

**MSc Project Report**

**A Real-Time System for Automatic Vocal Harmonization**

**Name: Daniel Ayodele Oluyomi**

**Student Number: 120383168**

**Supervisor: Dr Simon Dixon**

**Submission Date: 2nd September 2013**

# ACKNOWLEDGEMENTS

Firstly, I am extremely thankful to GOD Almighty for his blessings throughout the project endeavours. I want to thank my supervisor Dr. Simon Dixon for his suggestions, encouragement and guidance. Furthermore, I would like to thank his research group and in particular, Dr. Matthias Mauch for his valuable feedback early on in the project.

I would also like to thank Sam Myer for his incredible help from his work on chord detection. Finally, I would like to thank my family for their continuous support.

# ABSTRACT

The definition of harmonization in the context of this report is the addition of scale tones to an existing melody to form harmony. The process of performing this task automatically is part of the music information retrieval problem domain, which include examples such as automatic transcription and music annotation. Previous commercial approaches have seen the use and development of pitch and chord detection algorithms as a solution for automatic harmonization usually implemented as an offline process.

This project implementation report documents the design and execution of a real time system whereby the harmonic information of a polyphonic instrument is directly applied to another monophonic instrument, specifically the human voice. The intention was to develop an algorithm capable of synthesizing harmonies in real time from a vocal melody with respect to the detected chords generated by an accompanying instrument. The algorithm in its current form optimises existing methods for pitch and chord detection and is able to sufficiently detect the pitch of a human voice and the chord of an accompanying instrument in real time.

Table of Contents

[ACKNOWLEDGEMENTS 2](#_Toc365876925)

[ABSTRACT 3](#_Toc365876926)

[1. INTRODUCTION 6](#_Toc365876927)

[1.1 Project Overview 6](#_Toc365876928)

[1.2 Objectives and Scope of the Project 8](#_Toc365876929)

[1.3 Organisation of the Project Report 9](#_Toc365876930)

[2. BACKGROUND RESEARCH 11](#_Toc365876931)

[2.1 What is a Note 11](#_Toc365876932)

[2.2 What is a Chord 14](#_Toc365876933)

[2.3 Technical Concepts 17](#_Toc365876934)

[2.3.1 Pitch Detection 17](#_Toc365876935)

[2.3.2 Pitch Shifting 20](#_Toc365876936)

[2.3.3 Chord Detection 24](#_Toc365876937)

[2.4 Related Work 32](#_Toc365876938)

[3. REQUIREMENTS 33](#_Toc365876939)

[3.1 Harmonizer properties 33](#_Toc365876940)

[3.2 Pitch Detector 33](#_Toc365876941)

[3.3 Pitch Shifter 34](#_Toc365876942)

[3.4 Chord Detector 34](#_Toc365876943)

[3.5 Audio input expectations 34](#_Toc365876944)

[4. DESIGN 36](#_Toc365876945)

[4.1 Pitch Detection 36](#_Toc365876946)

[4.1.1 Autotune 39](#_Toc365876947)

[4.2 Pitch Shifting 40](#_Toc365876948)

[4.3 Chord Detection 42](#_Toc365876949)

[4.4 Harmony Synthesis 46](#_Toc365876950)

[5. IMPLEMENTATION & TESTING 49](#_Toc365876951)

[5.1 Pitch Detection 50](#_Toc365876952)

[5.2 Pitch Shifting 54](#_Toc365876953)

[5.3 Chord Detection 58](#_Toc365876954)

[5.4 Harmony Synthesis 60](#_Toc365876955)

[5.5 Latency 65](#_Toc365876956)

[5.6 Evaluations 67](#_Toc365876957)

[6. CONCLUSIONS AND FURTHER WORK 68](#_Toc365876958)

[6.1 Future Work 69](#_Toc365876959)

[REFERENCES 70](#_Toc365876960)

[APPENDIX A – Note Frequencies under equal tuning 74](#_Toc365876961)

[APPENDIX B - MIDI, Frequency, Pitch and Period Conversion Table 75](#_Toc365876962)

[APPENDIX C – Formants 77](#_Toc365876963)

[APPENDIX D – Results of Soak Test 79](#_Toc365876964)

[APPENDIX E – Chord Annotations 80](#_Toc365876965)

# 1. INTRODUCTION

During the last few years, considerable progress has been made in the development of music information retrieval techniques. To highlight the research outcomes, offline software applications such as Sonic Visualiser (CENTRE FOR DIGITAL MUSIC, QUEEN MARY, UNIVERSITY OF LONDON, 2010) have been engineered to provide a number of visualisation tools aimed for the analysis and the annotation of music audio files (CANNAM, C. et al., 2010) . One area of music informatics research that has seen little progression however, is the application of transcription tools for real time performances.

In recent times, there has been an increased reliance on the use of digital implementations of audio effects for live performances. Typical examples include digital reverberations for singers and digital replications of stomp box effects such as phasers and flangers for guitars. Even controversial pitch controlled effects such as pitch correction are being utilised more frequently by singers on stage (SAVAGE, M., 2010) . Naturally, the next step for live performances is the exploitation of audio effects, which currently are only capable of being run an offline manner.

## 1.1 Project Overview

Ergo, the proposed system discussed in this report is set on creating distinguishable backing vocals found in a typical band setting of a lead singer with two backing singers. The additional information necessary to harmonize the backing vocals is obtained through the real time harmonic analysis of an accompanying polyphonic instrument. In other words, the system deals with the real time detection of chords from the performance of a single polyphonic instrument and its application to the modification of another monophonic instruments performance via pitch detection and pitch shifting.

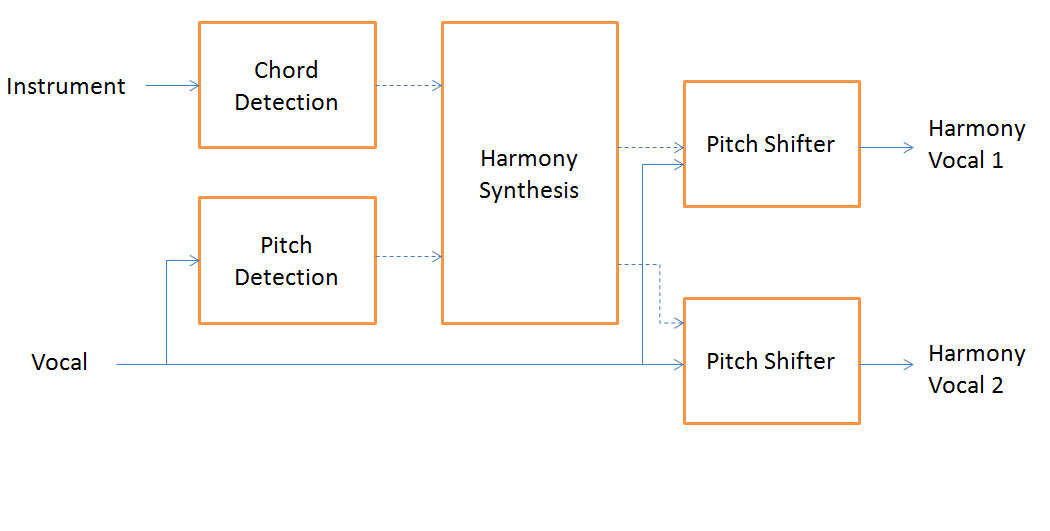


Figure 1 - Proposed System for Real-Time Automatic Vocal Harmonization

For the singer/performer who lacks the budget for backing vocalists, or for the amateur musician, this system is intended as a harmony device to enhance the live vocal performance.

The proposed framework, shown in Figure 1, contains all the system’s building blocks. The components within blocks highlight the type of processing required from this real time system. The arrow lines represent the flow of audio and dashed arrow lines represent a control signal. The pitch and chord information obtained from the respective detectors will be used as control signals to determine a suitable harmony.

At this time of writing, there are very few, if any, software applications attempting to achieve the same design functionality. Zplane’s vielklang (ZPLANE, 2013) comes relatively close as it advertises its ability to produce instant harmony to any melody. It however, processes the harmonies off-line for editing and is aimed primarily for the mixing environment.

## 1.2 Objectives and Scope of the Project

The aim of the project is to design, implement and evaluate a system that acts as a vocal processing tool to aid a musician using existing algorithms. The desired goal of the system is to imitate the harmonies sung by backing vocalists in a performance using the lead singer’s voice. To accomplish this, the system must be able to detect the pitch of a vocalist and the chords of an accompanying instrument in real time. After this, the vocal should be pitch shifted, generating up to two additional voices, based on the detected chord to harmonize with the original vocal melody line simultaneously.

As latency must be minimized as much as possible, this will prohibit the use of complex methods and algorithms. Its implementation as a program to run in real time prevents the use of offline software applications such as MATLAB as a viable programming platform. Instead, it was decided early on to write the program in C++ using the JUCE (Jules' Utility Class Extensions) class library, which takes advantage of the Virtual Studio Technology (VST) software development kit (SDK). Developed by Steinberg , VST is the dominant plug-in format on the market, supported by the majority of digital audio workstations (DAW)(STEINBERG MEDIA TECHNOLOGIES GMBH, 2011). VST’s are by default, audio plug-in systems that operates real time. The result would see the framework implemented as a VST plug-in capable of running inside the increasing amount of digital audio workstations (DAWs) that support VSTs.

As stated previously, the emphasis of this project is about the design and implementation of the framework, which allows the real time automatic vocal harmonization. Therefore, any existing algorithms that could be adapted to advance the project were utilised where possible. For example, the work on chord detection techniques achieved by Matthias Mauch (MAUCH, Matthias, 2010) and Sam Myer (MYER, Sam, 2011) in their respective papers, were instrumental to the progress made on the system framework. More information about their methodologies are explained in future chapters.

Offline chord detection success rates from the 2012 MIREX Chord Detection tasks (MIREX, 2012), peak at 83%. The leading algorithm is a system that is based completely on machine learning techniques with no reference to expert human knowledge (NI, Yizhao et al., 2012). Attempting to achieve 100% accuracy of a detected chord in real-time is clearly an unrealistic expectation hence the targeted goal for this system was 70%.

## 1.3 Organisation of the Project Report

The structure of this report mostly follows the chronology of the progress made. First, basic concepts pertaining to relevant music theory are presented in chapter 2 as well as technical concepts regarding existing algorithms necessary to realise the system framework. Based on the notions presented, a requirements specification is summarised in the following chapter. Afterwards, the design of the several framework components are examined in Chapter 4. Chapter 5 focuses on the implementation and testing of the real time system. The report concludes with a summary of the accomplished targets and suggestions on potential improvements for future work.

# 2. BACKGROUND RESEARCH

This chapter includes all the information necessary to arrive at a stage where a sufficient design specification can be formed. First, it is necessary to discuss relevant musical concepts and terminology. Music can be defined as organised sound. Although the semantics of music are subjective by nature, there are certain music principles that are generally recognised by human listeners.

## 2.1 What is a Note

A note is typically described as a sound event that encapsulates four perceptual qualities: duration, timbre, loudness and most importantly pitch (THOMAS, J. M., 2012).

The perceived duration of a note is nearly identical to the physical duration of a sound. Timbre is the most subjective feature of a sound and consists of properties (e.g. instrumentation, playing style) that discriminate two notes of the same pitch, duration and loudness. The perceived loudness of a sound is represented computationally as the mean-square power on a logarithmic (decibel) scale barring any reference to complex psychoacoustic factors (KLAPURI, Anssi and Davy, Manuel, 2006).

The terms “pitch” and “frequency” are typically used interchangeably when considering a specified note. Nevertheless, while closely correlated, pitch and frequency are not equal. Frequency is described as the number of cycles per unit time (units in Hz), and is mathematically defined as:

Equation 1 - Frequency Equation (where f is frequency and T represents the time period in seconds)

Pitch however, is considered as “*an attribute of an auditory sensation to the perceived position of a sound on a scale from low to high* (ANGUS, J and Howard, D, 2009)”. It is clear that both mathematical concepts and the human perception of sound must be taken into account together when describing pitch [1].

Once a pitched sound is made, an acoustic pressure waveform is produced and repeated periodically. Every repetition is considered a cycle and the length of time the cycle lasts is known as the fundamental period of a waveform. The number of cycles that occur in one second therefore, equals the fundamental frequency of that waveform.

For a real world signal, a pitched note however will also consist of partials at near-integer multiples of a given fundamental frequency (f0) dependent on the timbre of the audio signal among other things. The nth partials of the fundamental frequency will have a frequency of approximately (n x f0).

Table 1 shows the harmonic series for a periodic sine wave with a f0 of 261.63 Hz (C4) middle C. It is interesting to note that the first five overtones consist of the major triad based from the fundamental frequency.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Harmonic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Frequency | f0 | 2f0 | 3f0 | 4f0 | 5f0 | 6f0 | 7f0 | 8f0 |
| Ratio | 1/1 | 2/1 | 3/2 | 4/3 | 5/4 | 6/5 | 7/6 | 8/7 |
| Pitch | C3 | C4 | G4 | C5 | E5 | G5 | Bb5 | C6 |
| Interval | I | I | V | I | III | V | VII- | I |

Table 1 - Harmonic Series

Changes in intensity and duration of a signal can influence humans’ perception of pitch as Gerhard’s report on the history of pitch extraction (GERHARD, D., 2003) confirms. Nevertheless, according to the place theory of human pitch perception (ANGUS, J and Howard, D, 2009), the fundamental frequency of a sound along with the spacing of its partials, remain the most important aspects when perceiving the pitch of a sound.

This assumption provides the basis for finding the fundamental frequency of a sound when attempting to determine its pitch (GERHARD, D., 2003).

The mapping of frequencies to musical pitches is determined by two factors; the concert pitch and the tuning system. In order to allow multiple instruments to perform together, the concert pitchis a reference frequency that all instruments are tuned to. This is generally set to 440 Hz (A4).

