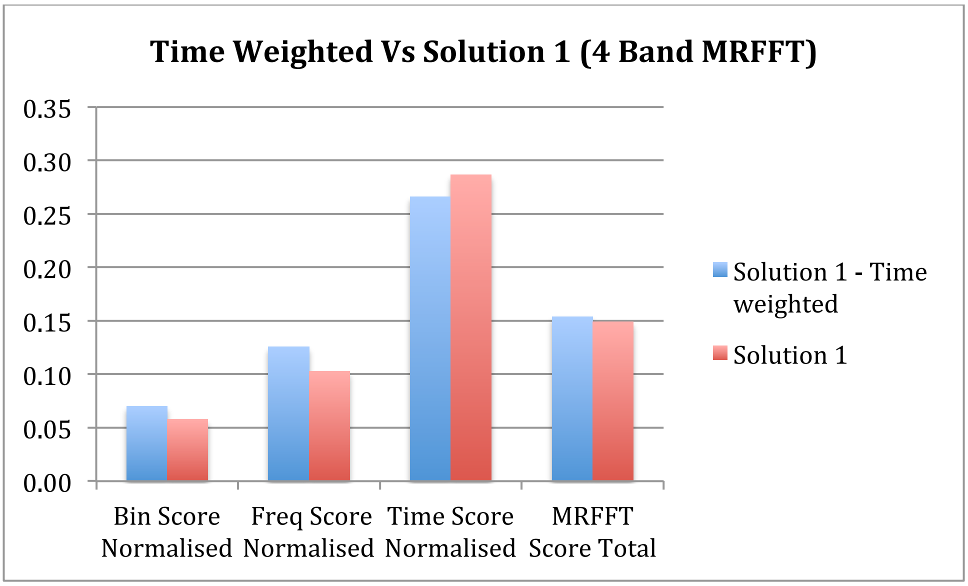


Figure 7-6 Time weighted 4 Band MRFFT comparison

As expected, the time error score is decreased and predictably the frequency error score is increased due to the weighting. Less predictable is how the weighting affects note to bin alignment. The Bin Error Score generated by the weighted solution is increased compared to solution 1, yet there is very little difference between the two MRFFT total scores. These results suggest the optimisation process can be further tuned to reflect a particularly desirable property without fully compromising on the other properties.

7.8.6 Single Band Optimisation

In response to the results of Solution 6, all FFT lengths were evaluated on a single frequency band contain 69 notes between 98Hz and 5000Hz. The optimised single band FFT is 6016 samples at 44.1Khz. The scores for this are compared to the solution 6 scores in Figure 7-7.

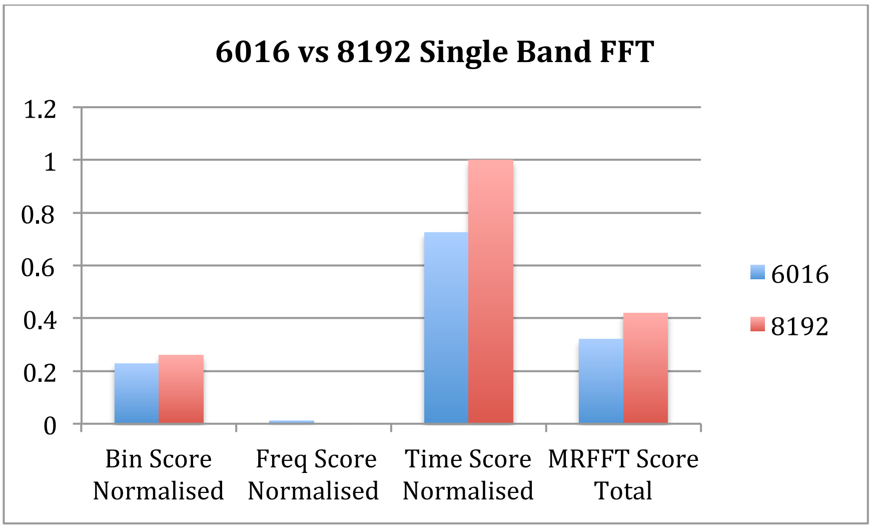


Figure 7-7 Optimised single band FFT

These results suggest that the 6016 FFT length will provide a more accurate transform result across the 3 properties of frequency resolution, time resolution and note to bin alignment. The time resolution score is improved significantly for little frequency trade off. However, the total MRFFT score is still poor compared to a multiresolution solution.

7.8.7 Optimisation comparisons and expectations

Table 19 Rates each presented solution in terms of rank to easily compare the optimised performance of each solution for time resolution, frequency resolution, note-bin alignment and overall MRFFT.

	Ranking (1=best, 6=worst)				
Solution	Time Resolution Rank	Frequency Resolution Rank	Note-Bin Alignment Rank	MRFFT Rank	Average Rank
1) 4 Band MRFFT	2	3	1	1	1.75
2) 3 Band MRFFT	3	2	4	2	2.75
3) Dresslers MRFFT	1	6	3	4	3.50
4) Dressler FFT Length, Optimised Bands	5	5	5	5	5.00
5) 256 – 2048 FFT range (Authors Optimised parameters)	4	4	2	3	3.25
6) 1 Band 8192 FFT	6	1	6	6	4.75

Table 19 - MRFFT Solution Performance Ranking

Dressler's MRFFT is the highest ranking performer for time resolution due to its shorter FFT lengths. This will result in more accurately placed note candidates in the time domain. The optimised 4 band MRFFT and 3 Band MRFFT are 2nd and 3rd. They feature longer FFT lengths as they are also optimised for frequency and note-bin alignment. As expected, the single band FFT is the lowest ranked in terms of time resolution due to it's relatively long fixed 8192 FFT length. However, due to this it ranks as the best solution for frequency resolution meaning it should identify a frequency accurately and produce a corresponding note candidate. The 3 band and 4 band MRFFT solutions are ranked2nd and 3rd which is an improvement over Dressler's MRFFT which is ranked as 6th. The optimised 3band and 4 band MRFFT solutions have found a 'mid ground' in the time-frequency trade off, where as Dressler's solution is weighted more towards an improved time resolution.

The 4 Band MRFFT is top ranking for note-bin alignment meaning it's note candidates should be stronger with less cross channel interference when compared to the other solutions. However, when looking at the actual results as shown in Figure 7-5 there is minimal difference between solutions 1 – 5, with only the single band MRFFT showing a particular lack of optimisation.

The 4 Band MRFFT solution is ranked as the best optimisation so it is anticipated it will produce the highest quality note candidates.

The MRFFT scoring shows solution one to be theoretically optimised for automatic music transcription algorithms based upon the 3 parameters chosen.

7.9 Time domain and F-measure Testing

The aim of the time domain and F-measure testing is to validate the search process and fitness function has optimised the MRFFT performance.

The secondary testing performed on the presented sets of MRFFT involved a simple automatic transcription task. The purpose of this testing is to evaluate whether the theoretical optimisation of the MRFFT translates into performance improvements. In a similar method to Diniz et al. (Diniz F. , Kothe, Netto, & Biscainho, 2007) and Dressler (Dressler, 2006), the MRFFT was implemented as the front end low level processor for a sinusoidal extraction exercise.

A monophonic sine wave is used as the source audio to minimize variants and test the raw basic performance of each solution. Although this doesn't represent true audio transcription tasks it enables the solutions to be tested in a way that should practically demonstrate their characteristics as described by the theoretical optimisation results. Using a basic audio source will also attribute any issues or errors to the MRFFT process rather than the quality or complexity of the test audio.

Two files of chromatically climbing quaver length notes composed across the full frequency range for each presented solution were synthesized as sinusoids from a MIDI file. The MIDI files used to generate the audio files are used as a ground truth for the evaluation of the transcription. 1 file had a BPM of 120; the other had a BPM of 200.

BPM	Length of quaver (s)	length of 1 cycle of 98Hz (s)	Number of cycles of 98Hz in a quaver
100	0.15	0.01020408	14.7
200	0.075	0.01020408	7.35

Table 20 Note cycles at 98Hz

Table 20 shows the number of cycles of a 98Hz wave (lowest note analysed) contained in a quaver at 100bpm is 14.7, and so exceeds the minimum 10 cycles required for a human. At 200 bpm, the number of cycles is reduced to 7.35. This is below what a human requires to identify a pitch, so as a musical example it is unlikely that notes as low as 98Hz would be sounded as short as a quaver at 200bpm. As an example though it still provides an indication of how a system will fair transcribing shorter notes at a fast speed.

An algorithm was written using Matlab to automatically transcribe the audio files and output a generated MIDI file based upon the transcription. The main function of the algorithm was a simple energy based peak picker. A threshold is dynamically set for each analysis window of the STFT as a percentage of the maximum magnitude within the window, with a minimum threshold heuristically decided. If a bin magnitude exceeds the threshold a note is transcribed at that point. The code for this program is included in appendix 11.1

An evaluation function developed by Tavares et al (Tavares, Barbedo, & Lopes, 2008) was implemented to evaluate the transcribed MIDI file against the original MIDI file by generating recall, precision and F-measure results.

Figure 7-8 shows a simplified flow diagram of the process performed by the transcription algorithm.

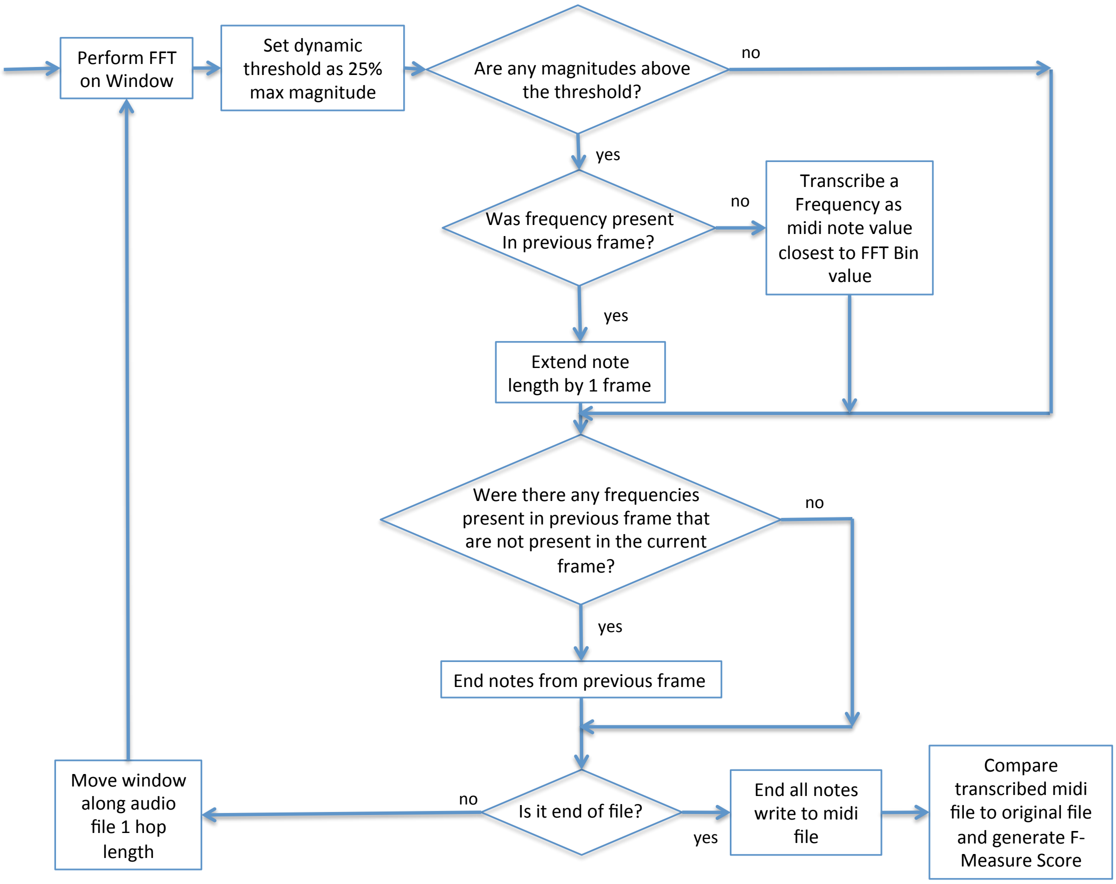

Perform FFT on Window
Set dynamic threshold as 25% max magnitude
Are any magnitudes above the threshold?
no
yes
Was frequency present In previous frame?
no
Transcribe a Frequency as midi note value closest to FFT Bin value
yes
Extend note length by 1 frame
Were there any frequencies present in previous frame that are not present in the current frame?
no
yes
End notes from previous frame
Move window along audio file 1 hop length
no
Is it end of file?
yes
End all notes write to midi file
Compare transcribed midi file to original file and generate F-Measure Score

Figure 7-8 Audio to midi transcription program

7.9.1 Time Domain and F-measure Results

Figure 7-9 and Figure 7-10 present the results for this evaluation of the MRFFT parameters.

Recall refers to the fraction of the relevant notes that were retrieved i.e. how many of the correct notes the system extracted.

Precision refers to the fraction of relevant notes retrieved, relative to the total number retrieved i.e. how many of the extracted notes that were correct.

F-Measure is the weighted mean of precision and recall.