Different tuning systems exist such as just intonation, well temperament and equal temperament. For this particular line of work, assuming that the audio signal is in equal temperament holds for the majority of contemporary music. In *equal temperament,* the octave is divided into twelve semitones with equal frequency spacing. As the octave represents a ratio of two, the ratio of frequencies between two adjacent semitones is the twelfth root of two. To find the frequency of a certain pitch the equal tempered tuned frequency of any note can be calculated as follows;

**Equation 2 - MIDI to Hz conversion [[1]](#footnote-1)**

In addition, any given frequency can be converted to its corresponding equal tempered pitch in following fashion;

Equation 3 - Hz to MIDI pitch conversion

This process of pitch/frequency mapping offers a handy way of working with pitch algorithmically. Appendix A shows a visual representation of the conversion formula.

## 2.2 What is a Chord

Before introducing the definition of a chord, it is important to start with the basics of its formation. An interval is the distance between two pitches. The perspective of an interval alters according to the note polyphony. If more than one note is played simultaneously, intervals are seen from a vertical approach representing harmony. Intervals are considered in context with a melody line when single notes are played successively. Starting with C, the ascending interval order is shown in Table 2.

All pitches of an octave relationship are grouped into a set called a pitch class. The pitch class of C contains the elements (..,C2, C3, C4...). The octave number is added to the pitch name to help differentiate the notes within the set.

|  |  |  |
| --- | --- | --- |
| Interval Name | # of semitones | Interval |
| Perfect Unison | 0 | C – C |
| Minor Second | 1 | C – C# |
| Major Second | 2 | C – D |
| Minor Third | 3 | C – D# |
| Major Third | 4 | C – E |
| Perfect Fourth | 5 | C – F |
| Tritone | 6 | C – F# |
| Perfect Fifth | 7 | C – G |
| Minor Sixth | 8 | C –G# |
| Major Sixth | 9 | C – A |
| Minor Seventh | 10 | C –A# |
| Major Seventh | 11 | C – B |
| Perfect Octave | 12 | C – C |

Table 2 - Intervals

When three or more notes appear near simultaneously and are perceived as a single musical entity, it is defined as a chord. Depending on the relationships between the note intervals comprising a chord, it can be expressed as consonant (stable, harmonious) or dissonant (unstable, displeasing).

The most fundamental chords are triads, consisting of three notes, the first, which is called the *root* note, a third and the fifth. A triad containing the root note, a major third interval then a minor third interval is called a *major chord*. *Minor chords* consist of the root note, a minor third interval followed by a major third interval.

A piece of music typically has at least one major or minor chord that represents the harmonic foundation. This particular chord is defined as the *key* of the piece of music. In the key of C using a major scale, playing a triad for each note in the scale gives the following chords: (Note: Major triads are represented by Roman numerals in uppercase while minor triads are represented in lowercase.)

|  |  |  |  |
| --- | --- | --- | --- |
| Scale Degree Name | Chord Name | Roman Numeral | C Major Notes |
| Tonic | C Major | I | C,E,G |
| Supertonic | D Minor | ii | D,F,A |
| Mediant | E Minor | iii | E,G,B |
| Subdominant | F Major | IV | F,A,C |
| Dominant | G Major | V | G,B,D |
| Submediant | A Minor | vi | A,C,E |
| Leading Tone | B Diminished | viio | B,D,F |

Table 3 - Summary of C Major Triad chords

The characteristics of chords evolve from their progressions that are established within a music piece. The most common chord progression in popular music is based on the three main degrees tonic, subdominant and dominant.

## 2.3 Technical Concepts

The following technical concepts introduced in this section will be directly relevant to algorithms used for the design and implementation of the vocal harmonization framework. Most of the tools mentioned involve the use of the short-time Fourier transform and to a lesser extent, probabilistic models that the reader is assumed to have some knowledge of.

### 2.3.1 Pitch Detection

The first and arguably most important aspect of the harmonizer’s abilities begins with the detection of the pitch of the vocal audio signal. Larson in his paper (MADDOX, R and Larson, Eric, 2005) stresses three important characteristics all real-time pitch trackers should display.

1. *Computation Time*
2. *Determination of voiced and unvoiced segments*
3. *Accuracy*

The difference in the pitch detector’s estimate compared to the true value must be as small as possible. The difficulty of finding the fundamental frequency of a waveform is dependent on the waveform itself. A waveform with less partials will make the period easier to detect. Most voice signals will have various partials that constitute most of the errors that pitfall pitch detectors (GERHARD, D., 2003). Furthermore, the unvoiced segments that include plosives and fricative consonants adversely affect pitch detectors accuracy.

The YIN algorithm was chosen as the vocal pitch detector based on evidence of its suitability given in (KNESEBECK, A. von dem and Zölzer, U, 2010) and (GERHARD, D., 2003).

The YIN algorithm’s foundation is based on the auto-correlation function (ACF) (KAWAHARA, Hideki and Cheveigné, Alain de, 2002). Five error correction schemes are then applied sequentially to end up with a robust form of pitch detection. Each scheme will be briefly discussed starting with the foundation.

The autocorrelation function (ACF) compares a signal to a time-shifted version of itself via convolution in the time domain. For a discrete signal x:

Equation 4 - ACF adapted from (KAWAHARA, Hideki and Cheveigné, Alain de, 2002)

where is the ACF of lag and with representing the window size. Given a periodic signal, peaks at period multiples of the fundamental frequency will arise. As the highest peak will occur when = 0, the location of the second highest peak gives an estimate of the period, and the height gives an indication of the periodicity of the signal. The sample rate is divided by the found period (in samples) to obtain f0.

A general problem with the ACF is its sensitivity to amplitude variations (KNESEBECK, A. von dem and Zölzer, U, 2010). An increase in the signal’s amplitude encourages peaks to grow as lag increases with the opposite also true. Sub periodic and super periodic errors whereby the selected peak is an octave below or above the correct peak respectively, occur often as a result.

The first adjustment the YIN algorithm introduces is the “Difference function” which reduces the ACF’s weakness to amplitude variations. The difference function measures the amount of energy in the signal “*which cannot be explained by a periodic signal of period* ” (DIXON, Simon, 2013). This is represented in the equation of the sum of squared differences where the period is found when the equation is approximately equal to zero.

Equation 5 - Difference Function adapted from (KAWAHARA, Hideki and Cheveigné, Alain de, 2002)

The difference function at this point is still susceptible to amplitude changes causing sub periodic errors. The cumulative mean normalised difference function (CMNDF) deals with this by averaging the current lag value with the previous values (KNESEBECK, A. von dem and Zölzer, U, 2010). The CMNDF starts with a value of one, dropping only when the current lag value is below the average of all the previous lags (KNESEBECK, A. von dem and Zölzer, U, 2010).

Equation 6 - CMNDF adapted from (KAWAHARA, Hideki and Cheveigné, Alain de, 2002)

As real world signals are never entirely periodic, the absolute threshold is introduced to select the smallest lag value that has a minimum lower than a pre defined threshold value (typically 0.1). The global minimum is chosen if none fall below the threshold value.

The fourth scheme is quadratic interpolation to allow sub sample accuracy, increasing the resolution of the result.

The final improvement is the best local estimate. It is described in the paper as median smoothing (KAWAHARA, Hideki and Cheveigné, Alain de, 2002). Once an estimate that has crossed the absolute threshold is found, there is a search for another minimum within a smaller range. This step reduces the error rate by 0.27% but increases latency.

### 2.3.2 Pitch Shifting

Pitch shifting is a well researched topic with various techniques devised to ultimately change the pitch of a signal without changing its length. A pitch-shifting algorithm’s ability to maintain the positions of the original formants (peaks in the spectral envelope) was of utmost importance when setting requirements. Furthermore, the algorithm’s capability for live use as well as it effectiveness against unwanted artefacts were sought after traits.

Developed in 1989, Lent’s Algorithm is a time domain pitch shifting method able to preserve the formants of the audio signal. In order to perform effectively, it needs the detected period of the current pitch and the desired period prior to processing.

The method starts by dividing a frame of audio data into small windowed segments   
(using a Hann window), where each segment holds a single period of the input signal (COSTELLO, Sean, 2009). Then Hann windowed segments are generated at a rate corresponding to the desired period. Lent's Algorithm does not actually obtain information regarding the formants, but inherently preserves them (JOHNSON, R. B., 1995).

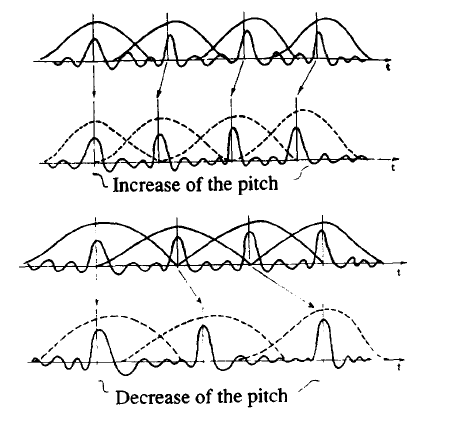


Figure 2 - Modifying pitch by moving Hann windows (GERBER, S and Scott, R., 1972)

The algorithm consists of three sections: a pitch tracker, time compressor/expander and a pitch shifter. To start, two arrays are generated to store input and output samples respectively. An input pointer and output pointer are set to the beginning of their respective array. The input pointer is incremented until its respective array is full. The first part of the algorithm uses a fourth order infinite impulse response (IIR) band pass filter on the input samples. A method of pitch detection known as the zero-crossing method follows.

The second part of the algorithm is the time compression/expansion decision. This section is reached when a zero crossing within the signal occurs (when a period is found). The decision is to find out whether the output pointer is running ahead of or behind the input pointer. This is necessary to establish whether periods of the input signal should be repeated (representing expansion) or skipped (representing compression). The algorithm allows the use of different input and output array lengths so it makes the comparison proportionally as:

Equation - Compression/expansion decision

Once a period has to be copied, only then will the algorithm move on to the pitch-shifting element. The pitch shifting section updates the output pointer position to represent the period length. The unprocessed input audio frame is then sampled with overlapping Hann windows where the width between each Hann window peak is the original period length. The windowing stops when the next Hann window will extend past the end of the current input frame. To output at a different f0**,** the Hann windows are arranged such that the length between the overlapping window peaks consists of the desired output period length. Increasing or reducing the period length between the windows is all that is required to increase or decrease the fundamental frequency (LENT, Keith, 1989). The block diagram of the processes involved in the original algorithm is presented in Figure 3.

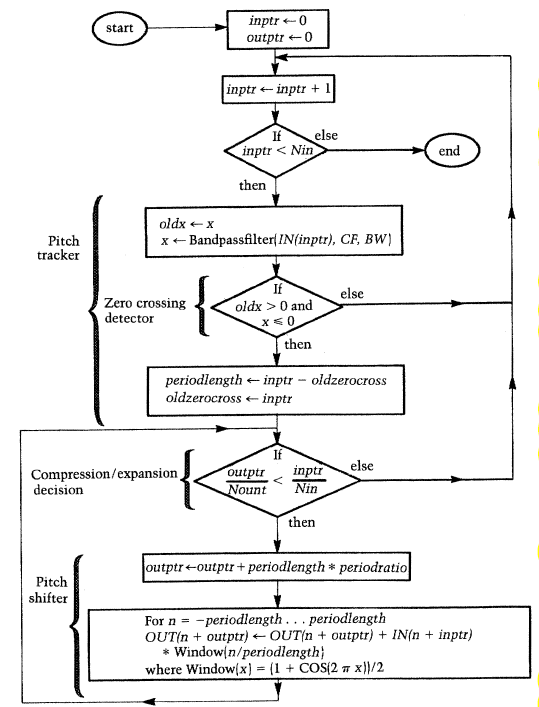
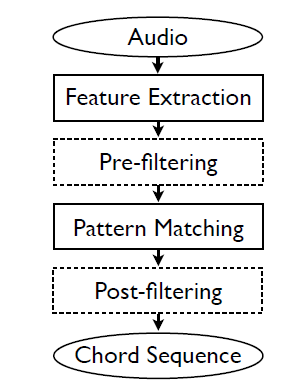


Figure 3 - Flow Diagram of Lent's Algorithm (LENT, Keith, 1989)

### 2.3.3 Chord Detection

The majority of available chord recognition systems follow a similar architecture consisting of four significant steps depicted in Figure 4. Each stage is discussed briefly. The specific algorithms chosen for chord detection taken from (MAUCH, M and Dixon, S., 2010) and (MYER, Sam, 2011) are examined.

Feature Extraction

The basis of most methods for extracting features of an audio signal lies with the chroma vector (also known as a pitch class profile) (MAUCH, Matthias and Dixon, Simon, 2010). It is a twelve-dimensional vector of real numbers representing the energy of the twelve pitch classes. The twelve pitch classes are derived via frequency bin mapping of the DFT spectrum to the corresponding pitch class for each fundamental frequency (SHEPARD, R., 1999). A chromagram therefore, is a sequence of chroma vectors that describes the pitch class content of an audio signal over time. With multiple methods devised to calculate chromagrams, it is recommended that the reader view (CHO, Taemin et al., 2010) for detailed information.

Pre-filtering

The next stage deals with filtering the chromagram in order to increase resistance against noise and transients affecting the feature values from the audio signal. The frame rate of the chromagram varies depending on the method used but usually resides between 24-256 ms (CHO, Taemin et al., 2010). This rate is purposely a great deal faster than the usual chord change rate in music to identify chord boundaries precisely.

Figure 4 - The basic architecture of the standard approach to automatic chord recognition (dotted lines indicate optional steps) (T.CHO et al., 2009)

Pattern Matching

There are two approaches to pattern matching that the majority of current methods operate with; deterministic or probabilistic. The deterministic approach promotes the use of a binary chord template (bit mask), which consists of a 12 by 1 vector containing a “1” where notes are present and “0” elsewhere. A C Major chord for example, would be represented as [1,0,0,0,1,0,0,1,0,0,0,0] (MAUCH, Matthias and Dixon, Simon, 2010). There exist many different matching methods to classify chords. One example is done by calculating the hamming distance between the

bit mask of the detected chord tones with each chord template bit mask (KNESEBECK, A. et al., 2011). The hamming distance between two bit masks are the number of positions where the vector elements are different. Therefore, the bit mask with the minimum hamming distance is regarded as the chord candidate for the frame.

Within probabilistic models, lie various approaches. Gaussian-based methods have the binary templates replaced with chord models specified by a multivariate Gaussian distribution in terms of the mean and covariance matrix. Jiang states that for Hidden Markov Model based approaches, “*in addition to Gaussian models, transition probabilities are needed to express the pass over from one chord label to another* (JIANG, Nanzhu et al., 2011)*”.* The probabilities are specified either manually from musical knowledge or automatically via training data. Furthermore, Jiang emphasises that *“a Viterbi decoder is needed to find the chord label sequence that jointly maximises the output probabilities defined from the Gaussian distributions and transition probabilities* (JIANG, Nanzhu et al., 2011)*”.*

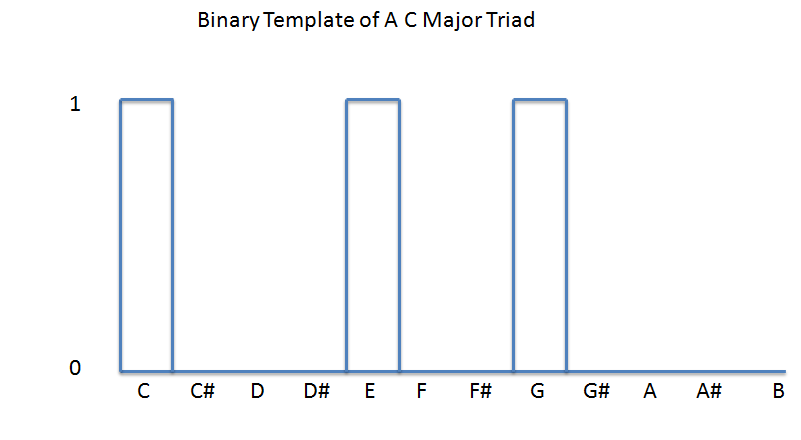


Figure 5 - Typical Binary Template

Post-filtering

This final stage is used to the smooth the current predicted chord sequences. This is usually performed by a Viterbi decoder, which locates the most probable progression of chords according to the chord-type probabilities calculated in the pattern matching step.

Going back to the first stage of chord detection algorithms, extracting the correct features from the audio is naturally the most important step as evidenced by the results seen from (CHO, Taemin et al., 2010). One problem chromagram extraction methods have to deal with is the impact of overtones. It has been made clear in (MAUCH, Matthias and Dixon, Simon, 2010) that chord identification is more difficult when note partials are not multiples of the fundamental frequencies of the chord notes. This occurs for example when playing an inversion of any triad chord. For example, if a first inversion C Major chord is played, the bass note’s first five partials will consist of E,E,B,E,G#. As B and G# are not part of the C major chord, this causes confusion when attempting to identify note salience via locating peaks across the magnitude spectrum of audio.

They present a new method for extracting chromagrams using the non-negative least squares algorithm devised in (HANSON, R. J. and Lawson, C. L., 1974) to form an approximate transcription of note start values. They argue that no partials would interfere with the signal if there were a perfect transcription of the notes prior to chroma extraction methods. An approximate transcription is obtained instead by using note salience calculation which involves using MIDI notes to find spectral peaks for every note in every frame. A more detailed description can be found from (MAUCH, Matthias and Dixon, Simon, 2010) and (MAUCH, M and Dixon, S., 2010). Preceding the chromagram extraction, there are three pre-processing steps performed.

First, because pitch is linear in log frequency, calculating pitch class representations in the frequency domain require a conversion from frequency to log frequency. Hence, a log-frequency spectrogram is formed with a resolution of three bins per semitone. The first process is to automatically find the reference frequency of the audio signal and adjust the spectrogram to compensate.

The next process involves minimising broadband noise and timbre fluctuations with a moving standardisation. This entails the subtraction of a smoothed spectrum, where any negative value is set to zero, and dividing by the standard deviation estimate of the amplitude.

Given a pre-processed log frequency spectral frame and a predefined note profile matrix, the goal was to find the note start vector that minimises the Euclidian distance

Equation – Minimising the Euclidian distance

that best approximates the log frequency spectral frame. Using the NNLS algorithm solves the problem resulting in the collection of a unique activation vector.

This very vector represents the semitone spectrum that is then used along with bass and treble note profiles to map bass and treble chromagrams respectively. To transcribe the chords, a beat-synchronous chromagram is generated from the chromas. Deductions are then formed using the Viterbi algorithm on a dynamic Bayesian network where the treble and bass chromas are the observed variables.

A Bayesian network is a graphical probabilistic model that symbolizes a set of random variables. Its conditional dependency structure is represented as a directed acyclic graph where every node signifies one random variable and every directed edge represents a direct dependency. A dynamic Bayesian network therefore, is a BN that represents sequences of random variables over neighbouring time steps. The random variables are interconnected through conditional probabilities. The network model used in the paper is shown in figure 6. The model is a time-invariant first order Markov process where the probability of being in state Z at time t given all previous states up to time t-1 depends only on the previous state Z-1. To add, a hidden Markov model (HMM) is just a Markov process augmented by a set of observable states and probabilistic relations between them and hidden states (BOYLE, Roger, Unknown).

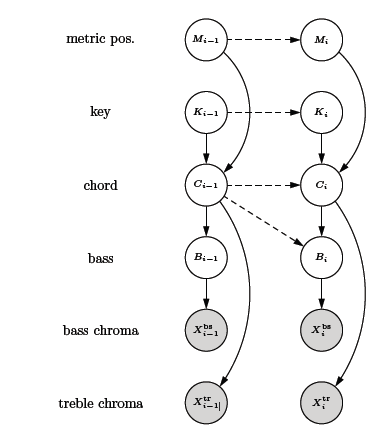


Figure 6 - DBN topology used in (MAUCH, Matthias and Dixon, Simon, 2010). Clear nodes are the random variables; the observed nodes are the chromas

The model is able to distinguish 121 different states: 12 for each of 10 chord types with 63.3% accuracy (MAUCH, Matthias and Dixon, Simon, 2010). These consist of the following: major, major in first inversion, major in second inversion, major 6th, major 7th, minor, minor 7th, dominant 7th, diminished, augmented and a no chord state.

Myer’s work on real time chord detection mostly follows the architecture mentioned previously (MYER, Sam, 2011). A simple DBN network similar to Mauch’s work from (MAUCH, Matthias and Dixon, Simon, 2010) is used omitting the metric and key nodes. In Mauch’s model, the initial state is assumed to be uniformly distributed. However, in Myer’s work, chord type initial probabilities are set. Two chromagrams representing a midrange and bass frequency range respectively are generated via the method described in (STARK, Adam and Plumbley, Mark, 2009). The midrange chroma is used to estimate the chord and the bass chroma, to approximate the root note. To estimate the root note probability, only three chroma values were considered: the root note, the perfect fourth and the perfect fifth. This was due to the root and the fifth notes being the most frequently played notes by the bass in contemporary music.

Trained Gaussian probability distributions are used to estimate chord probabilities at each window frame with the parameters of the Gaussian pdf function being the 12 values of the normalized chromagram. In his work, chord change probabilities are based on the observation that chords tend to be several frames long and so it is probable that a chord will not change in the next frame.

There is one representation for each chord type where each model consists of a probability density function for each pitch class relative to the root note of the chord. This was done under the basis that the profile for each chord type would look the same regardless of the root note.

There are 60 possible chords available for the model to differentiate via 12 possible root note and five chord types; major, minor, augmented, diminished and suspended. Although only triads were capable of being recognised, the frame-by-frame accuracy of the simulation done in Python was reported at 70%. More information regarding how the Gaussians were modelled using supervised learning can be read in (MYER, Sam, 2011).

## 2.4 Related Work

Two major commercial hardware products currently available, display similar attributes to the planned system framework presented in this report. These are “Harmony Singer” (TC-HELICON, 2013) and the “VoiceTone H1”, both produced by Canadian company TC-Helicon. The advertisements boast among other things, guitar-controlled vocal harmony identical to the system proposed in this report. The difference is in implementation as they are designed as guitar stomp box pedal effects. The implementation in software allows more scope for the inclusion of extra features.

One other implementation similar to the system presented here has been carried out using MATLAB in this paper (KNESEBECK, A. et al., 2011) but with some key differences. First, the calculation of the chromagram and pattern matching methods shown in the paper differ from the planned implementation of this project’s real-time system. Nevertheless, they have provided audio examples on their website (KNESEBECK, Adrian von dem, 2011), which will be used in later chapters to aid the evaluation of the framework presented in this report.

# 3. REQUIREMENTS

Certain restrictions had to be enforced in order to make the system design realisable. A concise requirement specification is presented here stating the quality necessary for the final format of the plug-in.

## 3.1 Harmonizer properties

* Must be successfully implemented as a VST plug-in using the JUCE C++ libraries.
* Obtain an overall system latency of less than 50ms given a sample rate of 44.1 kHz.
* Accept two separate monophonic signals for the vocal and accompaniment respectively.
* Implement some form of pitch correction to be applied only on the vocal signal in order to improve listening experience.
* Generate a maximum of two pitch shifted voices to coincide with the original signal.
* The user should have options to choose the type of harmonization seen in similar hardware.
* Provide visual feedback regarding the detected pitch and chords and subsequent harmonies.

## 3.2 Pitch Detector

* Accuracy of pitch detection must be within +/- 0.5 Hz between 100 Hz and 880Hz (within the average vocal range) (JONES, C., 2011).
* Determine voiced and unvoiced segments of the vocal input.

## 3.3 Pitch Shifter

* The maximum pitch shift range should be +/- 1 octave as no pitch shifting method exists that can handle any greater variation without severe corruption of the signal.
* Minimise audible sound artefacts in the pitch-shifted signals by any means necessary.
* Pitch shifting algorithm should maintain the formants of the input signal, i.e. maintain the original peaks in the spectral envelope of the pitch shifted sound.
* Provide user choice between a formant preserved and formant modified pitch-shifted signal.

## 3.4 Chord Detector

* Detect chords in a frame-by-frame manner correctly with at least 70% accuracy.
* Provide some form of parameters that can influence the recognition of a chord in real-time.

## 3.5 Audio input expectations

* The lowest fundamental frequency of an accepted voice is approximately 85Hz(equivalent E2 in MIDI keyboard scale)
* The highest fundamental frequency allowed from a voice will be approximately 1 kHz. These limits in the vocal frequency are based on a regular male and female vocal range
* The microphone used is of a satisfactory quality and positioned so that the singer’s voice is dominant, preventing other external sources interfering with the pitch tracker section of the harmonizer algorithm.
* To minimise crosstalk from the vocal, miking the accompanying instrument is not recommended.
* The vocal must not only be monophonic, but a solo voice (as opposed to a unison section).
* The accompanying instrument must be polyphonic and capable of playing chords.

# 4. DESIGN

The complexity of the problem is divided into four distinct areas, pitch detection, pitch shifting, chord detection and harmony synthesis. Figure 7 displays the final framework shown in block form highlighting the most significant algorithms and features within the system. The following sections discuss the steps that led to the final framework design.

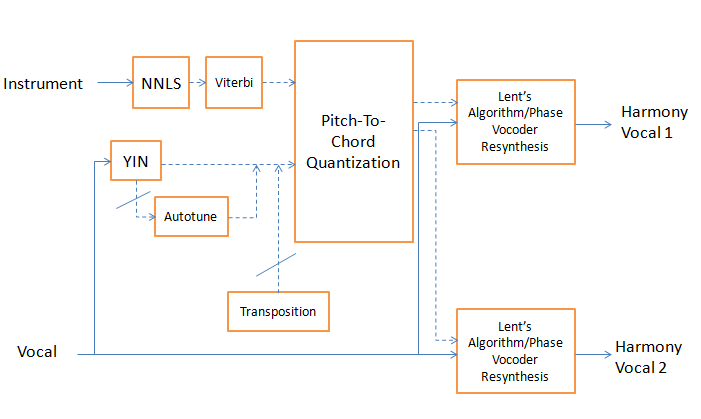


Figure 7 - Block diagram of framework. The crossed lines indicate that the section can be bypassed.

## 4.1 Pitch Detection

With the YIN algorithm as the pitch detector of choice, finding ways to maximise efficiency in each scheme was the prime target. For example autocorrelation can be achieved in the frequency domain where its equivalent to squaring the signal’s magnitude thus representing the power spectrum. It becomes an order N x log (N) operation when using the short time fast fourier transform (STFFT). To minimise spectral leakage during the transform, a Hann window was the chosen window function for this and all subsequent algorithms that involved entering the frequency domain.

The window size directly affects the smallest detectable fundamental frequency and is expected to be twice that particular frequency to properly correlate the signal waveform (MCLEOD, P., 2008). That same frequency would also affect the system input latency of the plug-in as lower frequencies have longer wavelengths. Based on the average vocal range (JONES, C., 2011) and the increased FFT efficiency when the size of the array is a power of two, a size of 1024 (210 ) samples was chosen. Hence for a sample rate of 44.1 kHz, the lowest detectable frequency with a window size of 1024 samples would be:

Equation 9 - Lowest detectable frequency

Therefore the minimum input latency of the plug-in with a sample rate of 44.1 kHz is:

Equation 10 - Minimum latency calculation

This acceptable compromise upholds part of the requirements of the specification.

Limiting the frequency range for the pitch detection via a low pass filter was important. By adhering to the general vocal range, this would reduce the computation time and improve the precision of the detected pitch estimates.

After analysing source code from an implementation of the YIN algorithm written in C (HORGAN, M., 2008), methods to optimise and enhance the various error correction schemes were developed. Adding the ability to discriminate between voiced and unvoiced frames was based on using the global minimum value to act as the upper threshold leaving a small region. Figure 8 better demonstrates the concept. In this arbitrary example, the absolute threshold (AT) is set to 0.1 and the global minimum to 0.5. Any dip in the normalised difference function signal that is below the absolute threshold is accepted as voiced signal. If the next frame of audio does not meet the AT but passes the global minimum threshold, it too is considered a voiced signal.