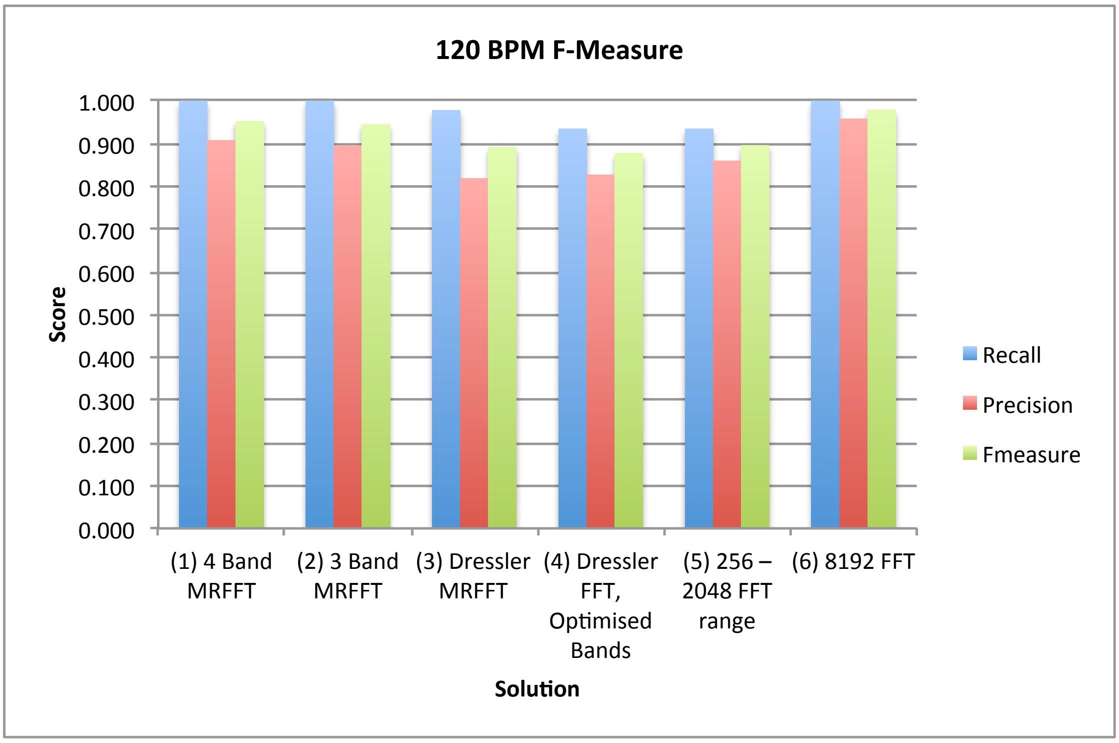


Figure 7-9 120BPM F-measure results

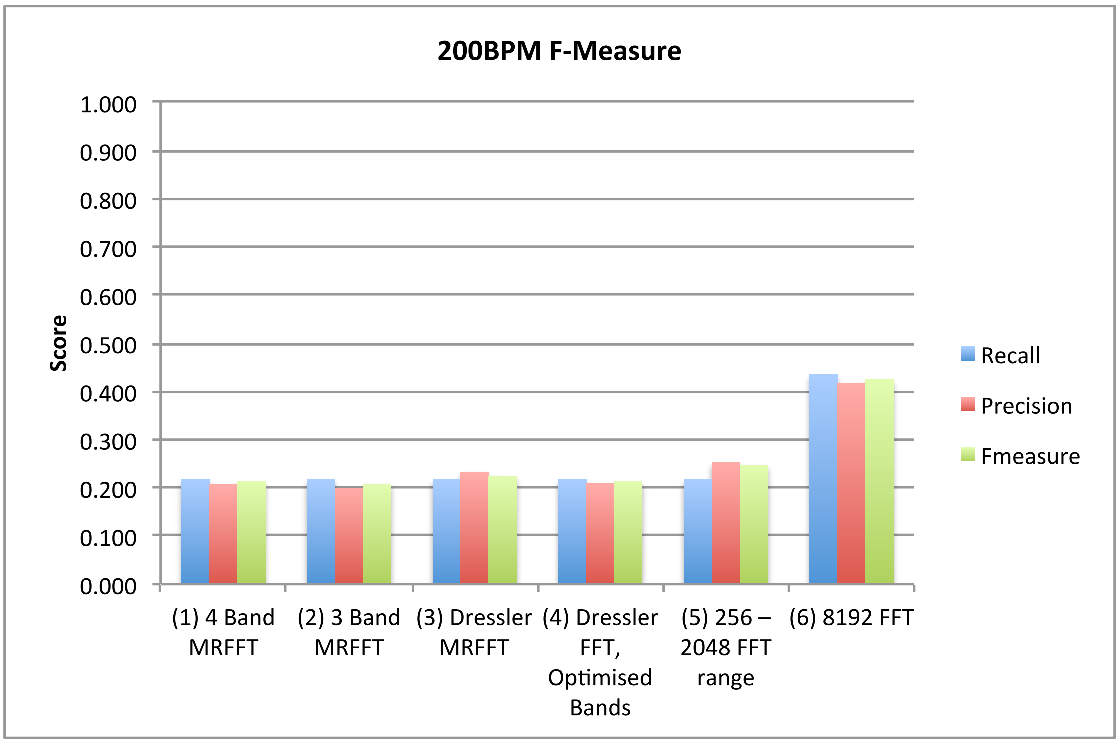


Figure 7-10 200BPM F-measure results

7.9.2 Time Domain and F-measure Discussion

The results for the F-Measure testing are disappointing in terms of the MRFFT performance, and upon further analysis they indicated issues that require further investigation. The expected performance of the MRFFT solutions based upon the optimisation results outlined in section **Error! Reference source not found.** has not materialized in the transcription results.

The F-measure results for the 120BPM file are all very positive and similar for each solution tested. Solutions 1 and 2 achieved slightly higher F-Measure scores than solutions 3,4 and 5, which indicates the optimisation has been successful. However, the margin of improvement is minimal. The high performance of the 8192 FFT – scoring the highest F-Measure result is an interesting result, which initially suggests the MRFFT solutions offer no real world benefits. However, the perceived optimisation of the 8192 FFT can be explained upon further analysis of results in section 7.10

The results for the 200BPM transcription show a disappointing performance of the MRFFT solutions upon initial analysis. Solutions 1 to 5 all score considerably lower than in the 120 BPM task.

Analysis of the transcribed MIDI files show that the lowest octaves of notes are particularly poorly transcribed in solutions 1 to 5.

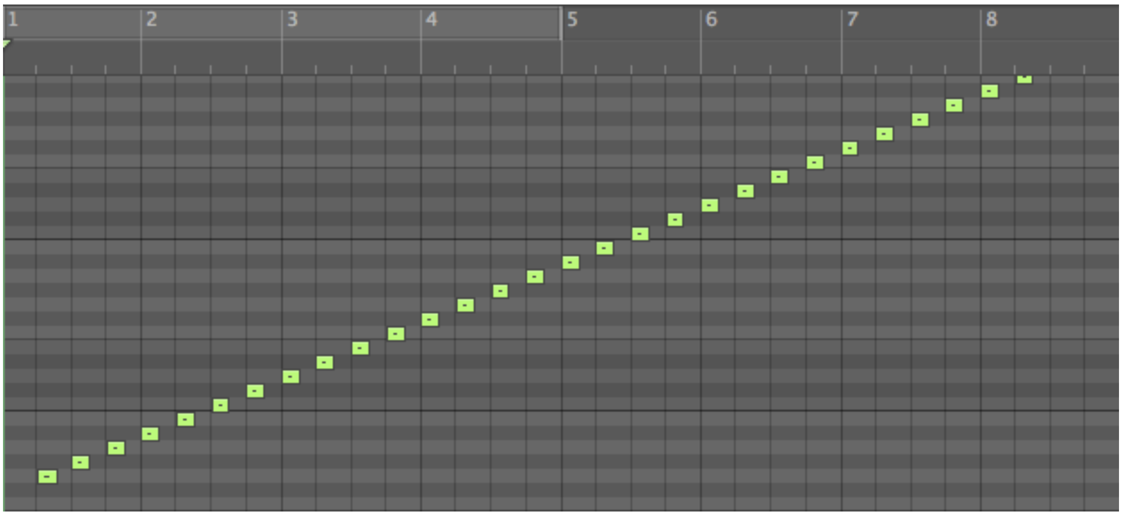


Figure 7-11 Original MIDI file

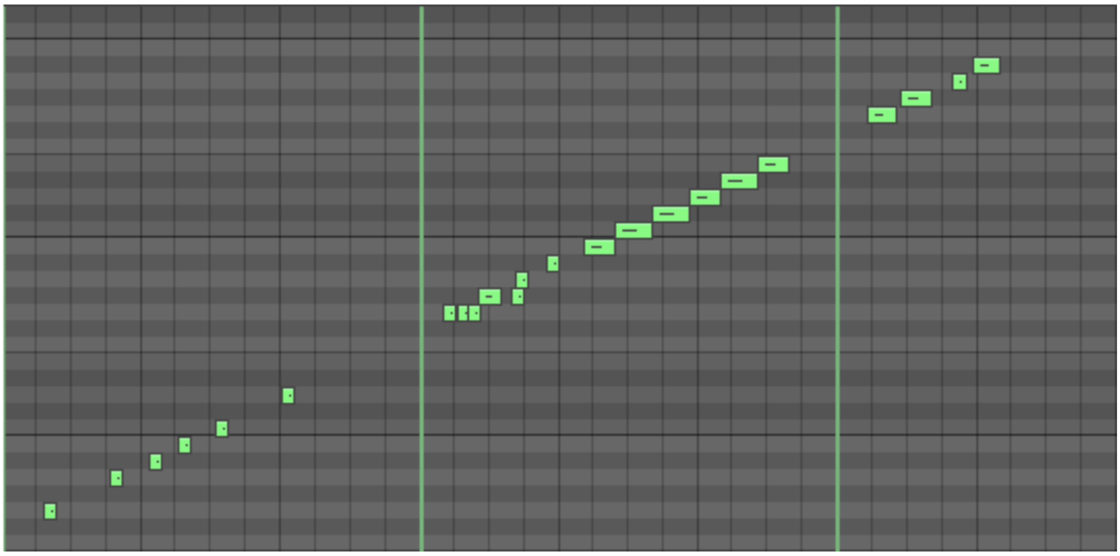


Figure 7-12 Solution 1 (4 Band MRFFT) 200BPM transcription

The vertical green lines in Figure 7-12 are the band cut off points. The lower 2 bands are shown in full. The missing notes in the lower band suggest the magnitudes generated by solution 1 in the low frequencies are not optimised for the peak picker. This is contrary to the results of the optimisation process, which suggested the longer FFT length and optimised note-bin alignment score would generate accurate note candidates.

Solution 6 performs the most accurate transcription. This is not expected based upon the optimisation results where a single band suffered from poor time resolution and note-bin alignment. The high performance of the single FFT despite this suggests that even with a high BPM, the emphasis should be on a longer FFT length and therefore a higher frequency resolution. Figure 7-13 shows the solution 6 transcription of the lower frequencies at 200BPM.

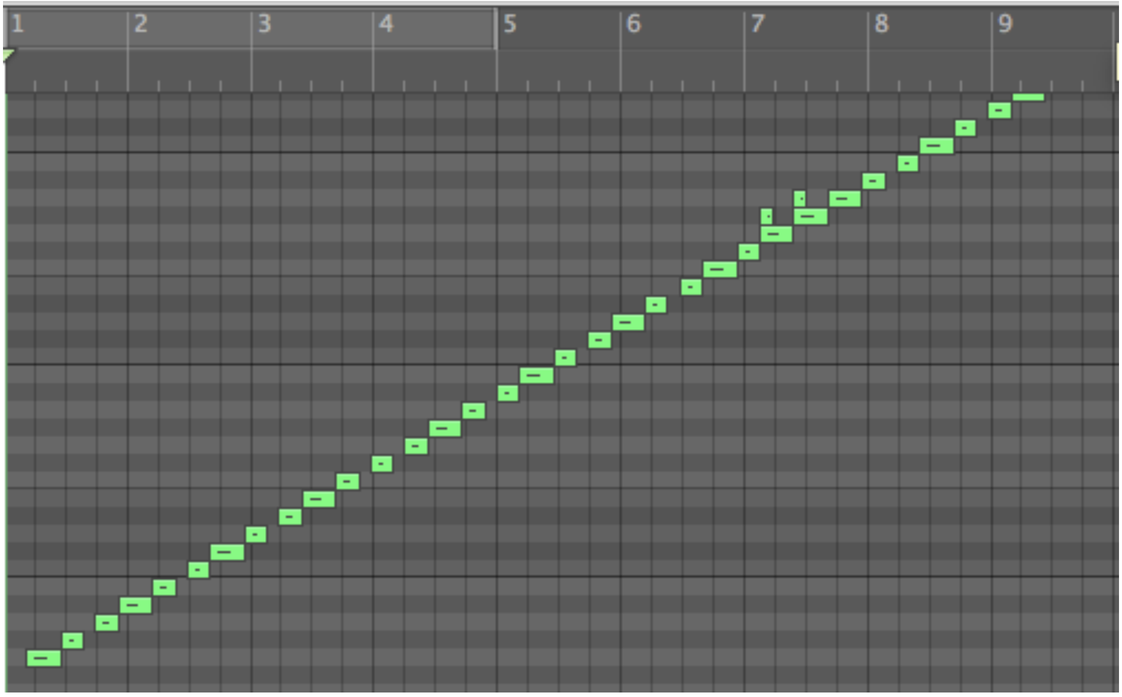


Figure 7-13 Solution 6 (1 Band 8192 FFT) 200BPM transcription - low frequencies

Compared to the original in Figure 7-11 solution 6 shows relatively few errors. The errors that do occur are mainly time based errors where notes are longer. This can be attributed to the lower time resolution of the 8192 FFT length, which is highlighted in the optimisation results.

Observation of the higher band of frequencies of the 4 band MRFFT (solution 1) suggests that as the delta frequencies become wider, transcription accuracy increases. Upon analysis of the transcribed midi data it can be concluded that it is the low extremes of the frequency range that are particularly poorly transcribed and significantly contribute to the low F-measure score. This is true for solutions 1 to 5.

Figure 7-14 shows a largely accurate transcription of the higher frequencies by the 4 band MRFFT (solution 1). This transcription improves on the transcription of the single band FFT (solution 6) for the same range (Figure 7-15). These transcriptions meet the expectations based upon the optimisation scores i.e. the MRFFT solution demonstrates a more accurate time resolution, and the note candidates (based upon the transcription process) are of a higher quality.