Furthermore, in order to reduce the possibility of sub periodic errors as well as minimise processing time, once a candidate dip reaches below the AT, any further calculations will be terminated and that value will be used as the basis for quadratic interpolation.

One caveat with the human voice is that the fundamental frequency is not always the most dominant harmonic. This is evident in Figure 9 where the second harmonic of the audio signal is similar in amplitude to the first. Because of this, sub periodic errors are still likely to occur within the YIN algorithm. For the best local estimate scheme within the YIN algorithm, the post process function within the open source dywapitchtrack library (SCHMITT, Antoine, 2010) was used. The function utilises methods that try to account for the possibility where pitch anomalies such octaves errors emerge by comparing the current frame with previous pitch estimates.

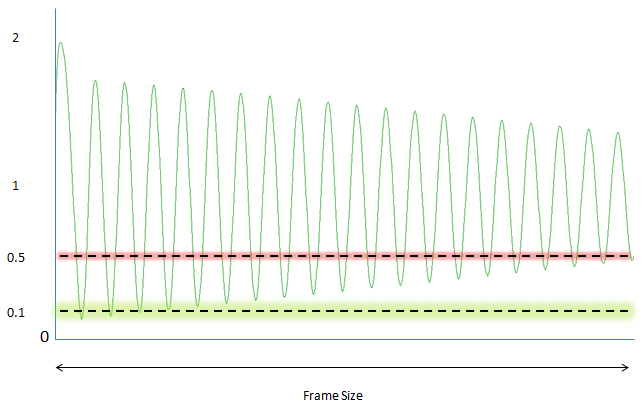


Figure 8 - Voiced/Unvoiced Transition where the green line represents the absolute threshold and the red line represents the global minimum threshold

### 4.1.1 Autotune

The inclusion of pitch correction for the vocal input was a necessary move as a means to improve the subjective quality of the audio output. Pitch correction works by quantizing a detected pitch to the nearest semitone. Commercial applications extend the quantization possibilities by including the feature of selecting a particular key and scale to improve the correction.

Once the f0 of an audio frame has been found, the next stage involves resynthesizing the audio signal to a new f0 according to the musical rules (key and scale) determined through user input. The most logical way to start the design was to apply the frequency to MIDI formula equation with no rounding or modification. Any sung vocal note will have some deviance to the actual targeted pitch and so will more often than not fall between two MIDI note intervals. Hence, the aim is to shift the semi-MIDI note representation to the MIDI note that is closest to the detected frequency given the key and scale.

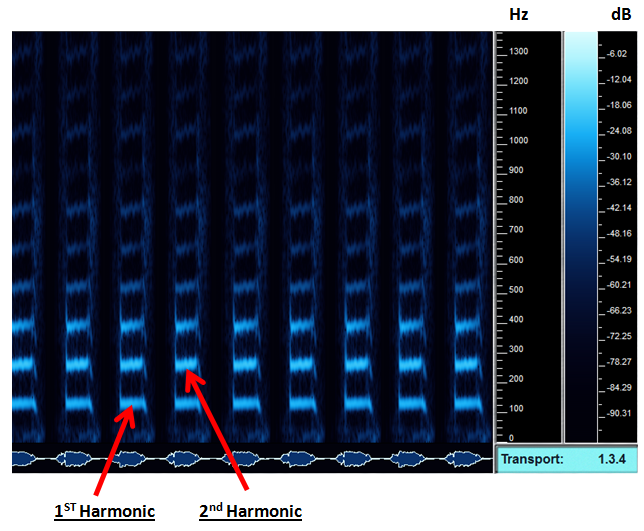


Figure 9 – Spectrogram of the author singing the word “C” with a pitch of “C2”

## 4.2 Pitch Shifting

Regarding the use of the Lent’s algorithm, it was mentioned in his paper that the zero crossing method of pitch detection was used within the pitch tracker component. The first modification made was to utilise the YIN algorithm to replace the previous method. The detected frequency within an audio frame is very likely to contain a non-integer number of periods. The pitch shifter algorithm functions with entire periods and so the output frame will need a larger amount of memory allocated to compensate. Other modifications include incorporating linear interpolation when storing the output frame to improve the output resolution. Using a fixed frame length of equal size to the pitch detection frame simplifies the management of input and output pointers..

Providing users the option to choose between a formant maintained and formant altered pitch shifter was important to extend the system’s creative capabilities. The decision was made to utilise Bernsee’s pitch shifting open source code from his online website (BERNSEE, S., 1999). The article explains in good detail, the steps involved to change the perceived pitch of an audio signal via a modified phase vocoder. The STFFT is used to represent an audio signal as a sum of sinusoids and hence apply pitch shifting by scaling the frequency of those sinusoids. The method does not preserve formants as desired and can be modified by optimizing the code to improve efficiency.

Furthermore, a transpose parameter for users to adjust was planned. With a range of two octaves, this gives the user the possibility of having pitch shifted vocals generated up to an octave above or below the vocal’s detected pitch at any time. This type of octave harmonization is typical in barbershop quartet and choral harmonies.

## 4.3 Chord Detection

The initial proposed plan was to utilise Myer’s open source code (MYER, Sam, 2013) as the base algorithm for the chord detection component. This is summarised in the following steps.

* Perform FFT on a Hann windowed frame of audio data
* Locate peaks in the frequency spectrum
* Compute mid and bass chromagrams from peaks.
* Determine the chord for each frame, using a DBN via a Viterbi decoder

However, Myer states in his report that the chord detection algorithm suffers from errors based on the overtones affecting the chroma values using the current chromagram extraction method. This issue is more prevalent when the supposed chord is played as an inversion as the note energy cannot be easily traced back to the original pitch class as mentioned in previous chapters (MAUCH, Matthias, 2010). This can result in chord ambiguity where the overtone interference causes one chord to be wrongly selected above another.

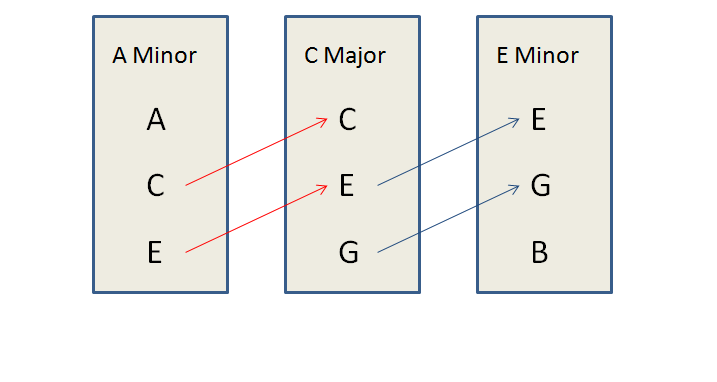


Figure 10 - Example of chord ambiguities

The goal of using the NNLS chroma method to approximate the note transcription of the audio signal will minimise this particular type of error when extracting chroma values.

Once the mid and bass chromagram values for an audio frame are calculated, the next step involves transcribing triads via a dynamic Bayesian network controlled by a causal implementation of the Viterbi algorithm.

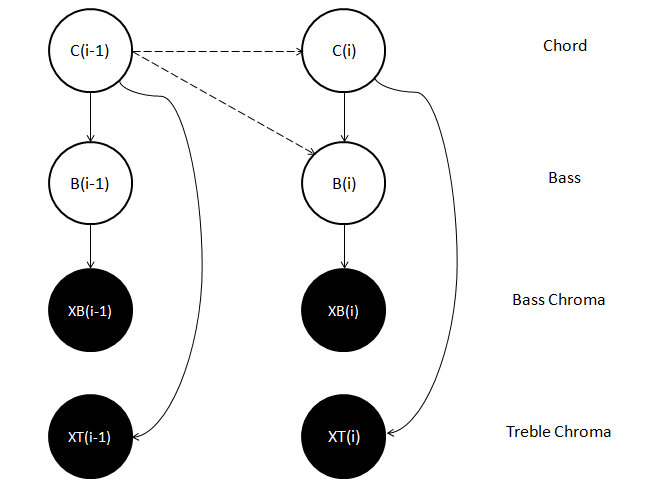


Figure 11 - DBN of the chord detection code from Myer

It is important to better identify the variables that make up the viterbi decoder component and a few of the modifications made to the chord recognition code by (MYER, Sam, 2013) intended for use.

The 60 possible chord models available are classified as a finite set of hidden states . The chroma features are the observable result of that process where there is one state per observation/audio frame.

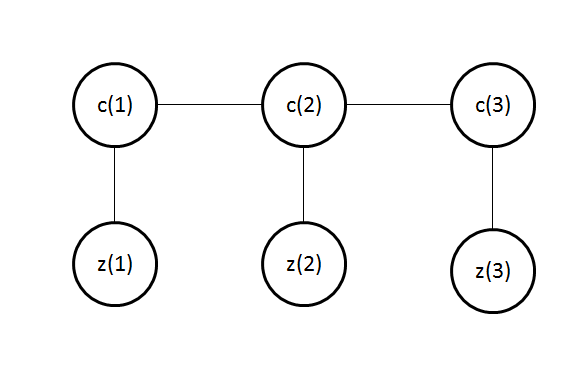


Figure 12 – For every observation per audio frame, there is one chord.

The hidden states and observations are related through the emission probability. This is the probability of observing a chroma vector at a frame given the chord . The next chord event only depends on the current chord to maintain the Markov properties. The goal henceforth is to find the most likely chord sequence that results on the current chromagram by using the Viterbi algorithm. The following parameters are identified:

* States () = A finite set of J chords which is 60 from the original algorithm.
* Observations ( the chromagrams for each audio frame
* Initial (prior) probability (π) is not the same value for the five chord types. A modified weighting that consists of the following: 0.4 for major, 0.5 for minor, 0.05 for diminished, 0.01 for augmented and 0.04 for suspended. In the original algorithm, the decision to incorporate a chord type weighting was based on fact that certain chord types (major, minor) are more prevalent in uncomplicated contemporary music.
* The transition probability between states is set to a modified default value of 0.6 where the probability that a chord change happens for a particular frame is 0.4. This is to serve as a starting point similar to (MYER, Sam, 2011).
* Probability of given - the emission probability is the 60x1 vector that contains all the probabilities for the chord models from the chromagrams using a bass probability lookup table and Gaussian pdfs identical to the original source code.

Once the transition matrix, the emission matrix and chord state probabilities from the previous frame have been computed, the product of the probabilities is calculated. For each chord in the current frame, the transition probability that maximizes its value is stored. The chord that has the highest probability path is sent to the output.

Equation – Finding the maximum probability

One further modification is the inclusion of the no chord frame model (based on amplitude) to account for when there is no audio input.

## 4.4 Harmony Synthesis

The aim in this block is to use the musical information from the chord detected to determine how many intervals to shift the vocal. It is expected there will be instances where the wrong chord is detected which could have a negative impact on the consonance of the overall musical output.

The setup will see the current detected pitch of the vocal quantized according to the notes of the current chord detected. This is considered as pitch correction of the singing voice with respect to the nearest chord note. Because of the probabilistic model used to determine chords, a deterministic approach is used to represent the detected chord. Regardless of the how the detected chord is played (root position, 1st,2nd inversion...etc), the representation will default to the root position of the detected chord.

As Lent’s pitch shifter processes periods of a waveform, the pitch shifting factors for the backing vocals is computed as the ratio between the expected shifted vocal pitch and the original vocal pitch where

Equation 12 – Shifting Factor for Voice 1

Equation 13 – Shifting factor for voice 2

Assuming the transpose parameter is set as default (no prior ratio to shift vocal), there are still various positions, where the synthesised backing vocal can be placed with respect to the original melody line. Giving the user the possibility to choose different arrangements of the two pitch shifted voices in accordance with the detected chord is a feature found on TC Helicon’s Harmony Singer and should be incorporated in this system.

The example presented and shown in figures 13 and 14 is based on the harmonic arrangement where the pitch-shifted voices are a 3rd and 5th above the original vocal. The perceived chord controls the type of harmony generated at all times. If a C major triad is detected, and the pitch detector senses the vocal singing C4 in this instance, the shifting ratio will be based on the chord notes within the triad nearest to the original vocal given the harmony arrangement. Therefore, for this case, the vocal will be pitch-shifted to E4 and G4 respectively.

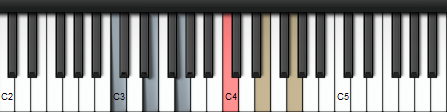


Figure 13 – Harmony example 1. The accompaniment (Blue) is playing a C Major triad. The vocal (Red) is singing a C4. The pitch-shifted voices (Beige) are E4 and G4 respectively

In the next example, a C major triad is detected and the vocal is singing A4. The vocal is not singing a note within the chord tones, but the same rules apply. Therefore, the vocal will be pitch-shifted to the nearest notes within the same harmonic arrangement, which happens to be C5 and E5 respectively.

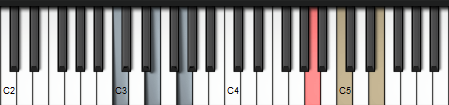


Figure 14 - Harmony example 2. The accompaniment (Blue) is playing a C Major triad. The vocal (Red) is singing an A4. The pitch-shifted voices (Beige) are C5 and E5 respectively

# 5. IMPLEMENTATION & TESTING

The following section describes details regarding the systems implementation as a VST Plug-in. Along with testing, evaluation results from some musicians will be discussed briefly as well.

When using external algorithms, they were modified where possible to improve functionality and efficiency and adapted to conform to the C++ class structure for easier usage in other applications.

Unless otherwise stated, this procedure was carried out when any significant block of code was written:

1. Debug the effect at runtime using a digital audio workstation (DAW).

2. Load the plug-in as an insert on a stereo audio channel track.

3. Following the signal chain, insert measurement plug-ins i.e. spectrum analyzer, spectrogram, test tone generator

4. Place breakpoints at points of interest in the code

5. Import mono/stereo audio tracks for analysis

6. Observe changes in parameters variables on a frame-by-frame basis and ensure methods are working correctly.

7. Fix any immediate issues or faults encountered during debugging and repeat until satisfied.

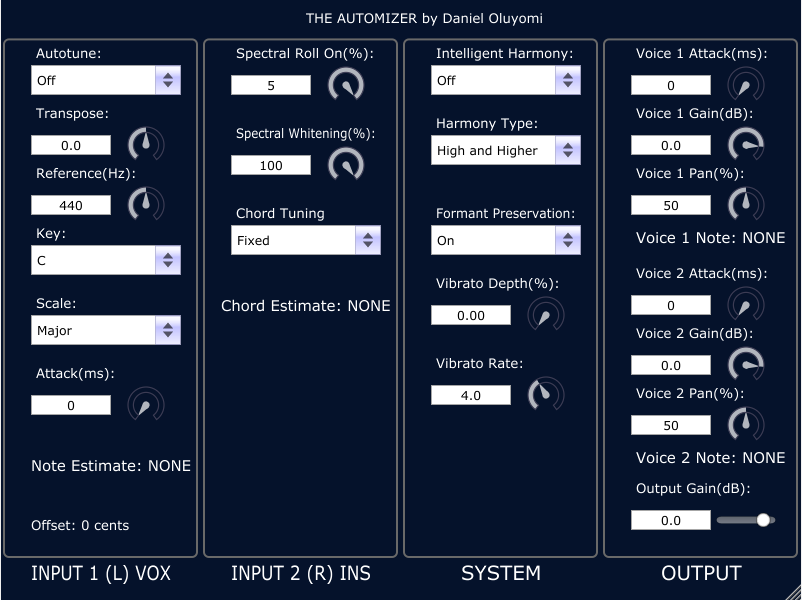


Figure 15 -The final GUI of the Automizer

The name of the VST Plug-in was called “The Automizer” referring to the project title.

## 5.1 Pitch Detection

In order to increase the frequency resolution as well as the efficiency of the pitch detection method, the audio signal is low pass filtered using an FIR filter function from (ORFANIDIS, Sophocles J., 2013) and then down sampled by a factor of four before the function is called. Given a sample rate of 44.1 kHz, the sample rate of the method is 11025 Hz, which is sufficient to find pitches up to 1 kHz.

To improve the detection and reduce the impact of partials affecting the detected pitch, a biquad low-pass infinite impulse response filter (IIR) was applied to the input audio frame prior to the frequency transform. Source code from (REDMON, Nigel, 2012) was used as the filter class.

As mentioned previously, the most efficient calculation of the ACF involves utilizing the FFT of an audio frame. The “Fastest Fourier Transform of the West” (FFTW) (JOHNSON, S and Frigo, M, 2012) library was implemented to meet any requirement where a frequency transform was necessary.

To reduce computation time during periods with no audio, a simple noise gate was implemented comparing the maximum magnitude to an arbitrary threshold while in the frequency domain. If the threshold was not met, the function exits the pitch detection class and returns a zero value that also represents no pitch was detected, bypassing the pitch shifter and pitch correction processes.

The implementation of the difference and CMNDF functions were adapted from the source code presented in (HORGAN, M., 2008). The absolute threshold is implemented by first finding a minimum within the normalised signal. Then a check for whether the minimum is below the absolute threshold or the global minimum. If the absolute threshold is reached, the function ends to save any further computation time. The minimum value therefore symbolizes the amount of aperiodic power accepted within a period signal. This value is used to determine whether the frame is voice or unvoiced. Once a frame is considered voiced, a quadratic interpolation function taken from (SIX, Joren, 2013) is applied to the minimum value. The post processing function from the dywapitchtrack library (SCHMITT, Antoine, 2010) was used as a replacement for the 6th scheme within the YIN algorithm to further reduce the chance of octave errors.

A few separate tests were formed to confirm the robustness of the YIN algorithm. The first test involved testing the accuracy of the pitch detector using simple signals. Then a short male vocal melodic phrase used previously in (KNESEBECK, A. et al., 2011) consisting of the words “Free Bird”, was assessed. The results are shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actual Hz | Waveforms | | | |
| Mean Detected Hz (10 frames approx 200ms) with sample rate = 44.1 kHz | | | |
| Sine Wave | Triangle Wave | Sawtooth Wave | Square Wave |
| 110Hz | 110.7 | 110.6 | 110.2 | 110.3 |
| 220Hz | 220.3 | 220.3 | 220.2 | 220.3 |
| 330Hz | 330.7 | 330.7 | 330.5 | 330.5 |
| 440Hz | 440.6 | 440.8 | 440.7 | 440.9 |
| 550Hz | 550.3 | 550.4 | 550.5 | 550.6 |
| 660Hz | 661.2 | 661.3 | 661.6 | 661.3 |
| 770Hz | 771.6 | 772.0 | 774.1 | 771.8 |
| 880Hz | 880.9 | 881.0 | 881.3 | 881.4 |

Table 4 - Accuracy of the implemented YIN algorithm

The accuracy is within +/- 1 Hz in the lower frequencies but slowly increases to +/- 2Hz above 660Hz. When testing the vocal phrase, the word “Bird” contains a plosive “B” and is evidently seen in figure 16 as a narrow peak at the beginning of the word. Removing the dywapitchtrack function however, revealed an octave error during the frames where the plosive part of the word “B” was active.

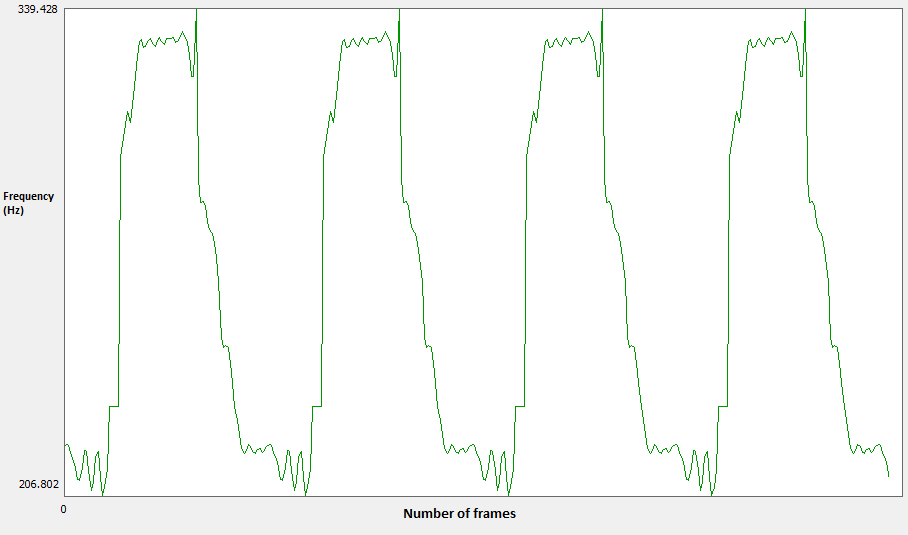


Figure 16 – Detected frequency at each frame for the vocal the phrase “Free Bird”

For the pitch correction feature, an implementation of a similar audio effect from (HUANG, Y., 2009) was used as a foundation to form the pitch correction process. The implementation was heavily modified to suit the pitch decision-making requirements mentioned in the previous chapter. These are the most common scales the user can choose: Major, Harmonic Minor, Pentatonic and Chromatic. The function to convert a pitch value into a MIDI note is called and the note interval closest to the current MIDI note given the key and scale, becomes the new MIDI note and hence the new fundamental frequency. The factor between the old and the new fundamental frequency becomes the shift factor that is used for the pitch shifter process.

Incorporating an attack parameter was important to allow for smoother transition between notes. Figures 17 and 18 show the effect of pitch correction on the “Free Bird” vocal phrase with no attack and an attack of 30ms respectively.

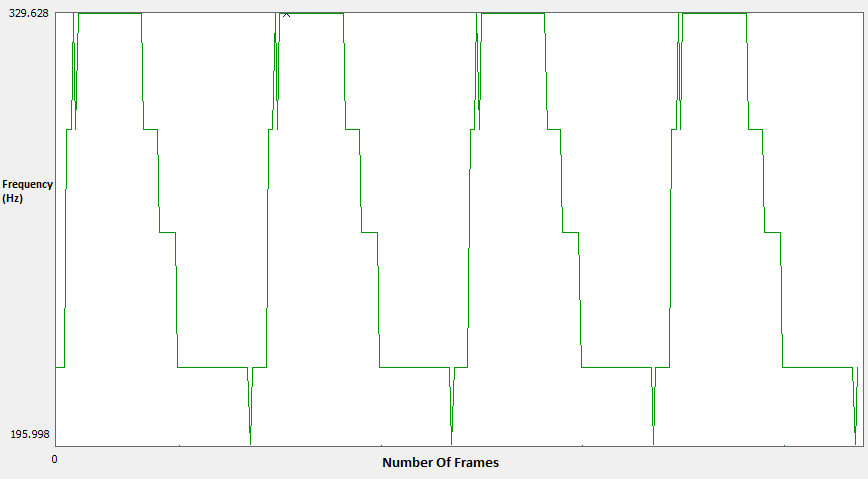


Figure 17 - Autotune with no attack

## 5.2 Pitch Shifting

The final period ratio to shift the input audio comes from the results of the pitch correction, transposition and pitch-to chord quantization processes. The inclusion of the pitch-shifting algorithm written in (BERNSEE, S., 1999) was straightforward. Replacing the default FFT function with the FFTW library implementation proved effective in reducing the CPU usage of the software plug-in by over 50%.

The implementation of Lent’s algorithm established in (GERMAIN, F., 2009), was used as the foundation for the second pitch shifting procedure. Once the ratio is found, a Hann window twice the length of the current detected period is calculated at each frame. In the original algorithm, the expansion/compression decision block of the algorithm requires time scaling based on the decision to allow the user to have variable input and output arrays. Because of the use of the YIN algorithm, the input and output buffers are fixed in size.

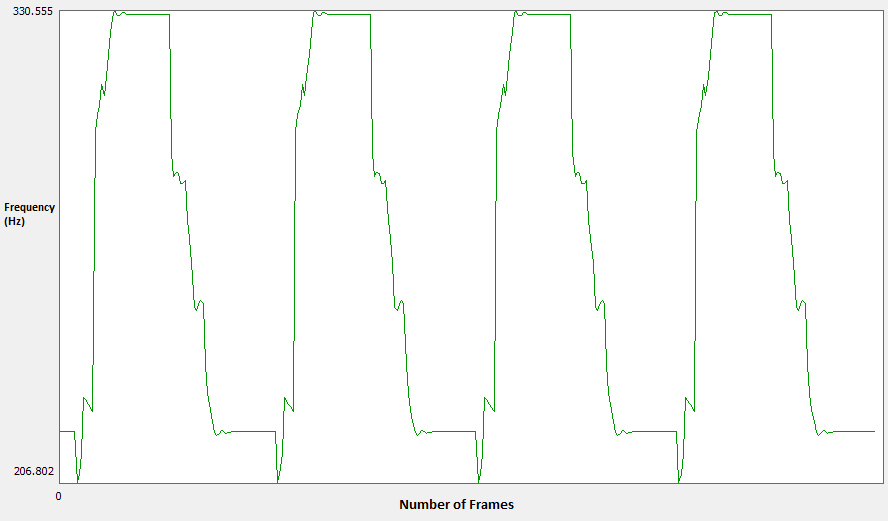


Figure 18 - Autotune with attack of 30ms

Equation 14 – Compression/expansion decision from Lent’s algorithm

Because Lent’s algorithm modifies the output samples retrospectively, the output reading has to be delayed with a delay line. The output queue is initialised to three times the length of the input buffer to avoid reading an output value before it was completely calculated. The most significant modification made to the source code was switching the direction of the read and write pointers that set the overlapping Hann windows. The original code suffered from audio discontinuities due to writing the new pitched signal sample values in a First In, Last Out scheme where the last sample of the pitch-shifted signal was stored in the output delay line first.

The transpose parameter effectively adjusts the pitch of the input frame prior to the pitch to chord quantization code block. Because of its position within the framework, it is possible to generate semi-intelligent harmonies based on the selected key and scale. Setting the transposition amount to a point in between intervals is the important factor. To shift up a third, a transpose setting of 3.5 semitones will have the input pitched note snap to a major third (4 semitones) or minor third (3 semitones) depending on the scale selected. Although it is not the aim of the project to produce semi-intelligent harmonization, it is still a possible implementation that perceptive users could take advantage of in combination with the automatic pitch to chord quantization of the harmonized voices.

When shifting a sinusoid with a non-integer period, it is shown that although the fundamental frequency is correctly shifted, additional partials are introduced to the signal. Figure 18 in particular shows the second harmonic frequency with an intensity almost identical to the fundamental frequency when shifting the pitch seven semitones below the original frequency of 440Hz.

In order to confirm that the formants of a vocal were retained, a spectrogram of the “Free Bird” phrase was analysed and compared against pitch-shifted versions of the same phrase. The phrase was shifted up and down by an octave (12 semitones) and it can be seen in appendix D that the figures shown verify the difference between the two different pitch shifting algorithms regarding the formants.

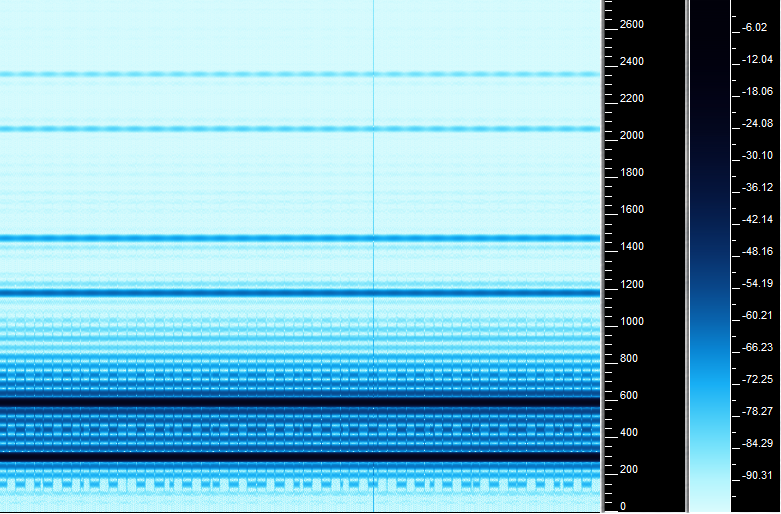


Figure 19 - Pitch shift ratio of 7 semitones downwards from 440Hz (expansion of 50%)

A soak test was applied by pitch shifting a sine wave a fifth above and below A4 440 Hz and leaving the plug-in running for 1 hour respectively. This test was done to uncover any faults concerning noise accumulation, memory leaks, DC offset and any long-term issues not noticeable upon initial execution. Based on the results shown in appendix D, the plug-in passed the soak test with no discernible difference between spectrum snapshots.