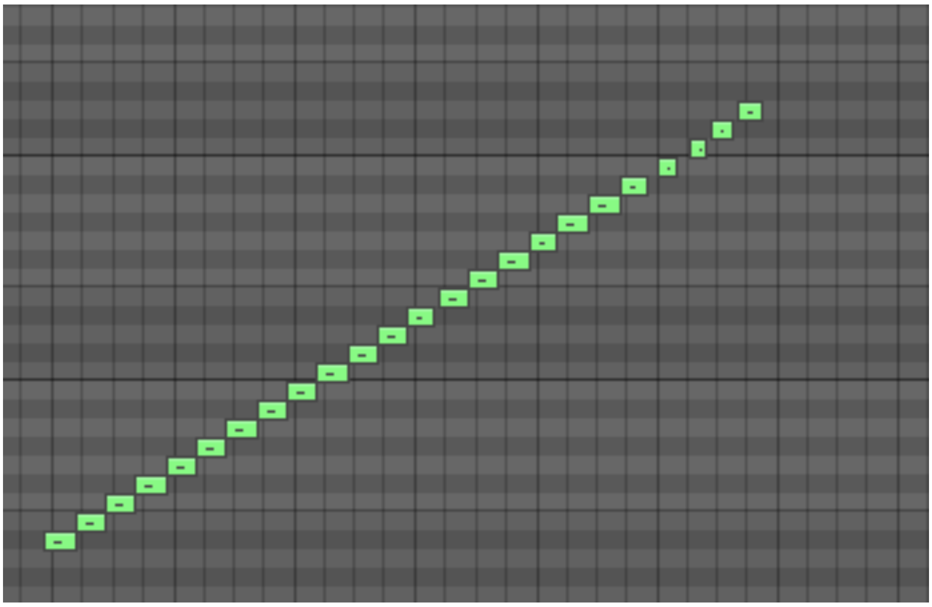


Figure 7-14 Solution 1 (4 Band MRFFT) transcription - high frequencies

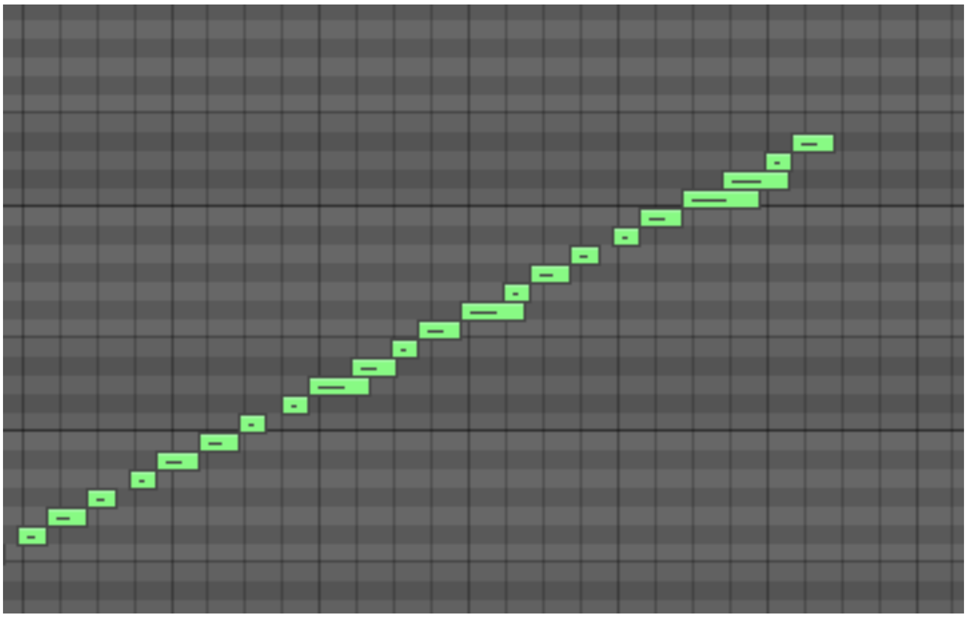


Figure 7-15 Solution 6 (1 Band 8192 FFT) transcription - high frequencies

The transcription by Solution 3 (Figure 7-16) demonstrates an accurate transcription of the time domain in terms of note length, as the scoring system suggested, but faired less well in the frequency domain compared to solution 1 (Figure 7-14). These characteristics are inline the findings of the parameter scoring and optimisation.

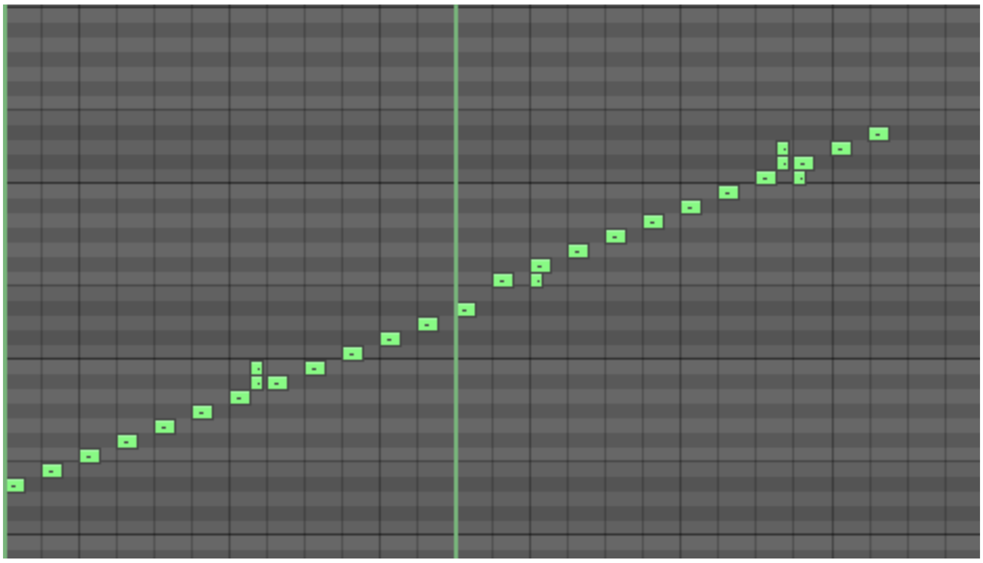


Figure 7-16 Solution 3 (Dressler Parameters) transcription high frequencies at 200BPM

The performance of solution 6 suggests a greater emphasis of the frequency resolution for the lower notes is required in the MRFFT optimisation. However, based upon the transcription of the higher frequencies, evidence suggests the MRFFT can outperform a single band analysis.

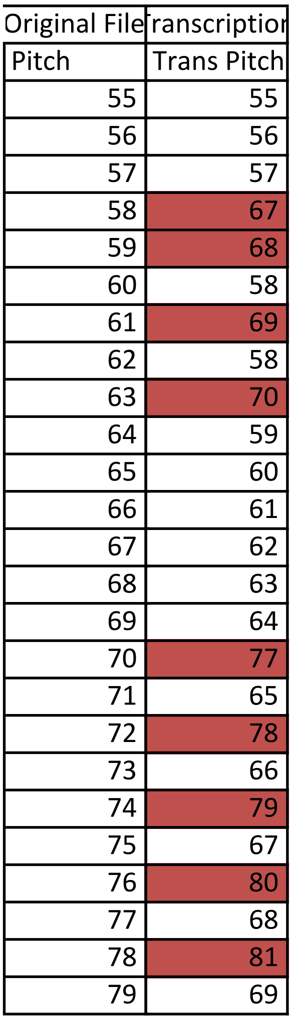
The optimisation process has evaluated all 3 criteria (time resolution, frequency resolution and bin alignment) equally, providing a ‘best fit’ solution. Weighting different bands to different criteria could reduce some of the problems outlined in this section.

Analysis of the MIDI transcriptions reveals characteristics that reflect the results of the scoring and optimisation process, but F-Measure results are severely hindered by poor transcription, and were not as expected. Some of the transcription errors can be explained by MRFFT properties such as shorter window length, however the universally poor F-Measure results for the MRFFT solution led to further investigation of why they performed so poorly.

7.10 Further Investigation of Errors

The poor F-measure results for the MRFFT solutions and the analysis in section 7.9.2 suggest errors are introduced to the system that affects the MRFFT solutions but not solution 6.

Table 21 shows the note information for the original file, and the note information for the solution 2 transcribed file. Pitch errors are highlighted in red.



Original File	ranscriptior
Pitch	Trans Pitch
55	55
56	56
57	57
58	67
59	68
60	58
61	69
62	58
63	70
64	59
65	60
66	61
67	62
68	63
69	64
70	77
71	65
72	78
73	66
74	79
75	67
76	80
77	68
78	81
79	69

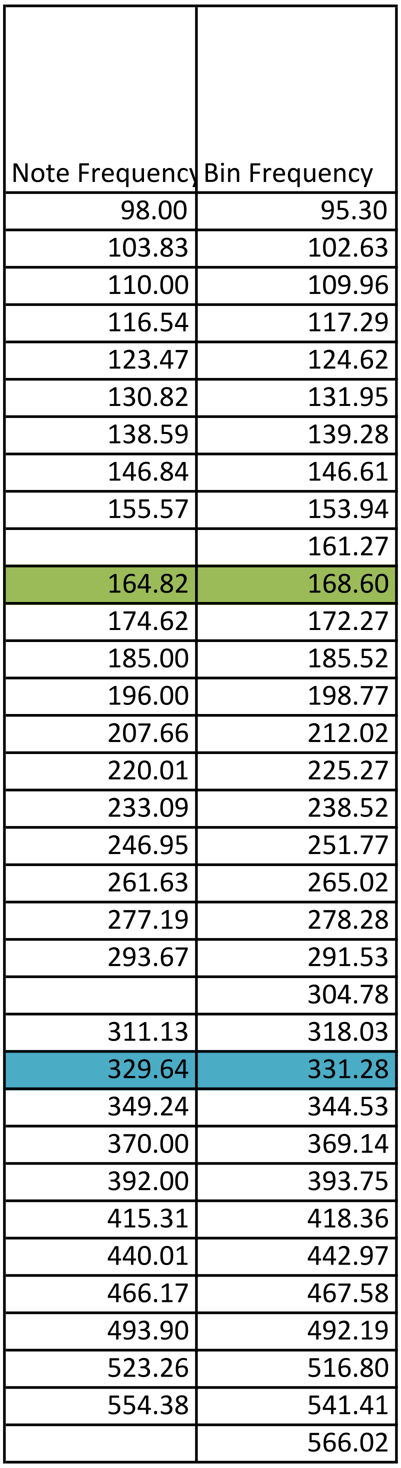
Table 21 3 Band MRFFT transcription errors

The pitch errors shown highlight a problem with spurious frequency maxima being detected. A repercussion of this is seen when a series of notes is correctly identified, but they are *out of position* compared to the original file. This could result in the correct notes being counted as false positives by the F-measure algorithm.

The incorrect frequency detections required further analysis on the frequency domain to determine why they were occurring in the MRFFT solutions, but not in solution 6.

7.10.1 Note Distribution Analysis

To investigate the errors in the MRFFT transcriptions, note distribution was analysed. Table 22 shows how the fundamental frequencies of notes align to the frequency bins of solution 1. The coloured rows denote the sub-band divisions.



Note Frequency	Bin Frequency
98.00	95.30
103.83	102.63
110.00	109.96
116.54	117.29
123.47	124.62
130.82	131.95
138.59	139.28
146.84	146.61
155.57	153.94
	161.27
164.82	168.60
174.62	172.27
185.00	185.52
196.00	198.77
207.66	212.02
220.01	225.27
233.09	238.52
246.95	251.77
261.63	265.02
277.19	278.28
293.67	291.53
	304.78
311.13	318.03
329.64	331.28
349.24	344.53
370.00	369.14
392.00	393.75
415.31	418.36
440.01	442.97
466.17	467.58
493.90	492.19
523.26	516.80
554.38	541.41
	566.02

Table 22 Solution 1 (4 Band MRFFT) note distribution

Table 22 shows the notes are successfully represented by individual bins. Table 23 shows the average distance of the note from the bin frequency as a fraction of the bin size (frequency resolution) for all solutions.

Solution	Average distance from center of band as a fraction. 0=centre, 0.5=half way between bins
1	0.2
2	0.2
3	0.272
4	0.287
5	0.2
6	0.264

Table 23 average distances from center of the bin

Table 23 shows the optimised solutions 1,2 and 5 to be better 'tuned' to note frequencies than solutions 3,4 and 6. As these theoretical optimisations are not supported by the F-Measure results, then the magnitudes generated must be analysed in 'real world' tests rather than theoretical.

7.10.2 Quality of Note Candidates – Polyphonic analysis

To indicate the tuned performance for analyzing music, each solution decomposed a polyphonic mixture of sine waves, with each sine equal to a different fundamental note frequency in the MRFFT range. All sine waves were generated with peak amplitude of 1. All notes in the range were represented so the results give an indication of the quality of note candidate generated by each MRFFT. This is the absolute worse case scenario as all note frequencies are sounding at once. This also serves to give an indication of the MRFFT performance for polyphonic analysis.

The results for each solution are represented in the figures below. The graph for each solution shows the spectral output of the solution, giving an indication of note distribution across the bands and the magnitudes generated. Bins that represent notes are red; bins representing non-note frequencies are blue. This indicates the level of cross channel interference into non-note representing bins.

7.10.3 Solution 1 (4 Band MRFFT) Note Candidates

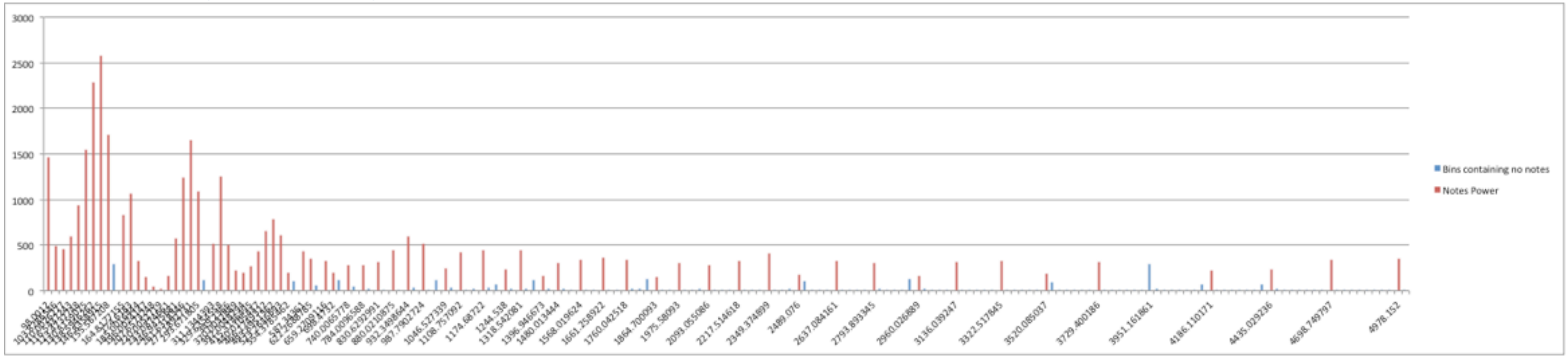


Figure 7-17 Solution 1 (4 Band MRFFT) Note Candidates

7.10.4 Solution 2 (3 Band MRFFT) Note Candidates

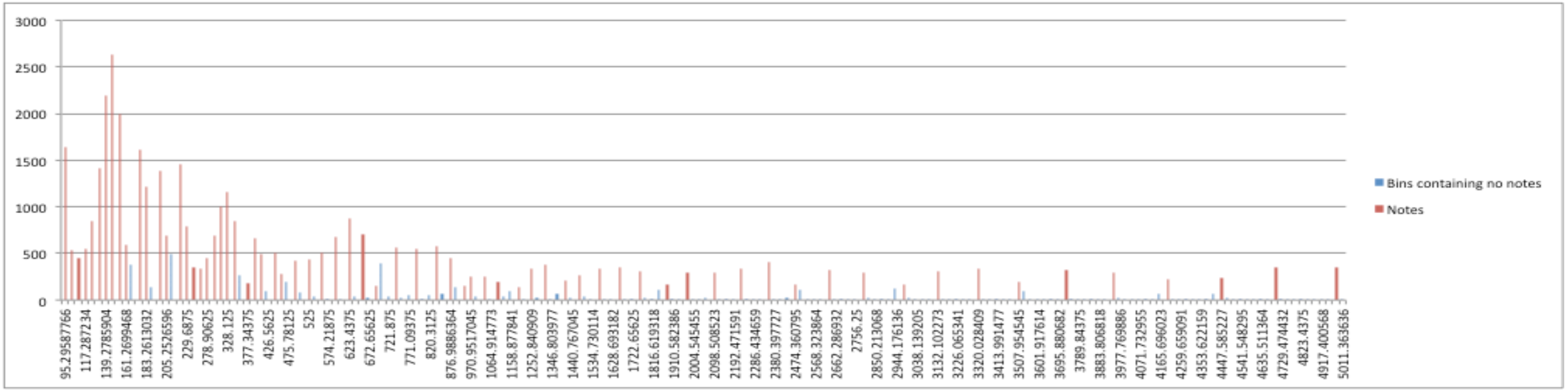


Figure 7-18 Solution 2 (3 Band MRFFT) Note Candidates

7.10.5 Solution 3 (Dressler MRFFT) Note Candidates

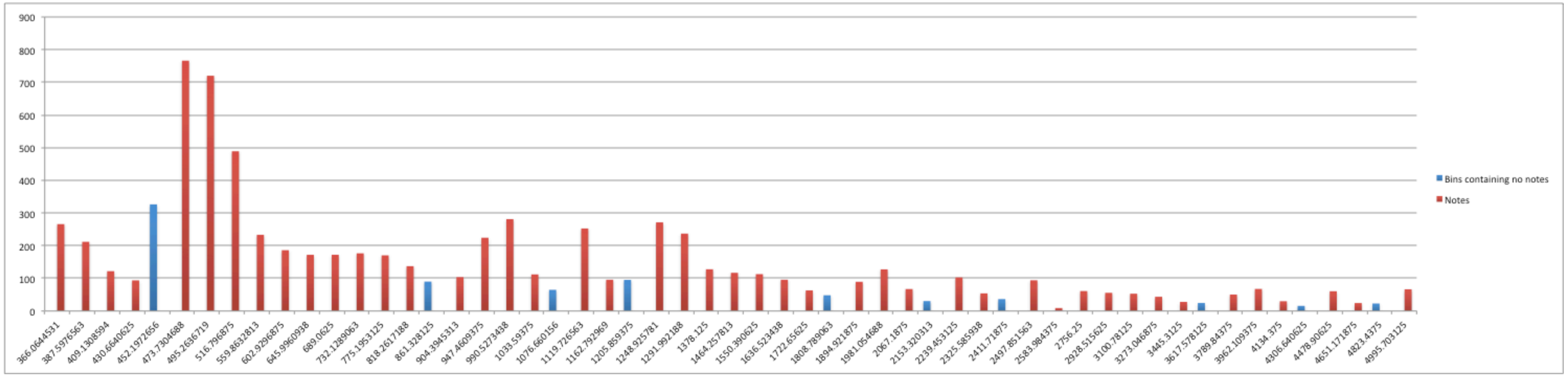


Figure 7-19 Solution (Dressler MRFFT) 3 Note Candidates

7.10.6 Solution 4 (Dressler FFT Length, Optimised Bands) Note Candidates

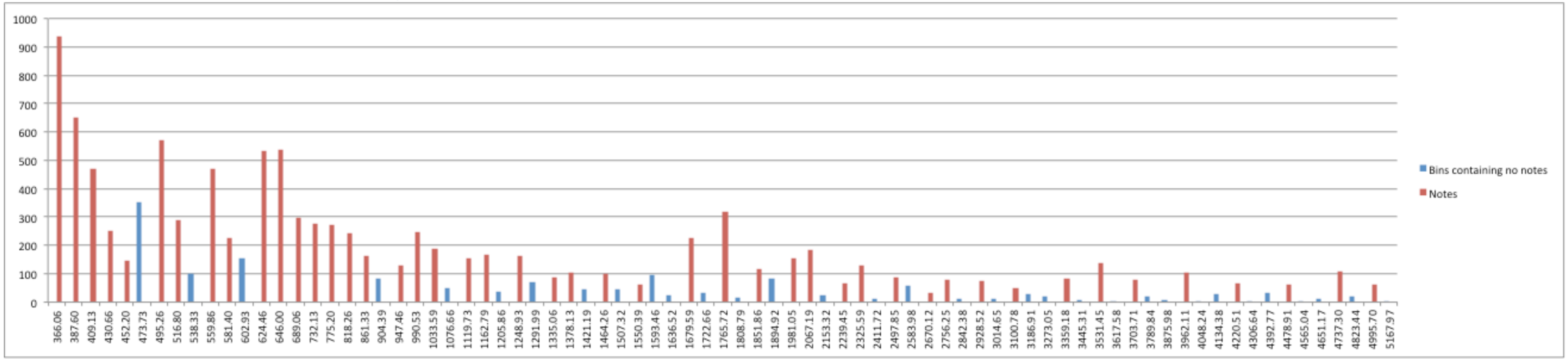


Figure 7-20 Solution 4 (Dressler FFT Length, Optimised Bands) Note Candidates

7.10.7 Solution 5 (256 – 2048 FFT range) Note Candidates

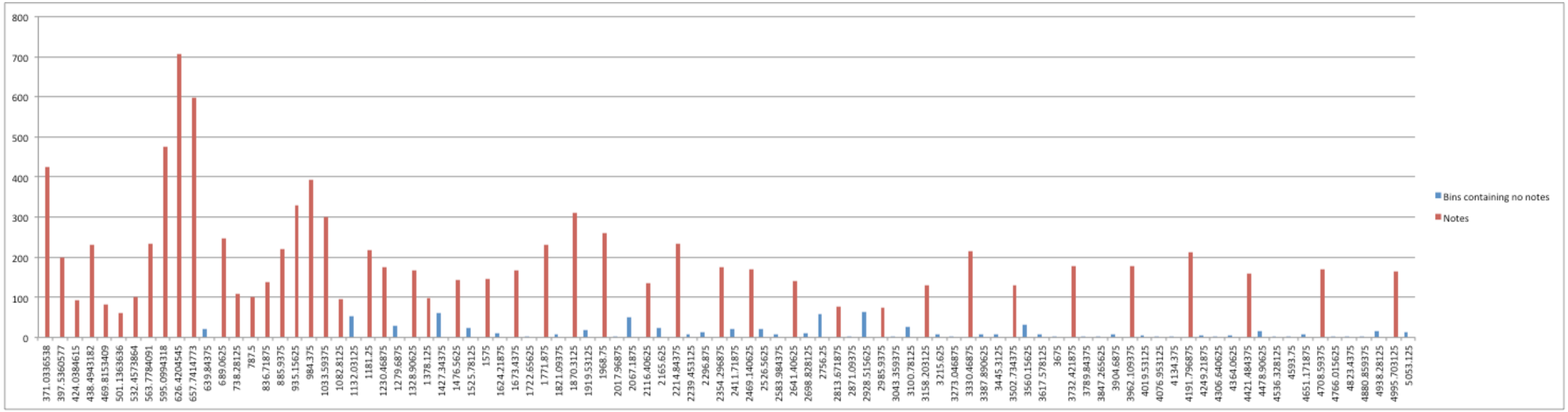


Figure 7-21 Solution 5 (256 – 2048 FFT range) Note Candidates

7.10.8 Solution 6 (8192 FFT) Note Candidates

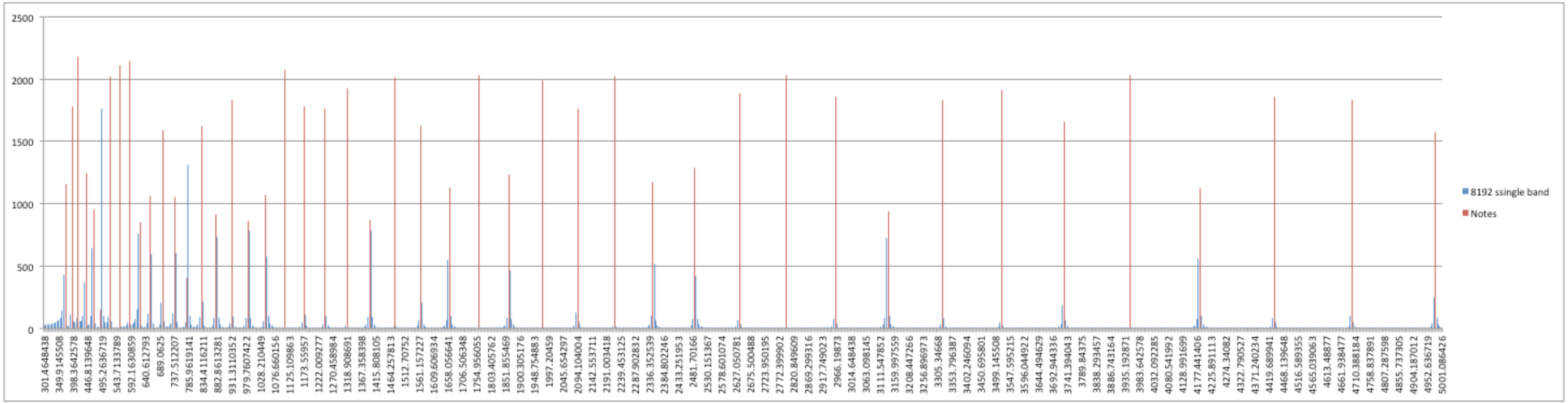


Figure 7-22 Solution 6 (8192 FFT) Note Candidates

7.10.9 Quality of Note Candidate Discussion

All solutions successfully extracted all note frequencies, so perform well under polyphonic conditions. However, closer analysis of the quality of polyphonic note candidates generated by each solution highlight some interesting characteristics of the presented MRFFT solutions.

Solution 6, which performed relatively well in the F-Measure test, generates strong note candidates. The magnitudes are high and consistent across the spectrum. This is due to the long FFT window, and provides an indicator of why the MRFFT solutions scored poorly.

The magnitudes of the note candidates for solutions 1 to 5 are generally lower than solution 6, but also vary significantly across the spectrum. The poor MRFFT F-Measure results may be explained by the inability of the peak picker to resolve a range of magnitudes.

For example, if a peak picker threshold was set to be 25%, the frequency magnitudes from solution 6 would be successfully picked (Figure 7-24). However, due to the variance of magnitudes in the MRFFT solutions, notes are not being picked Figure 7-23. This partially explains the missing notes in transcriptions. In the F-Measure test, the threshold is set dynamically to consider local maxima and set as a percentage of the maxima, so for an area where spectral magnitudes are consistent e.g. the higher frequencies in Figure 7-17 the dynamic peak picker would be successful. This is seen in the transcription of the higher frequencies (Figure 7-13).

Further inadequacies of the peak picker are again highlighted when considering low frequency maxima for the MRFFT solutions. Figure 7-23 shows the low frequency magnitudes for solution 1. A hypothetical threshold of 25% of the maximum magnitude within the 'local terrain', still results in 'missed' transcriptions as the variation is so great. The variations of maxima in the MRFFT solutions contribute significantly to the poor transcription performance in the lower frequencies.