The lack of a sophisticated transient detector does affect the quality of the output. An attempt was made to use overlapping Hann windows during the transition between voiced and unvoiced frames as a means to cross fade the original and shifted vocal input during transients. This scheme slightly reduced the distortion during transition for the Lent implementation but increased the amount of audio discontinues with Bernsee’s pitch shifter.

## 5.3 Chord Detection

The initial chord detection implementation relied on converting the freely available Java source code of Chordroid (MYER, Sam, 2013) into the C++ framework of the system. Like the pitch detection process, block (frame) processing was necessary to accumulate enough samples to perform the FFT. With offline processing, the amount of accumulated samples within a block typically consists of a large value. For example, the Vamp plug-in of the NNLS Chroma uses 16394 samples per block. As the frame size represents the input latency of the plug-in, 1024 samples per frame for the FFT buffer was used as a compromise. This would affect the frequency resolution of the instrument signal in the frequency domain especially in lower frequency bands. To counteract this problem, the same method of LP filtering (to avoid aliasing) and x4 down sampling used for the pitch detection process was repeated for the chord detection. However, given the case, that sample rate is 11025 and the buffer size is 1024, the distance between two DFT bins turns out to be 10.77Hz . This is greater than the distance between F3 and F#3 (MIDI notes 53 and 54).The saving grace is that closely spaced bass notes are unlikely to occur in most contemporary music.

The NNLS Chroma operation from Mauch’s Vamp plug-in was implemented to obtain the normal and bass chromagrams. The significant difference between a Vamp and a VST Plug-in is that the former does not process in real time and can be non-causal. Furthermore, user parameters from the Vamp plug-in were only available for manipulation before processing an audio file. Vamp plug-ins however receive data via block processing, making it easier to convert the offline processes as real-time functions.

The spectral roll-on and whitening user parameters from the original Vamp plug-in were altered to allow for real-time user adjustment.

After the implementation of the Viterbi source code from Myer (MYER, Sam, 2013) , a simple ascending C major triad scale was performed to evaluate the current condition of the chord detection system. The algorithms inability to recognise when no chord was played required attention as during periods of silence, wildly random chord guesses were made when testing in Cubase. A modification in the form of a noise gate was employed. If the magnitude of a frame did not exceed “0.1”, the current and previous frame of information regarding the most probable path stored in the Viterbi decoder is reset. The value of 0.1 is a low magnitude and was chosen to ensure that if a chord is struck quietly, it is still capable of being detected.

An evaluation of the overall chord algorithm was done using three re-recorded interpretations of the following songs.

* Happy Birthday by Patty and Mildred Hill (HILL, Patty and Hill, Mildred, 1893)
* Suspicious Minds by Elvis Presley (PRESLEY, Elvis, 1968)
* Heartless by Kanye West (WEST, Kanye, 2009)

“Suspicious Minds” and “Heartless” were re-recorded with an arrangement of only a sampled piano from Steinberg’s Halion 4 (STEINBERG, 2013) software virtual instrument and a vocal sang in the style of the original track. “Happy Birthday” uses a guitar only arrangement obtained from (ANDREAS, Jamie, 1999) with vocals added. These songs were chosen based on their familiarity with the general public and genre (STEADMAN, Ian, 2013). Assessing the quality of the chord detection has to be compared to a ground truth transcription done by a human expert. 40-second versions of the interpretation of these songs were manually annotated and assessed by using the relative correct overlap (RCO) method. It measures the proportion of the duration of a musical piece whereby the chord detector output is identical to the ground truth annotation. Instrumentals of the re-recorded arrangements of the three songs were used to with the RCO method.

Equation 15- RCO equation

The results for each song are listed in Table 5. Appendix E contains all the chord annotations for each song as well as the chord annotations from the chord detector output. The MIREX mapping standard only recognises “major”, “minor” and “no chord” labels (MIREX, 2009). This was accounted for by setting the chord type probabilities for augmented, diminished and suspended chords to zero with major and minor chord types reset to 0.5 each.

|  |  |  |
| --- | --- | --- |
| Happy Birthday | Suspicious Minds | Heartless |
| RCO = 82.27% | RCO = 88.04% | RCO = 93.93% |

Table 5 - RCO results

## 5.4 Harmony Synthesis

Implementing the pitch to note quantization method first entailed setting up a multidimensional array consisting of two octaves worth of chord triad intervals for each available chord type. The current detected note of an audio frame, the detected chord and chord type as well as the user-selected harmony were input parameters for the harmony decision process. The detected pitch note was converted into a MIDI note representation and depending on the detected chord, the pitch was quantised to the nearest note interval of the triad chord.

The different interval arrangements of the two pitch shifted voices in accordance with the detected chord were set as follows:

* **High and Higher –** Voice 1 = 3rd Above, Voice 2 = 5th Above
* **Low and Lower –** Voice 1 = 4th Below, Voice 2 =6th Below
* **Higher and Lower –** Voice 1 = 5th Above, Voice 2 =6th Below
* **High and Low -** Voice 1 = 3rd Above, Voice 2 =4th Below

The reasoning behind the choice of particular arrangements was based on how the Harmony Singer TC –Helicon attempts to harmonize the backing vocals. Giving the user choice over the type of harmonies they desire is important due to the subjective nature of how people perceive “good” or “bad” harmonies.

In an attempt to humanise the pitch-shifted voices, an implementation of the vibrato effect from (HEATH, Mark, 2011) was applied. Vibrato refers to small, quasi-periodic variations in the pitchof a tone. In the singing voice, vibrato is produced by modulating the tension of the vocal folds for a sung voice. To apply vibrato to the pitch shifted voices, it is necessary to apply a quasi-periodic frequency shift by using a modulated delay line. A low frequency oscillator controls the variation/rate of the frequency shift and a parameter for the amplitude of the oscillator represents the depth. An attack parameter similar to the one seen to adjust the pitch correction speed, for each voice was included to minimise artefacts. The pitch-shifted voice starts from the input vocal note given the value of the transpose parameter and reaches the nearest chord tone according to the type of harmony selected.

A survey was conducted to identify what people consider to be a “good” backing vocal harmony within the context of a song. The songs used to evaluate the chord detection process were utilized. The general familiarity of the songs within the public would make it easier for someone to make the decision whether the harmonies heard at a particular point in time satisfied their expectations. The exercise required the listener to concentrate on two things during playback of the audio:

1. The harmonies generated.
2. The quality of the voices.

A lead sheet was constructed for each song with blank boxes at the location of a change in chord. The listener was required to tick or cross the boxes at these points whether they liked or disliked the harmony generated from that time instant to the proceeding box. The test also involved changing the pitch shifter process from Lent’s algorithm to Bernsee’s pitch shifter. After listening to the audio up to four times, a short questionnaire regarding each song asked the following questions:

* Have you previously at any point in time heard, sang or performed this song prior to this survey?
* How satisfied are you with the harmonies generated over the course of the song?
* Rate the tonal quality of the voices heard during the 1st three times the song was played.
* Rate the tonal quality of the voices heard during the final time the song was played.

Listeners had to decide via a rating system from 1-5 where “1” was designated “very dissatisfied/bad” and “5” was considered “very satisfied/excellent”. A sample of 10 people including the author took the test. The figures for the survey results are shown below.

Certain stipulations were set during the test procedure. The greater the shift, the more inaccuracies appear for the pitch shifting methods so to keep the shifted voices within a small range, only the “High and Low” harmony was tested as this represented a shifted voice a 3rd above and 4th below the vocal. The attack parameters of the voices were set to 0ms (inactive), the vibrato depth was set to zero and the pitch correction function was turned off. The pan settings for voice 1 and 2 was set to 0 %( L) and 100% (R) respectively. The test involved listening via headphones so having the voices separated in the stereo field would allow listeners to hear the pitch-shifted voices with more clarity.

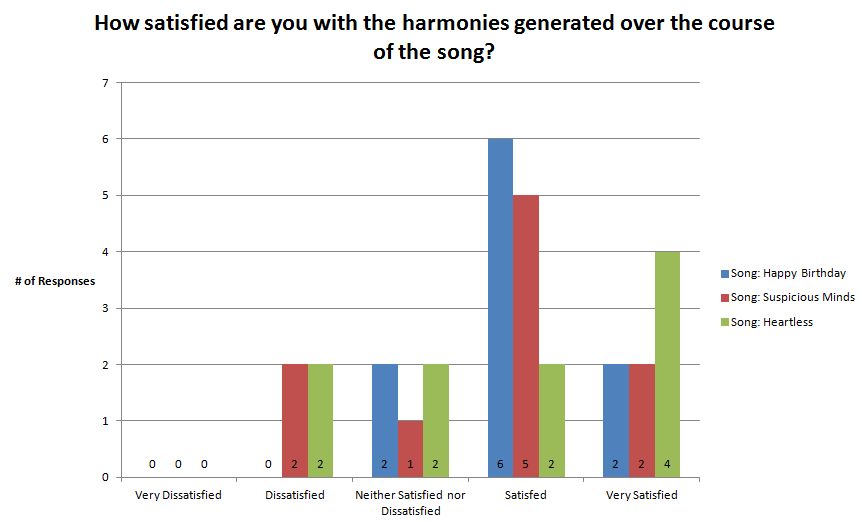


Figure - Survey Results 1

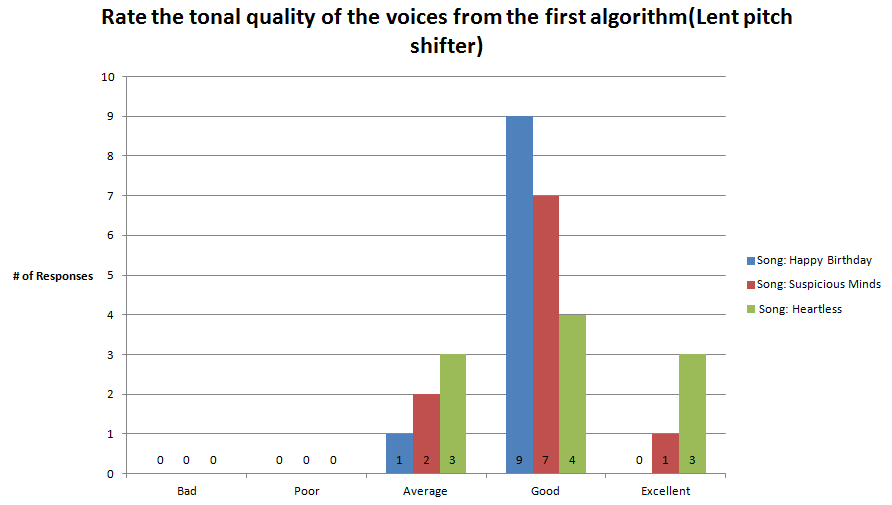


Figure - Survey Results 2

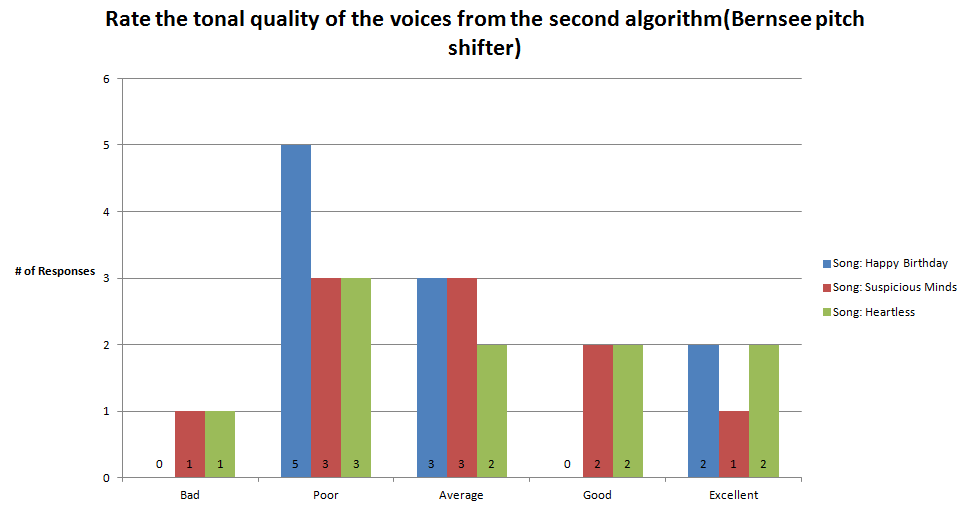


Figure - Survey Results 3

## 5.5 Latency

The VST Plug-in Analyser by Christian Budde (BUDDE, Christian, 2009) is an all-purpose application that analyses various aspects of a VST plug-in such as its frequency response, harmonic distortion and for the purposes of this project, the overall processing latency. The results from the application showed an estimated delay of 3072 samples. With a sample rate of 44.1 kHz the overall latency of the system is:

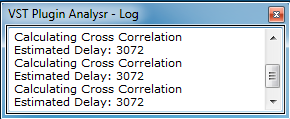


Figure - Estimated Latency Value

To ensure the validity of the result, a manual method was set up using Cubase. One cycle of a 440Hz sinusoid was recorded to represent a ping. The track that contained the sinusoid was sent to a new track where the “Automizer” plug-in was inserted in between. The difference measured between the recorded ping through the plug-in and the original ping would be the overall latency. Cubase and other DAWs apply automatic plug-in delay compensation to all software plug-ins, which involves delaying all active plug-ins by the highest reported plug-in latency. The reported latency for the plug-in was initially set to “1024”. As a result, for this manual method, Cubase will already account for 1024 samples. Therefore, the total amount of latency is the measured difference + 1024. Judging from the test, an additional 2048 sample delay was observed, confirming the accuracy of the result from the VST Plugin Analyser.