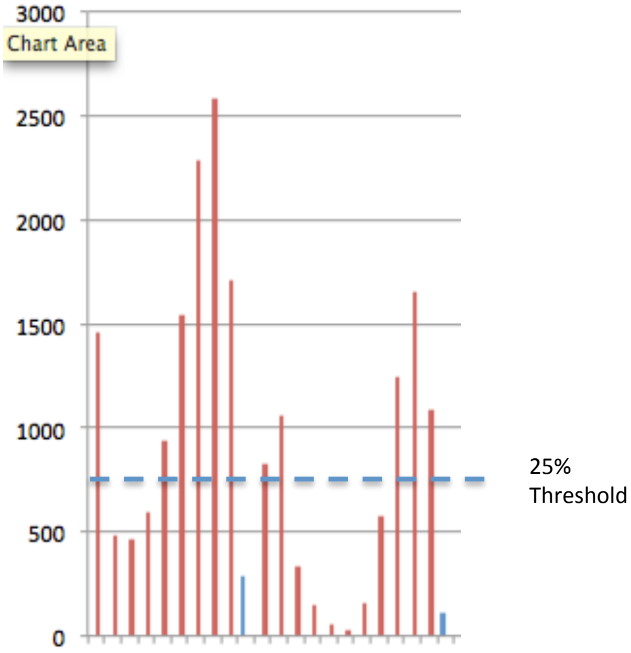


Figure 7-23 Solution 1 (4 Band MRFFT) Low Frequency Magnitudes

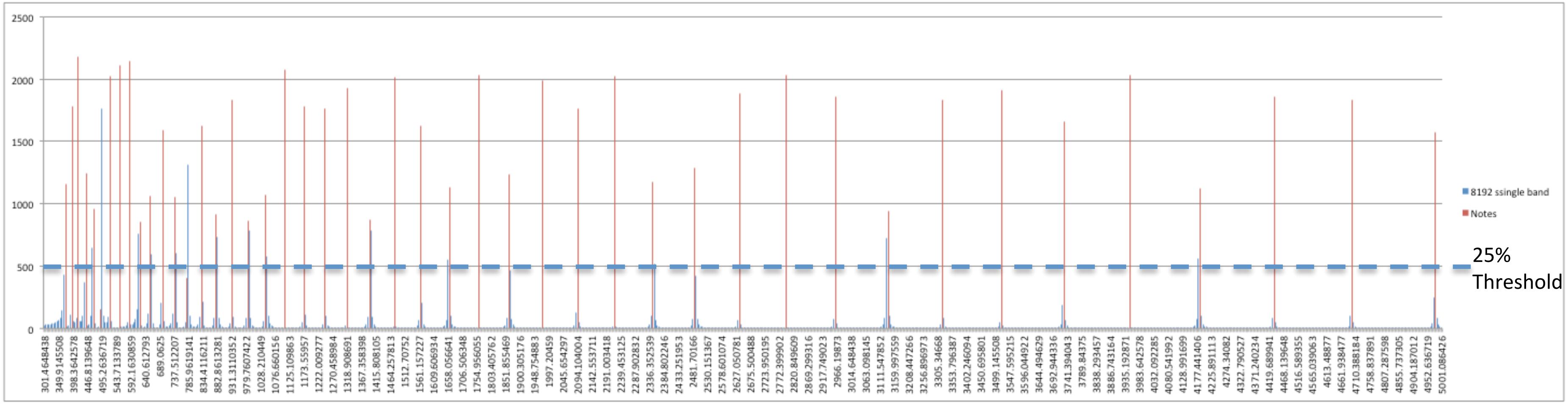


Figure 7-24 Solution 6 (8192 FFT) 25% threshold

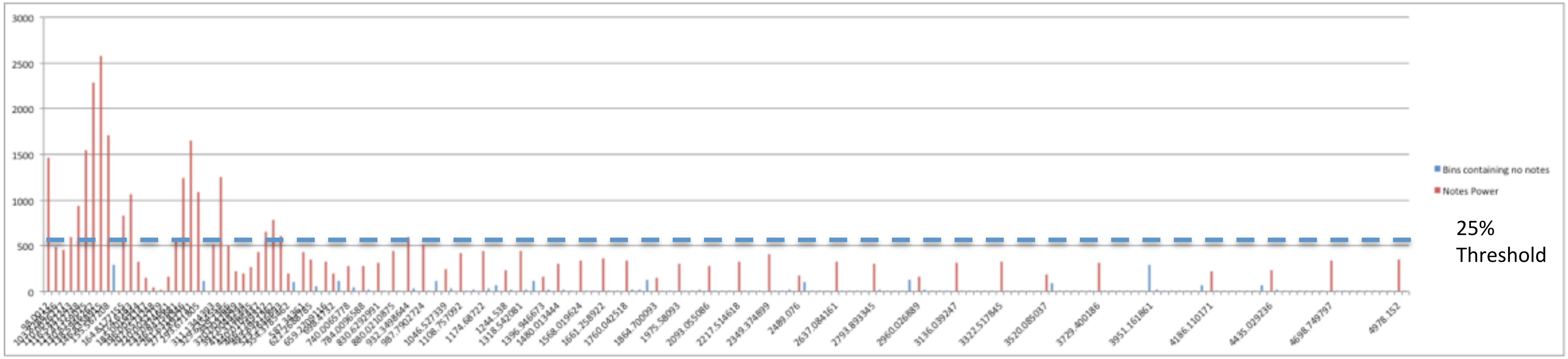


Figure 7-25 Solution 1 (4 Band MRFFT) 25% threshold

The variation of maxima in the MRFFT solution can be partially attributed to the variation of FFT window length, which is constant in solution 6. However, based on Figure 7-23, variations are occurring within a single band of FFT.

Although the above figures do not show the phase of the signals it is important to mention at this stage as there will be inconsistencies in the phase information, as well as the energy magnitude. The phase of the magnitudes are an important feature for high level processing. For example, the tracking phase vocoder (TPV) will assign each peak to a 'frequency track' by matching the peaks of a previous frame with those of the current frame. This peak tracking uses phase information to identify a continued peak from one frame to another (Roads, 1996, p. 571). As the presented MRFFT doesn't feature any phase correction, the phases of fundamentals and harmonics, which over lap sub-band divisions will have distorted phases due to the change in data length used to decompose the original signal. This will result in poor quality note candidates. Phase issues do not affect the sinusoidal extraction test results, as the source audio is non-harmonic pure tones. Also, the peak picker does not use phase information, so although phases are distorted it will have no bearing on these results, but it is an important consideration for future implementations for the reasons outlined.

Some of the variation of the energy magnitudes can be attributed to cross channel interference summing to create larger magnitudes. Table 24 compares to spacing of notes between solution 1 and 6.

Note Frequency	Bin Frequency
98.00	96.90
103.83	102.28
110.00	107.67
	113.05
116.54	118.43
123.47	123.82
130.81	129.20
	134.58
138.59	139.97
155.57	145.35
	150.73
155.57	156.12
	161.50
164.82	166.88
174.62	172.27
	177.65
185.00	183.03
	188.42
196.00	193.80
	199.18
	204.57
207.65	209.95
	215.33
220.00	220.72
	226.10
233.08	231.48
	236.87
	242.25
246.94	247.63
	253.02
	258.40
261.63	263.78
	269.17
	274.55

Note Frequency	Bin Frequency
98.00	95.30
103.83	102.63
110.00	109.96
116.54	117.29
123.47	124.62
130.82	131.95
138.59	139.28
146.84	146.61
155.57	153.94
	161.27
164.82	168.60
174.62	172.27
185.00	185.52
196.00	198.77
207.66	212.02
220.01	225.27
233.09	238.52
246.95	251.77
261.63	265.02

Table 24 Solution 6 (8192 FFT) Bin Spacing and Solution 1 (4 BAND MRFFT) Bin Spacing

The bin spacing of solution 6 results in empty bins more regularly between note frequencies than solution 1. Solution 1 optimisation has resulted in all but 1 adjacent bin representing a note. This means any cross channel interference will contribute to a bin representing another note. In solution 6 there is a 'buffer zone' of non note representing bins for cross channel interference to feature before contributing to bins representing note frequencies. This may account for some of the variation in spectral magnitudes for the MRFFT solutions.

Solution	average power magnitude in note bins	average power in non note bins	average power in non note bins as % of power in note bins
1	538.79	23.4338906	4.35
2	594.48	32.7189717	5.50
3	157.7	29.0526256	18.42
4	216.75	45.368934	20.93
5	208.73	35.0150222	16.78
6	1527	27.4607458	1.80

Table 25 Summary of note distribution magnitudes

Based upon the note distribution testing, Table 25 shows the average power in non-note representing bins as a percentage of the average power in note representing bins. This shows the cross channel interference is greater in the MRFFT solutions than the 8192, but the optimised solutions 1 and 2 are improvements over solutions 3,4 and 5.

Another contributing factor to cross channel interference is the process of constructing the MRFFT. Taking the lower band of solution 1 as an example, a 6016 FFT is performed on the entire frequency spectrum. The spectral information is then filtered to include only the frequencies required by that band.

Figure 1-1 Superposition of wavesFigure 7-26 shows a scenario where a note frequency (orange magnitude) not in the frequency band considered, generates

cross channel interference (red magnitudes) that contributes to the magnitudes in the sub-band of interest.

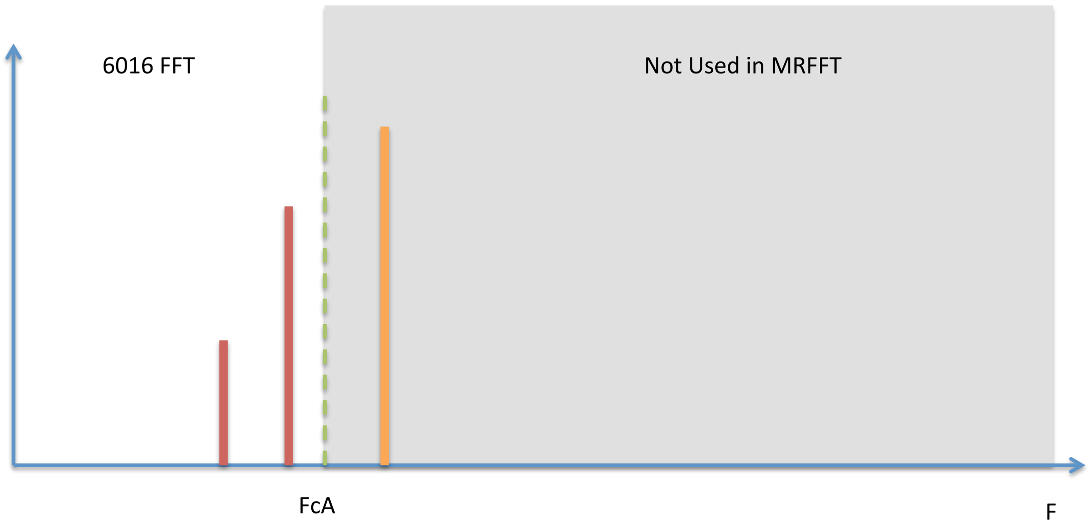


Figure 7-26 MRFFT Band interference

This scenario could explain why there are large magnitudes near the lowest sub-band boundary of solution 1.

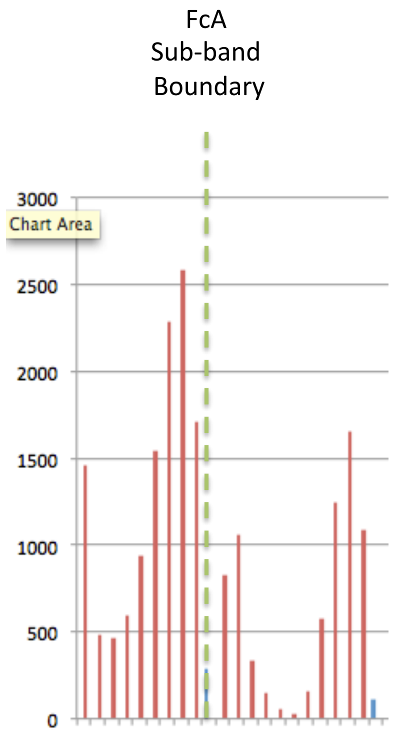


Figure 7-27 Solution 1 (4 Band MRFFT) FcA Sub-band boundary

Implementing a filter in the time domain with steep pass bands to divide the spectrum into sub-bands and minimise spectral leakage could help reduce the variation of magnitudes within bands.

7.10.10 Investigation conclusions

The results of the F-Measure are largely disappointing, and can be attributed to the inadequacies of the implemented peak picker to handle fluctuations in magnitude of local maxima. Characteristics of the MRFFT, like adjacent note representing bins, and interference generated by sub-band division methods contribute to this problem.

Future development of more robust peak picker to handle magnitude fluctuations, implementing spectral processing, and time domain filtering could all improve the performance of the MRFFT to reflect more convincingly it's theoretical advantages in real world testing.

7.11 Optimised MRFFT and Polyphonic Transcription Testing

Although the ability to handle and generate polyphonic note candidates has been discussed in section 7.10.2, the MRFFT solutions presented have not been tested for their polyphonic music transcription abilities. Polyphonic testing is essential further work but there are several reasons why it has not been performed as part of this thesis.

> The primary purpose of the transcription testing was to evaluate the characteristics of each MRFFT in generating note candidates as predicted by the optimisation results and not to evaluate its success as a complete transcription system.
>
> Following the monophonic analysis and the subsequent flaws it highlighted it was clear that the peak picker and construction of the MRFFT would not be robust enough to cope with polyphonic transcription.
>
> As previously discussed, polyphonic music transcription is a complex task so to test the MRFFT it would have to be used as a front end to a far more complex algorithm (as Dressler used (Dressler, 2006)) than that implemented in section 7.9, which would be beyond the scope of this project.

8 Conclusion

An in-depth review of methods and techniques as well as the challenges of automatic music transcriptions has been presented. Descriptions and discussions of both high level and low level processes have been included to place in context the investigation of the effect of the FFT parameters upon it's behavior and characteristics for automatic music transcription.

Reviewing current literature indicated that there was no set of standard FFT parameters for use in automatic music transcription algorithms. This provided the motivation for the investigation to determine a set of optimised standard FFT parameters for use in automatic music transcription algorithms.

The purpose of this investigation was to optimise the parameters of a multiresolution FFT (MRFFT) to increase the quality of the note candidates in the MRFFT spectral output. It is envisaged that providing 'stronger' note candidates will lead to more successful higher level processing.

The effect of varying the FFT parameters on the FFT spectral output was analysed to determine which parameters would be scored and optimised for a MRFFT suitable for music transcription algorithms. This investigation determined that frequency resolution, time resolution and note-to-bin alignment are three characteristics that are of primary importance for generating quality note candidates.

The MRFFT parameters that were optimised were the FFT length and the cutoff frequencies between FFT subbands. Both 3 band and 4 band MRFFTs were optimised. Zero padding, hop size and window shapes were investigated and discussed but not included as parameters to be optimised. Investigations concluded that the performance of each of these parameters depends in part on the characteristics and content of the input audio. The methodology of this work does not account for variations in the input signal that would affect the performance of zero padding, window shape and hop overlap. Therefore, the system cannot optimise these particular parameters.

The novel element of this work was the scoring of the FFT parameters and exhaustively checking all combinations of sub-band divisions and FFT data length to select an optimised set. The FFT parameter sets were scored based on the following criteria:

- **Time Resolution** – Determined by the data length used in the FFT.
- **Frequency Resolution** –Determined by the data length used in the FFT.
- **Bin Tuning** – This is based on the positioning of fundamental note frequencies of a 440Hz tuned scale within the FFT bins. The closer to the center of the bin a note value is, the better the score for that bin. If multiple notes are in a single bin, the bin is penalized. The score for each bin is combined to give an overall score.

The search method used was an exhaustive search with the objective of minimising a combined 'error score' for frequency resolution, time resolution and note-to-bin alignment. The error score is calculated based upon the difference between the best possible frequency resolution, time resolution and note-to-bin alignment and those generated by the combination of FFT Length and sub band divisions in the MRFFT.

The generated scores determined an optimised set of parameters for a 4 band MRFFT and a 3 band MRFFT. Three further 4 Band MRFFTs were used as a comparison as well as a single band 8192 FFT. One of the 4 band MRFFTs was by Dressler (Dressler, 2006) and two variations on this were created to facilitate comparison.

The recommended parameters for MRFFT implementation based upon the presented search and optimisation are shown in Table 26. The recommended parameters are for use when transcribing instruments tuned to A=440Hz and using standard equal-tempered tuning. These parameters assume a 44.1KHz sample rate.

	3 Band MRFFT		**4 Band MRFFT**	
	Range	**FFT Length**	**Range**	**FFT Length**
Band 1	98Hz-226Hz	6016	98Hz-169Hz	6016
Band 2	227Hz-855Hz	2688	170Hz-339Hz	3328
Band 3	856Hz-5KHz	1408	340Hz-1209Hz	1792
Band 4			1210Hz-5Khz	1408

Table 26 Recommended MRFFT Parameters

The optimisation process shows that the 4 band MRFFT parameters as used by Dressler can be optimised to improve the quality of the note candidates produced.

The implication of this theoretical optimisation is that the note candidates produced by the optimised MRFFT will be of better quality than those produced by Dressler. The advantages of the optimised 4 band MRFFT compared to Dresslers were a better note-to-bin alignment score and significantly improved frequency resolution. Dressler's MRFFT offered a better time resolution, but not significantly enough to better the MRFFT score of the optimised solution.

The set of optimised solutions generated were tested on a sinusoidal extraction task and evaluated based upon the F-measure of their transcriptions. These experiments used a simple threshold based peak picking algorithm to select from note candidates generated by MRFFT.

The sinusoidal extraction test demonstrated disappointing F-measure results for all of the MRFFT solutions (including Dressler's MRFFT and variations) compared to the single band 8192 FFT. Close analysis of the transcribed files showed positive aspects of the optimised MRFFT analyses as performance improved in the higher frequencies as notes were more accurately transcribed in the time domain.

Further investigation of the sinusoid extraction results revealed inadequacies in the simple peak picker and also indicated issues with the construction of the MRFFT. In particular, cross channel interference affected adjacent bands of the MRFFT and caused false positive note candidates, which may account in part for the poor F-measure results.

The inadequacies of the peak picker resulted in limited polyphonic testing. Results showed that all solutions could successfully extract note frequencies from a polyphonic mixture of sine waves, but all MRFFT solutions presented a variable quality of note candidates. This was attributed to cross channel interference, increased by the implementation of the MRFFT, and in part the optimisation process, which favours a shorter FFT length and therefore generates smaller magnitudes in the FFT bins.

The failings of the simple peak picking and transcription algorithm resulted in no testing of harmonically rich audio. Consequently no conclusions about the performance of the MRFFT with harmonically rich content can be made.

The current work is only optimised for standard tuning to A (440Hz). It is expected that there will be a graceful degradation of performance as the tuning moves away from 440Hz.

If an instrument were tuned to an alternative tuning, the MRFFT would no longer be optimised. Although there would still be advantages in terms of time and frequency resolutions by using the optimised MRFFT over a single resolution FFT, the non-alignment of instrument note frequencies with the bin frequencies would result in a spectral output of the MRFFT which would feature a large amount of cross channel interference. This cross channel interference would potentially result in reduced magnitudes, less accurate representations of the frequency being decomposed and a poorer quality of note candidates being presented.

9 Further Work

Further testing is required of the presented parameters. Only simple sinusoidal extraction tasks have been performed to test effectiveness of the optimisation. Further work would include the testing of these parameters as a front end for an established pitch recognition algorithm transcribing 'real' instruments to ascertain if any real gains have been made.

The implementations of such a simple peak-picking algorithm has coloured the results. Therefore, development and implementation of a robust peak picker would generate more accurate representations of the qualities of the MRFFT solutions.

The implementation of the optimised MRFFT as part of a complete automatic music transcription algorithm would allow for thorough testing on real musical examples, which would provide more complete comparisons with other current low level processing techniques. It will also give true indicators of performance and possible problems when dealing with harmonic content.

The development of time domain filtering to minimize cross channel interference at sub-band divisions could also improve the performance.

Extensive testing with a variety of musical style, instruments, complexities and tempos would generate a broader spectrum of results and give a more accurate indication of the effects of optimisation.

The study presented is very focused on a small set of parameters. Further work would have to consider the use of popularly used parameters like window shape, hop size and zero padding in the scoring of FFT parameters.

The study focuses only on the tuning of bins to fundamental frequencies of the western scale. Future considerations should be given the placement of harmonics within the bin alignment as harmonic patterns are important in several automatic music transcription techniques.

Also, no consideration has been given to the phase shift effect of the multi windowed FFT. The method used to generate multiple windowed FFTs results in phase inconsistencies across the analysis windows. The phase of the output of the FFT is of importance for many algorithms, including the popular phase vocoder peak picker. Therefore, further attention should be given to the phase output of the optimised MRFFT.

The method of scoring the MRFFT parameters is mathematically sound, but could be further enhanced by introducing weightings for certain criteria. An example where this could be useful is when transcribing a piece of music with high BPM it may be beneficial to have a high time resolution to detect fast transients, which could be gained by compromising note-to-bin alignment.

The presented optimisation is based upon a standard tuning of 440Hz and equal tempered instruments and music. To account for alternative tunings a selection of optimised MRFFT could be developed which are generated using bin alignment scores based upon alternative tunings.

The performance of the single band FFT in the extraction test and the subsequent investigation exceeded expectations, and demonstrated a suitable single band FFT can be effective as a low level processing tool. However, despite the problems encountered with the peak picker and MRFFT construction, the advantages of a multi resolution solution in terms of time and frequency resolution were still evident in the analysis of the extraction task. These advantages would be significantly accentuated with improved implementation, which it is anticipated would improve the note candidates generated compared to a single band solution.

10 References

Akansu, & Haddad. (1992). Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets (2nd Edition ed.). Newark: Academic Press.

Anderson, D. (1996). Speech Analysis and Coding Using a Multiresolution Sinusoidal Transform. International Conference of Acoustics, Speech and Signal Processing (pp. 1037-1040). Atlanta: IEEE Signal Processing Society.

Arons, B. (1992). A Review of the Cocktail Party Effect. Journal of the American Voice I/O Society, 12, 35-50.

Azizi, Faez, Delui, & Rahati. (2009). Automatic Music Transcription Based on Wavelet Transform. International Conference on Intelligent Computing 2009. LNCS 5754, pp. 158 - 165. Springer.

Barabell, & Crochiere. (1979). Sub-Band Coder Design Incorporating Quadrature Filters and Pitch Prediction. IEEE International Conference on ICASSP79. (pp. 530-533). IEEE Signal Processing.

Bello, Daudet, Abdallah, Duxbury, Davies, & Sandler. (2005). A Tutorial on Onset Detection in Music Signals. IEEE Transactions on Speech and Audio Processing, 13 (5), 1035-1047.

Benaroya, Blouet, Fevotte, & Cohen. (2006). Single Sensor Source Separation Using Multiple-Window STFT Representation. International Workshop on Acoustic Echo and Noise Control. Paris: Hindawi.

Benetos, & Dixon. (2011). Polyphonic Music Transcription using Note Onset and Offset Detection. International Conference for Audio, Speech and Signal Processing (pp. 37-40). IEEE Signal Processing Society.

Betser, Collen, Bertrand, & Gael. (2006). Review and Discussion on Classical STFT Based Frequency Estimators. 120th AES Convention (pp. 1-11). Paris: AES.

Bregman. (1994). Auditory Scene Analysis: The Perceptual Organization of Sound. A Bradford Book.

Brown, & Puckette. (1992). An Efficient Algorithm for the Calculation of a Constant Q Transform. Journal of Acoustical Society of America, 92 (5), 2698-2701.

Campbell, & Greated. (1994). The Musicians Guide to Acoustics (2nd Edition ed.). New York: OUP Oxford.

Cancela, Rocamora, & Lopez. (2009). An Efficient Multi-Resolution Spectral Transform for Music Analysis. International Society for Music Information Retrieval Conference (pp. 309-314). ISMIR.

Carabias-Orti, Vera-Candeas, Ruiz-Reyes,˜adas-Quesada, C., & Mata-Campos. (2009). Overlapped Event-Note Separation Based On Partials Amplitude And Phase Estimation For Polyphonic Music Transcription. 17th European Signal Processing Conference (pp. 943-947). Glasgow: EUSIPCO.

Cassidy, & Smith. (2008). Auditory Filter Bank Lab. Stanford University, Department of Electrical Engineering. Stanford: Stanford University.

Cherry, E. (1953). Some Experiments on the Recognition of Speech, with One and Two Ears. Journal of the Acoustical Society of America, 25 (5), 975-979.

Cheveigne, & Kawahara. (2001). Comparative Evaluation of F0 Estimation Algorithms. 7th European Conference for Speech Communication and Technology. Aalborg: ISCA.

Cheveigne, & Kawahara. (2002). YIN, a Fundamental Frequency Estimator for Speech and Music. Acoustical Society of America, 111 (4), 1917-1930.

Cheveigne. (1991). Speech F0 Extractions Based on Licklider's Pitch Perception Model. International Congress of Phonetic Sciences (pp. 218-221). Aix-en-Provence: ICoPS.

Collins, N. (2005). A Change Discrimination Onset Detector with Peak Scoring Peak Picker and Time Domain Correction. Music Information Retrieval Exchange 2005. London: ISMIR.

Cooley, & Tukey. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation, 19 (90), 297-301.

Davy, M. (2006a). An Introduction to Signal Processing. In A. Klapuri, & M. Davy, Signal Processing Methods for Music Transcription (pp. 25-28). New York: Springer.

Davy, M. (2006b). Multiple Fundamental Frequency Estimation Based on Generative Models. In D. A. Klapuri, M. Davy, & M. D. Anssi Klapuri (Ed.), Signal Processing Methods for Music Transcription (pp. 203-227). New York: Springer.

Dessein, Cont, & Lemaitre. (2010). Real-Time Polyphonic Music Transcription With Non-Negative Matrix Factorisation and Beta Divergence. International Society for Music Information Retrieval Conference (pp. 489-494). ISMIR.

Diniz, F., Kothe, I., Netto, S., & Biscainho, L. (2007). High-Selectivity Filter Banks for Spectral Analysis of Music Signals. EURISP Journal on Advances in Signal Processing, 2007 (1), 164-176.

Djurovic, & Stankovic. (2003). Adaptive Windowed Fourier Transform. Signal Processing, 83 (1), 91-100.

Dressler, K. (2006). Sinusoidal Extraction Using An Efficient Implementation Of A Multi-Resolution FFT. 9th Int. Conference on Digital Audio Effects (pp. 247 - 252). Montreal: DAFx.

Dressler, K. (2009). Audio Melody Extraction for MIREX 2009. Fraunhofer IDMT, Germany. Ilmenau: Fraunhofer IDMT.

Duhamel, P., & Vetterli, M. (1990). Fast Fourier Transforms: A Tutorial Review and a State of the Art. Signal Processing, 19, 259-299.

Durrieu, Gael, & Bertrand. (2008). Main Melody Extraction from Polyphonic Music Excerpts Using a Source/Filter Model of the Main Source. Music Information Retrieval Exchange. Philadelphia: MIREX.

Duxbury, Bello, Davies, & Sandler. (2003). Complex Domain Onset Detection for Musical Signals. 6th International Conference on Digital Audio Effects (pp. 1-4). London: DAFx.

Duxbury, Bello, Davies, & Sandler. (2004). A Comparison Between Fixed and Multiresolution Analysis for Onset Detection in Musical Signals. International Conference on Digital Audio Effects (pp. 207-212). Naples: DAFx.

Ellis, D. (2009, July 7). Gammatone-like Spectrograms. Retrieved January 21, 2012, from Columbia University The Fu Foundation School of Engineering and Applied Science:
ttp://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram/

Filipe, Diniz, Luiz, Biscainho, & Netto. (2006). A Bounded-Q Fast Filter Bank for Audio Signal Analysis. International Telecommunications Symposium (pp. 971-975). Fortaleza: IEEE.

Flanagan, & Golden. (1966). Phase Vocoder. Bell System Technical Journal, 45 (9), 1493-1509.

Fletcher, H. (1938). The Mechanism of Hearing as Revealed Through Experiment on the Masking Effect of Thermal Noise. Proceedings of National Academy of Sciences of USA, 24 (7), 265-274.

Frigo, M., & Johnson, S. (2005). The Design and Implementation of FFTW3. Proceedings of the IEEE, 93 (2), 216-231.

Garas, & Sommen. (1998). Time/Pitch Scaling Using The Constant-Q Phase Vocoder. Proceedings of STW's 1998 Workshops CSSP98 and SAFE98, (pp. 173-176).
Eindhhoven: Eindhoven University of Technology.

Gerhard, D. (2003). Pitch Extraction and Fundamental Frequency: History and Current Techniques. University of Regina, Department of Computer Science. Regina: University of Regina.

Ghahramani, Z. (2001). An Introduction to Hidden Markov Models and Bayesian Networks. Journal of Pattern Recognition and Artificial Intelligence, 15 (1), 9-42.

Godwin, M. M. (1997). Adaptive Signal Models: Theory, Algorithms, and Audio Applications. University of California, Electrical Engineering and Computer Science. California: University of California.

Gold, B., & Morgan, N. (2000). Speech and Audio Signal Processing: Processing and Perception of Speech and Music (1st Edition ed.). New York: John Wiley and Sons Inc.

Goldenshluger, & Nemirovski. (1997). On Spatial Adaptive Estimation of Nonparametric Regression. Mathematic Methods of Statistics, 6 (2), 135-170.

Goto, M. (2000). A Robust Predominant F0 Estimation Method for Real Time Detection of Melody and Bass Lines in CD Recordings. IEEE ICASSP. 2, pp. 757-760. Istanbul: IEEE.

Goto, M. (2002). A Predominant-F0 Estimation Method for Real-world Musical Audio Signals: MAP Estimation for Incorporating Prior Knowledge about F0s and Tone Models. National Institute of Advanced Industrial Science and Technology, Japan Science and Technology Corporation. Ibaraki: PRESTO.

Goto, M. (2006). Music Scene Description. In A. Klapuri, Signal Processing Methods for Music Transcription (pp. 327-359). New York: Springer.

Gouyon, F., Pachet, F., & Delerue, O. (2000). On the use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. COST G-6 Conference on Digital Audio Effects 2000 (pp. 147-152). Verona: DAFX.

Grey, J. M., & Gordon, J. W. (1978). Perceptual Effect of the Spectral Modifications of Music Timbres. Acoustical Society of America, 61 (5), 1270-1277.

Gutman, R. W. (1999). Mozart: A Cultural Biography. Orlando, Florida: Haughton, Miffiln and Harcourt.

Harris, C. M., & Weiss, M. R. (1963). Pitch Extraction by Computer Processing of High Resolution Fourier Analysis Data. Journal of the Acoustical Society of America, 35 (3), 339-343.

Harris, F. J. (1978). On The Use Of Windows For Harmonic Analysis With The Discrete Fourier Transform. Proceedings of the IEEE, 66 (1), 51-83.

Hsieh, I.-H., & Saberi, K. (2007). Temporal Integration in Absolute Identification of Musical Pitch. Hearing Research, 233, 108-116.

Hsu, C.-L., & Jang, J.-S. R. (2010). Singing Pitch Extraction MIREX 2010. International Society for Music Information Retrieval. Utrecht: ISMIR.