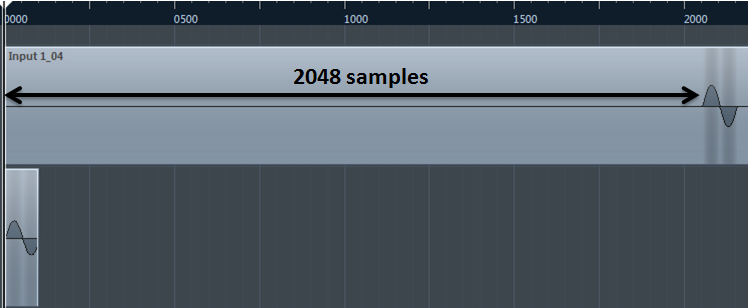


Figure - Results from manual measurement for plug-in latency

The requirement for the system latency of 50 milliseconds was not achieved. However, with further optimization on the source code, it should be possible to reduce the latency to meet the requirement.

## 5.6 Evaluations

With all the modifications applied to the pitch detector, it is fairly robust with the ability to handle pitch correction, voiced/unvoiced segments and plosive with satisfactory accuracy. The main weaknesses concerning the pitch detection methods include its inability to detect frequencies below F2 for the case of the fixed input frame length of 512 samples at a sample rate of 44.1 kHz. Transient material could be handled better by the addition of a transient detector to optimise pitch detection processing.

The pitch shifter processes based from the results of the survey show a clear preference towards the Lent pitch shifter. This is mostly likely due to its formant preserving properties that allow it to mimic the nuances of the human voice when pitch shifted far better than Bernsee’s pitch shifter. In general, the organization of pitch shifting class component within the system makes it very possible to import enhanced pitch shifting algorithms for future use.

The RCO results from the chord detection show very promising signs. The main issue is the fact that testing only three songs is not enough to boast the claim of achieving above 90% RCO accuracy. With more time permitted, several more songs would have put the chord detector to the test.

The results from the survey overall indicate that the harmonies generated by the software plug-in are satisfactory given the harmony setting. Prior to starting the survey, it was the author’s opinion that the large majority would prefer the Lent pitch shifter implementation over Bernsee’s. Comparing figures 21 and 22 prove that the former is indeed preferred. Given its wider distribution of responses regarding its quality, it was still a surprise to see some consider Bernsee’s pitch shifter as “excellent”. These instances confirmed the decision to incorporate choice over the type of pitch shifting algorithm used.

Overall, the robustness of the current system framework does leave room for improvement. However, there is certainly commercial value for the overall framework once the inaccuracies with the chord detection have been minimised and the latency further reduced.

# 6. CONCLUSIONS AND FURTHER WORK

During this report, the topics of pitch detection, pitch mapping, pitch shifting and chord detection have been discussed in adequate detail to support the realisation of how the real time system for automatic vocal harmonization was developed. The real-time system’s intention was its usage in live scenarios for musicians to enhance a performance. This goal limited the scope of the algorithms available for use. Hence, the combination of the external open source algorithms used was critical to the progression made thus far on the software plug-in.

The overall algorithm functions in real time with a latency of approximately 69 milliseconds which still enables the use of the VST plug-in as a live effect. Given the size of the chord buffer, the outcome of the chord accuracy was much better than expected. However, the need for further testing using more songs and documenting how it handles against more complex chords is crucial. Nevertheless, the framework for real-time automatic vocal harmonization has been achieved successfully. The harmonization of the synthesised backing vocals has been accomplished based on the chords detected from an accompanying instrument.

## 6.1 Future Work

* Extending the chord detection to detect seventh chords will allow the formation of more complex harmonies.
* Reducing the current latency of the system is an area well worth investigating through the optimization the current source code.
* The need for a transient detector can improve the output of both the pitch detection and shifting components.
* Enhancing the GUI of the VST plug-in is a potential area of interest. This could be achieved by providing improved visual response regarding the changes in harmony and the pitch and chord detection. An example would to be to add a virtual keyboard indicating the current note and chord performed per frame.
* Implement preset saving functionality for the plug-in for easier recall of settings

# REFERENCES

ANGUS, J and D HOWARD. 2009. *Acoustics and Psychoacoustics*. Oxford: Focal Press.

BERNSEE, S. 1999. *Pitch Shifting Using The Fourier Transform*. [online]. [Accessed 31 August 2013]. Available from World Wide Web: <<http://www.dspdimension.com/admin/pitch-shifting-using-the-ft/>>

BUDDE, Christian. 2009. *VST Plugin Analyser*. [online]. [Accessed 30 August 2013]. Available from World Wide Web: <<http://www.savioursofsoul.de/Christian/programs/measurement-programs/>>

CANNAM, C., C. LANDONE, and M. SANDLER. 2010. Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files. *In*: *ACM Multimedia*. Firenze.

CENTRE FOR DIGITAL MUSIC, QUEEN MARY, UNIVERSITY OF LONDON. 2010. *Sonic Visualiser*. [online]. [Accessed 2 August 2013]. Available from World Wide Web: <<http://www.sonicvisualiser.org/>>

CHO, Taemin, Ron J. WEISS, and Juan P. BELLO. 2010. EXPLORING COMMON VARIATIONS IN STATE OF THE ART CHORD RECOGNITION SYSTEMS. *In*: *Sound and Music Computing Conference (SMC)*. Barcelona, Spain, pp.1-8.

COSTELLO, Sean. 2009. *Auto-Tune, autocorrelation and seismic analysis*. [online]. [Accessed 31 August 2013]. Available from World Wide Web: <<http://cnx.org/content/m33242/latest/>>

DIXON, Simon. 2013. *Pitch Transcription*. [online]. [Accessed 12th April 2013]. Available from World Wide Web: <<http://qmplus.qmul.ac.uk/course/view.php?id=2420>>

GERBER, S and R. SCOTT. 1972. Pitch-Synchronous time-compression of speech. *In*: *Proc. Conf. Speech. Communications*., pp.63-65.

GERHARD, D. 2003. *Pitch Extraction and Fundamental Frequency: History and Current Techniques*. Saskatchewan.

GERMAIN, F. 2009. *LentPitShift.h*. [online]. [Accessed 29 August 2013]. Available from World Wide Web: <<https://ccrma.stanford.edu/software/stk/LentPitShift_8h_source.html>>

HANSON, R. J. and C. L. LAWSON. 1974. *In*: *Solving Least Squares Problems*, Prentice-Hall, p.chapter 23.

HILL, Patty and Mildred HILL. 1893. *Happy Birthday*. Warner/Chappell Music.

WEST, Kanye. 2009. *Heartless*. Chicago: Def Jam.

HEATH, Mark. 2011.*.NET Voice Recorder*. [online]. [Accessed 25 August 2013]. Available from World Wide Web: <[http://voicerecorder.codeplex.com/SourceControl/latest](http://voicerecorder.codeplex.com/SourceControl/latest%20) >

HORGAN, M. 2008. *Automatic Accompaniment for Electric Guitar*. [online]. [Accessed 31 August 2013]. Available from World Wide Web: <<https://ccrma.stanford.edu/~mhorgan/220c/>>

HORVATH, Michael. 2008. *Music frequency diatonic scale*. [online]. [Accessed 2 August 2013]. Available from World Wide Web: <<http://en.wikipedia.org/wiki/File:Music_frequency_diatonic_scale-3.svg>>

HUANG, Y. 2009. *voice2midi*. [online]. [Accessed 31 August 2013]. Available from World Wide Web: <<http://code.google.com/p/voice2midi/>>

JOHNSON, R. B. 1995. A Detailed Analysis of a Time-Domain Formant-Corrected Pitch-Shifting Algorithm. *Journal of the Audio Engineering Society*. **43**(5), pp.340-352.

JOHNSON, S and M FRIGO. 2012. *FFTW*. [online]. [Accessed 16 August 2013]. Available from World Wide Web: <<http://www.fftw.org/>>

JONES, C. 2011. *Range*. [online]. [Accessed 16 August 2013]. Available from World Wide Web: <<http://cnx.org/content/m12381/latest/>>

KAWAHARA, Hideki and Alain de CHEVEIGNÉ. 2002. YIN, a fundamental frequency estimator for speech and music. *Acoustical Society of America*. **111**(4).

KLAPURI, Anssi and DAVY, Manuel (eds). 2006. Introduction to Music Transcription. *In*: *Signal Processing Methods for Music Transcription*, Springer US, pp.3-20.

KNESEBECK, Adrian von dem. 2011. *Department of Signal Processing and Communications: Harmonization Examples*. [online]. [Accessed 15th August 2013]. Available from World Wide Web: <<http://www.hsu-hh.de/ant/index_Xy1gsz5MM71VDx4a.html>>

KNESEBECK, A., S. KRAFT, and U. ZÖLZER. 2011. A system for synthesis of backing vocals by intelligent harmonization. *In*: *Digital Signal Processing (DSP), 2011 17th International Conference*., pp.1-6.

KNESEBECK, A. von dem and U ZÖLZER. 2010. Comparison of pitch trackers for real-time guitar effects. *In*: *Proceedings of the 13th Int. Conference on Digital Audio Effects (DAFx-10)*. Hamburg, pp.266–269.

LENT, Keith. 1989. An Efficient Method for Pitch Shifting Digitally Sampled Sounds. *Computer Music Journal*. **13**(4), pp.65-71.

MADDOX, R and Eric LARSON. 2005. *Real-Time Time-Domain Pitch Tracking Using Wavelets*. Urbana.

MAUCH, Matthias. 2010. *Automatic Chord Transcription from Audio Using Computation Models of Musical Context*. [online]. [Accessed 30th March 2013]. Available from World Wide Web: <<http://matthiasmauch.de/_pdf/matthiasmthesis.pdf>>

MAUCH, Matthias. 2010. *mauch ISMIR2010 slides*. [online]. [Accessed 31 August 2013]. Available from World Wide Web: <<http://www.youtube.com/watch?v=gxCbcQk3r68>>

MAUCH, Matthias and Simon DIXON. 2010. Approximate note transcription for the improved identification of difficult chords. *In*: *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR*. Utrecht, pp.135-140.

MAUCH, M and S. DIXON. 2010. MIREX 2010: Chord detection using a dynamic bayesian network. *In*: *In Proceedings of Music Information Retrieval Evaluation eXchange (MIREX)*.

MAUCH, Matthias and Simon DIXON. 2010. *Simultaneous Estimation of Chords and Musical Context from Audio*. London.

MCLEOD, P. 2008. *Fast, Accurate Pitch Detection Tools for Music Analysis*. Dunedin.

MIREX. 2012. *MIREX2012 Results*. [online]. [Accessed 12 August 2013]. Available from World Wide Web: <<http://www.music-ir.org/mirex/wiki/2012:MIREX2012_Results>>

MONZO, Joe. 2001. *A Gentle Introduction to the MIDI Tuning Specification*. [online]. [Accessed 12 August 2003]. Available from World Wide Web: <<http://www.tonalsoft.com/monzo/miditune/miditune.aspx>>

MYER, Sam. 2011. *Real-time Chord Detection*. London.

MYER, Sam. 2013. *Chordroid*. [online]. [Accessed 20 August 2013]. Available from World Wide Web: <<https://github.com/sammyer/Chordroid>>

NI, Yizhao, Matt MCVICAR, Raúl SANTOS-RODRÍGUEZ, and Tijl De BIE. 2012. An End-to-End Machine Learning System for Harmonic Analysis of Music. *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING*. **20**(6), pp.1771-1783.

ORFANIDIS, Sophocles J. 2013. *Lab 3*. [online]. [Accessed 23 August 2013]. Available from World Wide Web: <<http://eceweb1.rutgers.edu/~orfanidi/ece348/lab3.pdf>>

REDMON, Nigel. 2012. *Biquad C++ source code*. [online]. [Accessed 23 August 2013]. Available from World Wide Web: <<http://www.earlevel.com/main/2012/11/26/biquad-c-source-code/>>

SAVAGE, M. 2010. *How commonplace is auto-tune?* [online]. [Accessed 2 August 2013]. Available from World Wide Web: <<http://www.bbc.co.uk/news/entertainment-arts-11056840>>

SCHMITT, Antoine. 2010. *dywapitchtrack*. [online]. [Accessed 23 August 2013]. Available from World Wide Web: <<http://www.schmittmachine.com/dywapitchtrack.html>>

SHEPARD, R. 1999. Pitch perception and measurement. *In*: P. COOK, (ed). *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*, MIT Press, pp.149-165.

SIX, Joren. 2013. *FastYin.java*. [online]. [Accessed 23 August 2013]. Available from World Wide Web: <<https://github.com/JorenSix/TarsosDSP/blob/master/TarsosDSP/src/main/java/be/hogent/tarsos/dsp/pitch/FastYin.java>>

STARK, Adam and Mark PLUMBLEY. 2009. Real-Time Chord Recognition for Live Performance. *In*: *International Computer Music Conference Proceedings*. Montreal.

STEADMAN, Ian. 2013. *Filmmakers claim 'Happy Birthday' is public domain, sue Warner for royalties*. [online]. [Accessed 1 September 2013]. Available from World Wide Web: <<http://www.wired.co.uk/news/archive/2013-06/14/happy-birthday-lawsuit>>

STEINBERG MEDIA TECHNOLOGIES GMBH. 2011. *3rd Party Developer*. [online]. [Accessed 13 August 2013]. Available from World Wide Web: <<http://www.steinberg.net/en/company/developer.html>>

PRESLEY, Elvis. 1968. *Suspicious Minds*. Las Vegas: RCA.