Humphrey, E. (2010). Automatic Charcterization of Digital Music for Rhythmic Auditory Stimulation. ISMIR (pp. 69-74). Utrecht: ISMIR.

Johannesma. (1972). The Pre-Response Stimulus Ensemble of Neurons in the Cochlear Nucleus. Proceedings of the Symposium of Hearing Theory (pp. 58-69). Eindhoven: IPO.

Kashino, K. (2006). Auditory Scene Analysis in Music Signals. In A. Klapuri, Signal Processing Methods for Music Transcription (pp. 229-325). New York: Springer.

Katkovnik, V., Egiazarian, K., & Shmulevich, I. (2001). Adaptive Varying Window Size Selection Based On Intersection Of Confidence Intervals Rule. Tampere University of Technology, Tampere, Finland, Signal Processing Laboratory.

Keiler, & Marchand. (2002). Survey on Extraction of Sinusoids in Stationary Sounds. International Conference on Digital Audio Effects (pp. 51-58). Hamburg: DAFx.

Keren, R., Zeevi, Y. Y., & Chazan, D. (1998). Automatic Transcription of Polyphonic Music Using The Multiresolution Fourier Transform. Proceedings of Ninth Mediterranean Electrotechnical Conference. 1, pp. 654-657. Tel-Aviv: MELECON98.

Kida, Sakai, Masuko, & Kawamura. (2009). Robust F0 Estimation Based on Log-Time Scale Autocorrelation and its Application to Mandarin Tone Recognition. Conference of International Speech Communication Association (pp. 2971-2974). Brighton: ISCA.

Klapuri, A. (2005). A Perceptually Motivated Multiple-F0 Estimation Method for Polyphonic Music Signals. IEEE Workshop on Applications of Signal Processing Audio to Acoustics. New Paltz: IEEE.

Klapuri, A. (2006a). Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes. International Conference on Music Information Retrieval (pp. 216-221). Victoria: ISMIR.

Klapuri, A. (2006b). Signal Processing Methods for Music Transcription. Tampere, Finland: Springer.

Klapuri, A. (2008). Multipitch Analysis of Polyphonic Music and Speech Signals Using an Auditory Model. IEEE Transactions On Audio, Speech, And Language Processing, 12 (2), 255 - 266.

Kumar, Jakhanwal, Bhowmick, & Chandra. (2011). Gender Classification Using Pitch and Formants. International Conference on Communication, Computing and Security (pp. 319-324). New York: ACM.

Lagrange, M. (2004). Modelisation Sinusoidale des Sons Polyphoniques. University of Bordeaux, Informatique. Bordeaux: University of Bordeaux.

Lahat, Niederjohn, & Krubsack. (1987). A Spectral Autocorrelation Method For Measurement Of Fundamental Frequency Of Noise-Corrupted Speech. IEEE Transactions on Acoustics, Speech and Signal Processing, 35 (6), 741-750.

Latham, A. (Ed.). (2002). The Oxford Companion to Music. Oxford: Oxford University Press.

Laurenti, N., Poli, G. D., & Montagner, D. (2007). A Nonlinear Method for Stochastic Spectrum Estimation in the Modelling of Musical Sounds. IEEE Transactions On Audio, Speech, And Language Processing, 15 (2), 531-541.

Lee, & Seung. (1999). Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature, 401, 788-791.

Lim, Y. C. (1986). Frequency Response Masking Approach for the Synthesis of Sharp Linear Phase Digital Filters. IEEE Transactions on Circuits and Systems, 33 (4), 357-364.

Lim, Y. C., & Farhang-Boroujeny, B. (1992). Fast Filter Bank (FFB). IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, 39 (5), 316-318.

Lyon, Katsiamis, & Drakakis. (2010). History and Future of Auditory Filter Models. IEEE Symposium on Circuits and Systems (pp. 3809-3812). Paris: IEEE.

Lyon, R. F., Katsiamis, A. G., & Drakakis, E. M. (2010). History and Future of Auditory Filter Models. IEEE 2010, (p. 3809). 3812.

Mallat, S. (2009). A Wavelet Tour of Signal Processing: The Sparse Way (3rd Edition ed.). Academic Press.

Mesaros, A., Lupu, E., & Rusu, C. (2003). Singing Voice Features by Time-Frequency Representations. Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis (pp. 471 - 475). Rome: IEEE.

MIREX. (2010a). In Veltkamp, Downie, & Remco (Ed.), Proceedings Of The 11th International Society For Music Information Retrieval Conference. Utrecht: ISMIR.

MIREX. (2010b). MIREX Home. (K. Choi, Editor) Retrieved October 14, 2011, from MIREX Wiki: http://www.music-ir.org/mirex/wiki/MIREX_HOME

MIREX. (2010c). Multiple Fundamental Frequency Estimation & Tracking. Retrieved October 17, 2011, from Mirex Wiki: http://www.music-ir.org/mirex/wiki/Multiple_Fundamental_Frequency_Estimation_&_Tracking

Moelants, D. (2002). Preferred Tempo Reconsidered. International Conference on Music Perception and Cognition (pp. 580-583). Sydney: IPEM.

Moler, C. (2005). Newsletter - Matlab New and Notes. Retrieved August 5, 2011, from Mathworks: http://www.mathworks.com/company/newsletters/news_notes/clevescorner/winter01_cleve.html

Moorer, J. A. (1975). On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer. Stanford University, Department of Music. Stanford University.

National Instruments. (2006, Sep 06). Zero Padding Does Not Buy Spectral Resolution. Retrieved November 01, 2010, from National Instruments: http://www.ni.com/white-paper/4880/en

Nunes, Esquef, & Biscainho. (2007). Evaluation of Threshold-Based Algorithms for Detection of Spectral Peaks in Audio. AES Regional 5th Brazil Conference. Sao Paulo: AES.

Olson, H. (1967). Music, Physics and Engineering (2nd Edition ed.). New York: Dover Publications.

Papoulis, A. (1977). Signal Processing. New York: McGraw-Hill.

Patterson, Nimmo-Smith, Holdsworth, & Rice. (1987). An Efficient Auditory Filterbank Based on the Gammatone Function. Speech-Group Meeting of the Institute of Acoustics on Auditory Modelling, Malvern: IoA.

Patterson, R., & Moore, B. C. (1986). Auditory Filters and Excitation Patterns as Representations of Frequency Resolution. In B. C. Moore, & B. C. Moore (Ed.), Frequency Selectivity in Hearing (1st Edition ed., pp. 123-177). London: Academic Press.

Pertusa, & Inesta. (2009). Note Onset Detection Using One Semitone Filter-Bank for Mirex 2009. Music Information Retrieval Exchange 2009. ISMIR.

Pierce, J. R. (1983). Mr. New York: Scientific American Books.

Poliner, Ellis, Ehmann, Gomez, Streich, & Ong. (2007). Melody Transcription from Music Audio: Approaches and Evaluation. IEEE Transactions on Audio, Speech, and Language Processing. , 15 (4), 1247-1256.

Puckette, & Brown. (1998). Accuracy of Frequency Estimation Using the Phase Vocoder. IEEE Transactions on Speech and Audio Processing, 6 (2), 166-176.

Quach, Q. (2008, April 17). MATLAB – FFT and Zero Padding. Retrieved 12 20, 2011, from an Engineering and MATLAB blog: http://blinkdagger.com/matlab/matlab-fft-and-zero-padding/

Rabiner, L. (1977). On the Use of Autocorrelation Analysis for Pitch Detection. IEEE Transactions on Acoustics, Speech and Signal Processing, 25 (1), 24-33.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77 (2), 257-286.

Rao, V., & Rao, P. (2008). Melody Extraction Using Harmonic Matching. MIREX 2008. ISMIR.

Rao, V., & Rao, P. (2008). Vocal Melody Detection in the Presence of Pitched Accompaniment Using Harmonic Matching Methods. Proceedings of the 11th International Conference on Digital Audio Effects. Espoo: DAFx.

Roads, C. (1996). The Computer Music Tutorial. Cambridge: MIT Press.

Roland-Mieszkowski, D. M. (1994). Common Misconceptions about Hearing. Advanced Research and Development in Acoustics. Halifax, Canada: Digital Recordings.

Rossing, T., Moore, R., & Wheeler, P. (2002). The Science of Sound (3rd Edition ed.). San Francisco, USA: Addison Wessley.

Ryynanen, M., & Klapuri, A. (2004, October). Modelling of Note Events for Singing Transcription. Tutorial and Research Workshop on Statistical and Perceptual Audio Processing. Jeju, Korea: ISCA.

Schorkhuber, C., & Klapuri, A. (2010). Constant Q Transform Toolbox for Music Processing. 7th Sound and Music Computing Conference. Barcelona: SMC.

Schroeder. (1968). Period Histogram and Product Spectrum: New Methods for Fundamental Frequency Measurement. Journal of the Acoustical Society of America, 43 (4), 829-834.

Shannon, & Paliwal. (2003). A Comparative Study of Filter Bank Spacing for Speech Recognition. 21st Microelectronic Engineering Research Conference (pp. 1-3). Rochester, USA: RIT.

Smaragdis, P., & Brown, J. C. (2003). Non-Negative Matrix Factorization for Polyphonic Music Transcription. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (pp. 177-180). New Paltz: IEEE.

Smith, J. O. (2009). Audio FFT Filter Banks. International Conference on Digital Audio Effects (pp. 1-8). Como: DAFx.

Smith, J. O. (2010, April 23). Filter-Bank Summation Interpretation of the STFT. Retrieved August 6, 2011, from Spectral Audio Signal Processing: https://ccrma.stanford.edu/~jos/sasp/Filter_Bank_Summation_FBS_Interpretation.html

Smith, S. W. (1997). The Scientist and Engineer's Guide to Digital Signal Processing. San Diego, California: California Technical Publishing.

Stevens, S. S., & Warhofsky, F. (1965). Life Science Library - Sound and Hearing. New York: Life Books.

Stevens, Volkman, & Newman. (1937). A Scale for the Measurement of the Psychological Magnitude of Pitch. Journal of the Acoustical Society of America, 8 (3), 185-190.

Stracha, J. (2008, September). Wave Behaviours. Retrieved November 3, 2011, from KSSD Physics Course Material: http://www.kss.sd23.bc.ca/staff/jstracha/physics_11/course_material/unit7/U07L02.htm

Swets, J. A., Green, D. M., & Tanner, W. P. (1962). On the Width of Critical Bands. The Journal of the Acoustical Society of America, 34 (1), 108-113.

Szczerba, & Czyzewski. (2005). Pitch Detection Enhancement Employing Music Prediction. Journal of Intelligent Information Systems, 24 (2), 223-251.

Tan, Zhu, & Chaisorn. (2010). Mirex 2010 Audio Onset Detection. Mirex 2010. Utrecht: ISMIR.

Tavares, T. F., Barbedo, J. G., & Lopes, A. (2008). Towards the Evaluation of Automatic Transcription of Music. 6th Congress of the AES (pp. 96-99). Sau Paulo: AES.

Tolonen, & Karjalainen. (2000). A Computationally Efficient Multipitch Analysis Model. IEEE Transactions on Speech and Audio Processing, 8 (6), 708-716.

Tyagi, & Bourland. (2003). On Multi-Scale Fourier Analysis of Speech Signals. Dalle Molle Institute for Perceptual Artificial Intelligence. IDIAP-RR.

Uchida, & Wada. (2010). Simultaneous Estimation Method of Musical Instrument and Tone by Using MFCC. In B. Flinchbaugh (Ed.), Signal and Image Processing. 710, p. 24. Lahaina: Acta Press.

Veeneman, D. (1988). Speech Signal Analysis. In C. H. Chen, & C. H. Chen (Ed.), Signal Processing Handbook (1st Edition ed., pp. 511-548). New York: Dekker.

Wang, B., & Plumbley, M. (2005). Musical Audio Stream Separation By Non-Negative Matrix Factorization. Queen Mary, University of London, Department of Electronic Engineering. London: Queen Mary, University of London.

Wegel, & Lane. (1924). The Auditory Masking of One Pure Tone by Another and its Probable Relation to the Dynamics of the Inner Ear. The American Physical Society Review, 23 (2), 266-285.

Wen, & Sandler. (2007). Calculation of Radix-2 Discrete Multiresolution Fourier Transform. Signal Processing, 87 (10), 2455-2460.

White, P. (2001, August 1). Using Equalisation. Retrieved January 31, 2012, from Sound on Sound: http://www.soundonsound.com/sos/aug01/articles/usingeq.asp

Wilmering, T., Fazekas, G., & Sandler, M. (2010). The Effects of Reverberation on Onset Detection Tasks. 128th Audio Engineering Society Convention 2010 (pp. 666-677). London: AES.

Yegnanarayana, & Murty. (2009). Event-Based Instantaneous Fundamental Frequency Estimation From Speech Signals. IEEE Transactions on Audio Speech and Language Processing, 17 (4), 614-624.

Yeh, C. (2008). Multiple Fundamental Frequency Estimation of Polyphonic Recordings. Universite Paris VI, IRCAM. Paris: UPMC.

Yoshii, K., & Goto, M. (2010). Infinite Latent Harmonic Allocation: A Non parametric Bayesian Approach to Multipitch Analysis. International Society for Music Information Retrieval Conference, (pp. 309-314).

Zhou, R. (2006). Feature Extraction of Musical Content for Automatic Music Transcription. ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, à la faculté des sciences et techniques de l'ingénieur. Beijing: Lausanne, EPFL.

Zhou, Reiss, Mattavelli, & Zoia. (2009). A Computationally Efficient Method for Polyphonic Pitch Estimation. EURASIP Journal on Advances in Signal Processing, 2009 (Article ID 729494,), 1-11.

Zwicker, E. (1961). Subdivision of the Audible Frequency Range into Critical Bands. Acoustical Society of America, 33 (2), 248-248.

11 Appendices

11.1 Matlab – Peak Picker to Midi Program

```
clearvars
StatStore=[];
StatsNameStore=[];
StatsScoreStore=[];
%setup excel file construction
File='C:\Documents  and  Settings\Administrator\Desktop\FFT  Time  Res
Testing\test.xls';
 Excel = actxserver ('Excel.Application');
    if ~exist(File,'file')
        ExcelWorkbook = Excel.workbooks.Add;
        ExcelWorkbook.SaveAs(File,1);
        ExcelWorkbook.Close(false);
    end
    invoke(Excel.Workbooks,'Open',File);
files=dir('*.wav');
%loop round for every wav file present in *dir
for n=1:length(files);
    basename = files(n).name;
    basename(end-3:end) = '';
    data.(basename)=wavread(files(n).name);

      %strip 'wav' and 'orig' to get file number -  used to create
      trans file
    s = regexprep(basename, '.wav', '');
    s = regexprep(s, 'Orig', '');
    audioname=files(n).name
    number = int2str(n)
    midiname = strcat('file_tr',s, '.mid');
    origmidi = strcat('file',s,'.mid');

    wav = wavread(audioname);
    elements = max(size(wav));
    fs=44100;
    a4=440;
    threshold =0;
    msgNum =2;
    analysisMatrix1=[];
    analysisMatrix2=[];
    deltaStart=[];
    deltaEnd=[];
    noteLength=[];
    bestQ=[];
    bestRecall=0;
    bestPrecision=0;
    bestF-measure=0;
    bestConstant=0;
    Recall = 0;
    Ben = 1
    Precision = 0;
    Constant = 0.99;
    marker = 1;
    fftlength = 8192;
    hop = fftlength; %data Size
    fftSlide = 512; % amount slide along audio file
    fftTotal = 1;
    window = 1;
    %Envelope initiation
    E = hop;                 %points for the envelope - same size as
window length
    w = hanning(E);
    %zero pad length
    zerolength = fftlength-hop; %number of zeros required in adition
to hop to equal fft length
    %FFT Setup
    %----------------------------------------------------------
    m = fftlength;          % FFT Window length
    n = m; %pow2(nextpow2(m));  % Transform length
    f = (0:n-1)*((fs)/n);   % Frequency range buckets
    bucketsize = (2*fs/n)-(1*fs/n);
    halfbucket = bucketsize/2;

    %while loop to increase constant if precision isn't 1
    while Precision ~=1
        if Constant >0.99 %break loop if constant gets larger than
0.99
            break
        end
        front = 1;
        back = hop;
        previousFrame=[];
        midiNote=0;
        j=1;
        p=1;
        noteArray =0;
        timeArray = 0;
        msgNumStr=0;
        msgNumFin=0;
        noteFrameArray = [];
        endNotes=[];
        midiNotes=[];
        first=0;
        index=0;

        %-------------------------------------------------------------
        %note detection algorithm
        %-------------------------------------------------------------
        while back<elements %

         paddingCounter=front;
            for i=1:hop,
                window(i) = wav(paddingCounter);
                paddingCounter=paddingCounter+1;
            end;
            %envelope first 2048 elements of array to avoid peaks due
            to zero padding
            for paddingCounter=1:hop
                  window(paddingCounter) =
window(paddingCounter)*w(paddingCounter);
                  paddingCounter = paddingCounter + 1;
            end;
            %Zero Pad Window
            x=[window zeros(1,zerolength)];%add zeros to window
            %FFT on enveloped zero padded FFT
            y = fft(x,n);      % DFT
            power = y.*conj(y)/n;    % Power of the DFT
            fftTotal = fftTotal+power;
            %--------------------------------------------------------
            %calculate time of note start and end - round to 9
            decimal
            %places. First time around time =0
            %--------------------------------------------------------
            if j==1
                time=0;
            else
                time= round2((1/fs)*(front),9);
            end
            endtime=round2((1/fs)*(hop*j),9);
            %--------------------------------------------------------
-
            %find midi Note from FFT
            %--------------------------------------------------------
-
            %set threshold
            maxMag = 0;

                for i=1:fftlength/2, %only use 1st half of FFT
                   if power(i) > maxMag
                       maxMag = power(i);
                   end
                end
                threshold = maxMag*Constant; %threshold is set as a
fraction of the maximum magnitude in the current window
                threshtext = int2str(threshold);
                titleName = strcat('{\bf FFT }',threshtext);
                h = plot (f,power);
                xlabel('Frequency (Hz)')
                ylabel('power')
                title(titleName)

                j2=int2str(j);
                fftFilename = strcat('orig',number,'fftPlot',j2);
                saveas(h,fftFilename,'jpeg');

                %find which F bin the largest magnitude is in
                for i=1:fftlength/2, %only use 1st half of FFT
                    if power(i) > threshold %& threshold > 2
                        frequency = f(i);
                        if frequency > 20 & frequency<13000
                        %converts from frequency to closest MIDI note
                        midiNote = round(12*log2(frequency/a4))+69;
                        noteFrameArray(p) = midiNote;
                        p=p+1;
                        end
                    end
                end
                i=i+1;

            numberOfElements=length(noteFrameArray);
            %------------------------------------------------------
            %search frame for multiple notes the same value
            %------------------------------------------------------
            for i=1:numberOfElements-1
                for y=1:numberOfElements
                    %don't do anything with the note you are checking
```

```
- skip over
                    if y==i
                        y=y+1;
                    end
                    %if repeat note in frame note=0
                    if noteFrameArray(i)==noteFrameArray(y)
                       noteFrameArray(y)=0;
                    end
                end
            end
            %strip  zeros  from  current  frame  notes,  leaving  only
unique note numbers
            index = find(noteFrameArray);

            %---------------------------------------------------------
            %noteFrameArray contains notes present in current frame
            %---------------------------------------------------------
            noteFrameArray=noteFrameArray(index);

            %---------------------------------------------------------
            %set controllers for elseif statements below. checking of
there is a
            %previous note or a current note. 1=true, 0=false
            %---------------------------------------------------------
            if isempty(noteFrameArray)==1 %if noteFrameArray is empty
                thisFrame = 0;
            else
                thisFrame = 1;
            end

            if isempty(previousFrame)==1 %if noteFrameArray is empty
                lastFrame = 0;
            else
                lastFrame = 1;
            end

            %used to populate previous frame at end of loop - can't
use
            %NoteFrameArray as processing occurs in ifelse which can
empty the
            %array eg. if a note is present but alrady turned on.
            currentNotes = noteFrameArray;


%**********************************************************************
            %---------------------------------------------------------
            %IF  Statement  1  -  previous  note  true  and  current  note
            true
            %---------------------------------------------------------

            if thisFrame~=0 & lastFrame~=0
            %disp('if  statement  1  -  previous  note  and  current  note
            true')
                %-----------------------------------------------------
                %check if note in previous frame is in current frame
                %if not set end time for note in previous frame
                %-----------------------------------------------------

                %zero pad note arrays to make the same size
                a = length(previousFrame);
                b = length(noteFrameArray);

                if a>b
                    b=[noteFrameArray zeros(1,a-b)];
                    a = previousFrame;
                else
                    a=[previousFrame zeros(1,b-a)];
                    b = noteFrameArray;
                end

                for i=1:length(b),
                    ind=find(a~= b(i));
                    a=a(ind);
                end

                %strip zeros from end notes (zeros due to padding)
                endNotes=a;
                index = find(endNotes);
                endNotes=endNotes(index);   %this array contains the
notes that ended in this frame


                %-----------------------------------------------------
                  %write the notes that ended in this frame to
                  midiNotes
                %-----------------------------------------------------

                if isempty(endNotes)==0 %if end note not empty
                    for i=1:length(midiNotes(1,:))
                       for x=1:length(endNotes(1,:))
                        %if note in midiNotes not already ended then
                        end the note
                           %with current time
                            if      endNotes(x)==midiNotes(1,i)      &&
midiNotes(3,i)==0;
                                midiNotes(3,i)=time;
                          end
                       end
                    end
                end

                %-----------------------------------------------------
                  %check notes in current frame are not already
                  turned on
                %-----------------------------------------------------


                for i=1:length(midiNotes(1,:))
                       for x=1:length(noteFrameArray)
                            if   noteFrameArray(x)==midiNotes(1,i)   &&
midiNotes(3,i)==0;
                                %if current note exists in array and
                                end time is 0
                                %delete note from noteframearray
                                noteFrameArray(x)=0;
                            end
                       end
                end
                %strip note values of 0 from current notes
                index = find(noteFrameArray);
                noteFrameArray=noteFrameArray(index);


                %----------------------------------------------------
                  %Set start times and end time=0 for note in current
                  frame
                %----------------------------------------------------
                startTime=0;
                endTime=0;
                for i=1:length(noteFrameArray)
                    startTime(i) = time;
                    endTime(i) = 0;
                    thresh(i) = threshold;

                end

                if isempty(noteFrameArray)==0 %if noteframe not empty
                    startMidiNotes                                   =
[noteFrameArray;startTime;endTime;thresh];
                    midiNotes = [midiNotes,startMidiNotes];
                end


%*********************************************************************
            %--------------------------------------------------------
            %IF Statement 2 - previous note false and current note
            true therefore all new
            %notes are new, not continued notes.
            %--------------------------------------------------------
            elseif lastFrame ==0 & thisFrame~=0
j
             %disp('if statement 2 - previous note =0 and current
            note true')
                %----------------------------------------------------
                  %Set start times and end time=0 for note in current
                  frame
                %----------------------------------------------------
                startTime=0;
                endTime=0;
                for i=1:length(noteFrameArray)
                    startTime(i) = time;
                    endTime(i) = 0;
                    thresh(i) = threshold;
                end

                startMidiNotes                                       =
[noteFrameArray;startTime;endTime;thresh];
                midiNotes = [midiNotes,startMidiNotes];


%*********************************************************************
            %--------------------------------------------------------
            %IF Statement 3 - previous note true and current note
            false therefore no new
            %notes so end all previous notes still turned on
            %--------------------------------------------------------
            elseif lastFrame ~=0 & thisFrame==0
               %disp('if statement 3 - previous note true current
note 0')
               for i=1:length(midiNotes(1,:))
                    if midiNotes(3,i)==0;
                       midiNotes(3,i)= time;
                    end
                end


%***********************************************************************
            %----------------------------------------------------------
            %IF Statement 4 - previous note false and current note
            false therefore no new
            %notes and no previous notes so do nothing.
            %----------------------------------------------------------
            elseif lastFrame==0 & thisFrame==0
                 %do nothing
            end

            %set previous frame ready for next itteration
            if isempty(currentNotes)==1 %if noteFrameArray is empty
                    previousFrame = [];
            else
                    previousFrame = currentNotes;
            end


            noteFrameArray = []; %empty current notes
            endNotes = 0;
            thresh = [];

            front = front+fftSlide; %move analysis frame along by hop
size
            back = back+fftSlide;

            j=j+1;
            p=1;
        end

      %if there are notes detected at the end of process -    write
      the midi file
        %end any midi notes still left on.
        if isempty(midiNotes)==0 %if noteFrameArray is not empty
           for i=1:length(midiNotes(1,:))
                if midiNotes(3,i) ==0
                midiNotes(3,i) = time;
                end
           end

        Q = zeros(length(midiNotes(1,:)),6);
        %turn next note on
        Q(:,1) = 1;              % all in track 1
        Q(:,2) = 1;              % all in channel 0
        Q(:,3) = midiNotes(1,:);      % note numbers
        Q(:,4) = 73;             % volumes
        Q(:,5) = midiNotes(2,:);      % note on
        Q(:,6) = midiNotes(3,:);       % note off

        %write matrix to mid file
```

```
midi_new = matrix2midi(Q);
writemidi(midi_new, midiname);

%enter threshold and magnitude in to data for spreadsheet
for i=1:length(Q(:,1))
    Q(:,7) = midiNotes(4,:);
end

if marker == 1
    %get original midi file notes
    origmidi = readmidi(origmidi);
    Notes = midiInfo(origmidi,0);
    analysisMatrix1=[];

    for i=1:length(Notes(:,1))
    analysisMatrix1(i,1) = Notes(i,5);
    analysisMatrix1(i,2) = Notes(i,6);
    analysisMatrix1(i,3) = Notes(i,3);
    end
end
analysisMatrix2=[];
for i=1:length(Q(:,1))
    analysisMatrix2(i,1) = Q(i,5);
    analysisMatrix2(i,2) = Q(i,6);
    analysisMatrix2(i,3) = Q(i,3);
end

%calculate recall and precision etc.
E = evaluate(analysisMatrix1,analysisMatrix2);
Recall = E(3)/E(1)
Precision = E(3)/E(4)
F-measure = (2*Recall*Precision)/(Recall+Precision);

if F-measure>bestF-measure
    bestQ=Q;
    bestRecall=Recall
    bestPrecision=Precision;
    bestF-measure=F-measure;
    bestConstant=Constant;
else
    if ben ==1
    bestQ=Q;
    ben=2;
    end
end


if Precision ==1 || Constant > 0.98

    Q=bestQ;
    Recall=bestRecall;
    Precision=bestPrecision;
    F-measure=bestF-measure;
    Constant=bestConstant;

    for i=1:length(Q(:,1))
        deltaStart(i,1) = Q(i,5)-Notes(i,5);
        deltaEnd(i,1) = Q(i,6)-Notes(i,6);
        noteLength(i,1)     =     (Q(i,6)-Q(i,5))-(Notes(i,6)-
Notes(i,5));
            end

            %column headings for spreadsheet
            P                                                                   =
{'channel','track','note','velocity','start','end','thresh','deltaSta
rt','deltaEnd','DeltaNoteLength','recall','precision','F-
measure','Constant'};
            PP                                                                  =
{'channel','track','note','velocity','start','end','On  Mess  #','Off
mess #',' '};

            %write headings to spreadsheet
            Data = [PP,P];
            xlswrite1(File,Data,audioname,'A1');

            %write midi files to spreadsheet
            Data = [Notes];
            xlswrite1(File, Data, audioname,'A2');
            Data = [Q];
            xlswrite1(File, Data, audioname,'J2');
            Data = [deltaStart,deltaEnd,noteLength];
            xlswrite1(File, Data, audioname,'Q2');
            Data = [Recall,Precision,F-measure,Constant];
            xlswrite1(File, Data, audioname,'T2');
            StatsName = {midiname};
            StatsNameStore = [StatsNameStore;StatsName];
            StatsScore             =             [Recall,Precision,F-
measure,Constant,fftlength,fftSlide];
            StatsScoreStore = [StatsScoreStore;StatsScore];
            break
        end

        else
            disp('!!!!!!!!!!!no notes were detected!!!!!!!!!!!')
        end

        if Precision~=1
            Constant = Constant+0.01
            marker = 0; %marker used to trigger/stop excel write
        end
    end
end

%Write data to excel spreadsheet
P = {'file','recall','precision','F-measure','Constant','fft length',
'hop'};
xlswrite1(File, P, 'TotalStats','A1');
xlswrite1(File, StatsNameStore, 'TotalStats','A2');
xlswrite1(File, StatsScoreStore, 'TotalStats','B2');


%Close excel file.
invoke(Excel.ActiveWorkbook,'Save');
Excel.Quit
Excel.delete
clear Excel
disp('Process Complete')
```