T.CHO, R. WEISS, and J. BELLO. 2009. *Exploring Common Variations in State of The Art Chord Recognition Systems*. [online]. [Accessed 30th March 2013]. Available from World Wide Web: <<http://www.ee.columbia.edu/~ronw/pubs/smc2010-chordreco.pdf>>

TC-HELICON. 2013. *Harmony Singer*. [online]. [Accessed 2 August 2013]. Available from World Wide Web: <<http://www.tc-helicon.com/products/harmony-singer/>>

THOMAS, J. M. 2012. *ROBUST REALTIME POLYPHONIC PITCH DETECTION*. Fairfax.

ZPLANE. 2013. *vielklang*. [online]. [Accessed 2 August 2013]. Available from World Wide Web: <<http://www.zplane.de/products/plugins/vielklang>>

# APPENDIX A – Note Frequencies under equal tuning

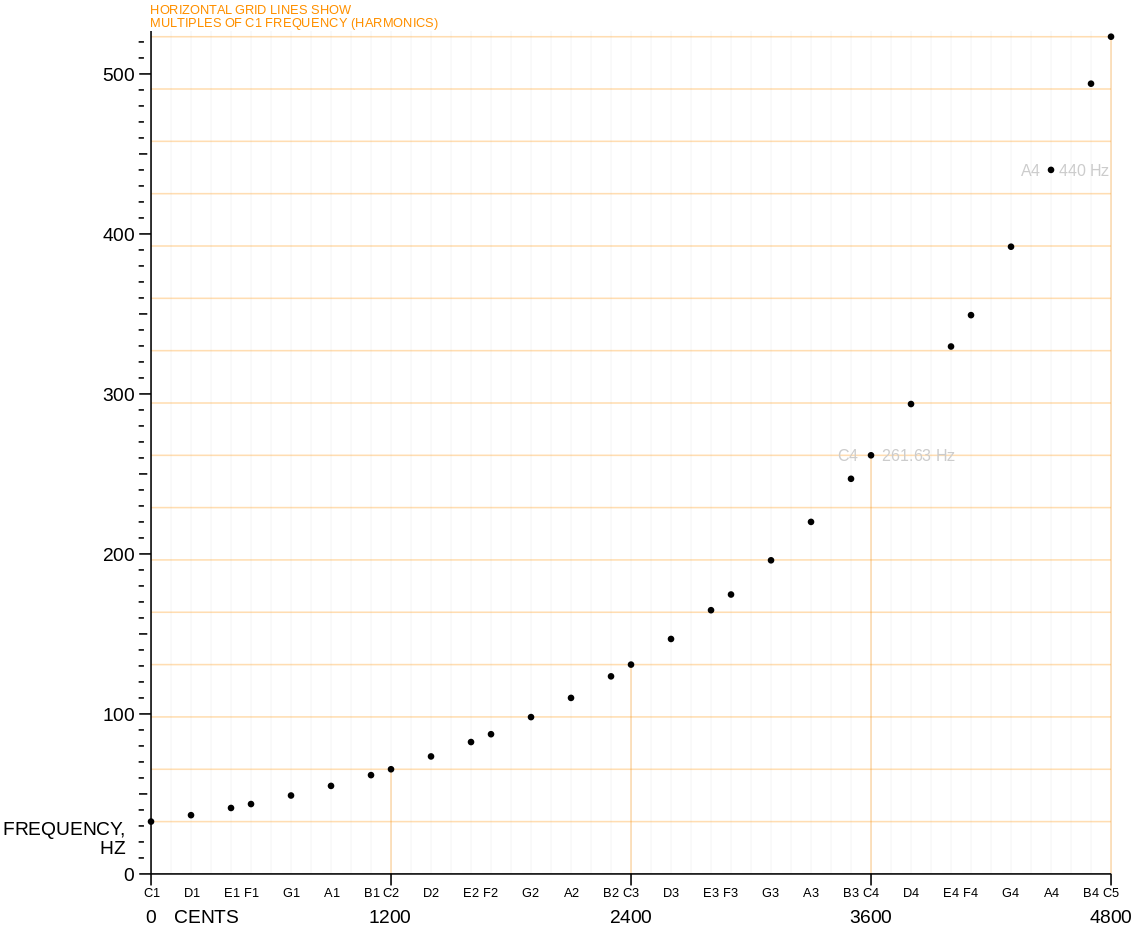


Figure 25 - Note frequencies under equal temperament tuning, four-octave C major scale, beginning with C1. (HORVATH, Michael, 2008)

# APPENDIX B - MIDI, Frequency, Pitch and Period Conversion Table

|  |  |  |  |
| --- | --- | --- | --- |
| **MIDI Note Number** (2 octaves) | **Frequency** (Hz) (1.d.p) | **Pitch** (Middle C = C4) | **Sample Period**  ( Fs = 44.1kHz) (1.d.p) |
| 54 | 185.0 | F#3 | 238.4 |
| 55 | 196.0 | G3 | 225.0 |
| 56 | 207.7 | G#3 | 220.4 |
| 57 | 220.0 | A3 | 200.5 |
| 58 | 233.1 | A#3 | 189.2 |
| 59 | 246.9 | B3 | 178.6 |
| 60 | 261.6 | C4 | 168.6 |
| 61 | 277.2 | C#4 | 159.1 |
| 62 | 293.7 | D4 | 150.2 |
| 63 | 311.1 | D#4 | 141.7 |
| 64 | 329.6 | E4 | 133.8 |
| 65 | 349.2 | F4 | 126.3 |
| 66 | 370.0 | F#4 | 119.2 |
| 67 | 392.0 | G4 | 112.5 |
| 68 | 415.3 | G#4 | 106.2 |
| 69 | 440.0 | A4 | 100.2 |
| 70 | 466.2 | A#4 | 94.6 |
| 71 | 493.9 | B4 | 89.3 |
| 72 | 523.3 | C5 | 84.3 |
| 73 | 554.4 | C#5 | 79.6 |
| 74 | 587.3 | D5 | 75.1 |
| 75 | 622.3 | D#5 | 70.9 |
| 76 | 659.3 | E5 | 66.9 |
| 77 | 698.5 | F5 | 63.1 |
| 78 | 740.0 | F#5 | 59.6 |

# APPENDIX C – Formants

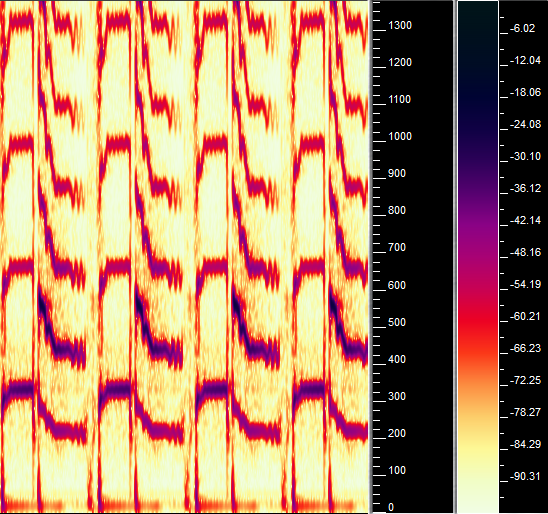


Figure - Spectrogram of original phrase repeated four times.

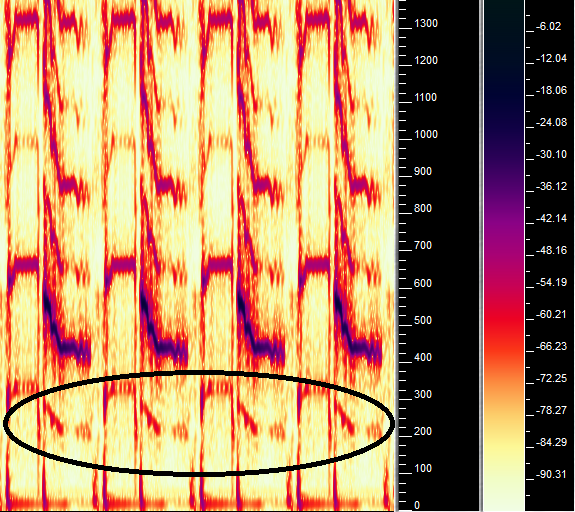


Figure - Lent pitch shifter. Formants are preserved as the vocal shape in the spectrogram is almost identical to the original

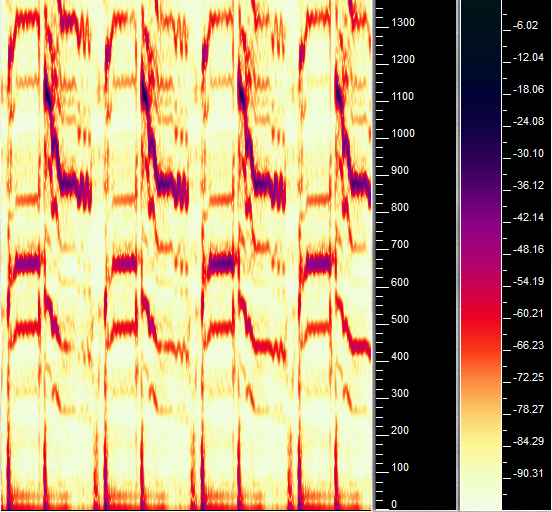


Figure - Bernsee pitch shifter. The formants of the vocal line have been clearly altered, no longer resembling the shape seen in figure 20

# APPENDIX D – Results of Soak Test

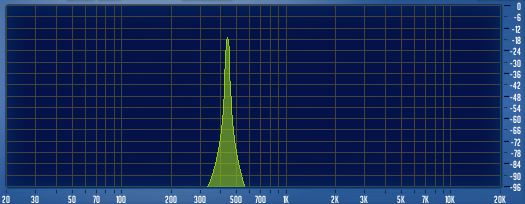


Figure - 440Hz Sine Wave

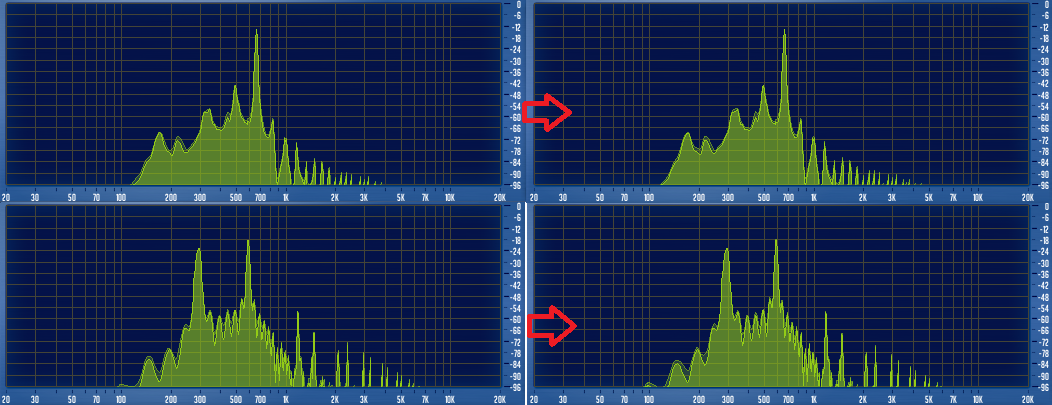


Figure - Soak Test Result. 1st Row: pitch shift up 7 semitones from 440Hz. 2nd Row, pitch shift down 7semitones from 440Hz.

# APPENDIX E – Chord Annotations

**Song: “Happy Birthday No Harmonization.wav”**

ONSET(s) OFFSET(s) CHORD LABEL

00.710 05.443 G

05.443 12.552 D

12.552 19.996 G

19.996 23.978 C

23.978 26.419 G

26.419 27.557 D

27.557 29.519 G

**Song: “Suspicious Minds No Harmonization.wav”**

ONSET(s) OFFSET(s) CHORD LABEL

01.400 05.602 G

05.602 09.814 C

09.814 11.984 D

11.984 14.090 C

14.090 15.176 D

15.176 16.185 C

16.185 17.203 Bm

17.203 18.251 D

18.251 20.404 C

20.404 22.512 G

22.512 24.637 Bm

24.637 25.671 C

25.671 26.751 D

26.751 28.876 Em

28.876 30.949 Bm

30.949 33.059 C

33.059 35.173 D

**Song: “Heartless Remake No Harmonization.wav”**

ONSET(s) OFFSET(s) CHORD LABEL

00.655 04.338 Bbm

04.338 07.075 Ab

07.075 09.805 F#

09.805 11.529 F

11.529 15.256 Bbm

15.256 17.975 Ab

17.975 20.691 F#

20.691 24.003 F

**Chord Detection Results from Song: “Happy Birthday No Harmonization.wav”**

ONSET(s) OFFSET(s) CHORD LABEL

00.000 00.114 NO CHORD

00.114 00.347 C

00.347 04.967 G

04.967 12.096 D

12.096 15.648 G

15.648 16.043 D

16.043 19.619 G

19.619 21.941 C

21.941 22.614 Em

22.614 23.315 G

23.315 23.520 C

23.520 23.915 D

23.915 25.958 G

25.958 27.119 D

27.119 27.398 Gm

27.398 29.519 G

**Chord Detection Results from Song: “Suspicious Minds No Harmonization.wav”**

ONSET(s) OFFSET(s) CHORD LABEL

00.000 1.415 NO CHORD

1.415 5.664 G

5.664 10.006 C

10.006 12.607 D

12.607 14.232 C

14.232 14.604 G

14.604 15.440 D

15.440 16.368 C

16.368 17.460 Bm

17.460 18.458 D

18.458 20.641 C

20.641 22.731 G

22.731 24.727 Bm

24.727 25.888 C

25.888 27.142 D

27.142 28.907 Em

28.907 29.232 D

29.232 31.043 Bm

31.043 33.737 C

33.737 35.173 D

**Chord Detection Results from Song: “Heartless Remake No Harmonization”**

ONSET(s) OFFSET(s) CHORD LABEL

0.679 4.500 Bbm

4.500 7.724 Ab

7.724 9.883 F#

9.883 11.687 F

11.687 15.326 Bbm

15.326 18.133 Ab

18.133 20.850 F#

20.850 24.003 F

1. where p is the pitch number, which ranges from zero to 127, representative of the MIDI standard pitch (MONZO, Joe, 2001). The constant 69 is the pitch number of A4 (the reference frequency). [↑](#footnote-ref-1)