Computational challenges in pitch detection algorithms

Abstract

Pitch detection refers to the problem of identifying pitches in audio signals. Its application is evident in many fields ranging from musical context such as tuning and automatic music transcription to studies in psychology related to prosody, word meaning and emotional content of human voice. Development of *pitch detection algorithms* (PDAs) is still a topic of intense study. This thesis elaborates on pitch detection from a computational perspective and lists different approaches and challenges faced by each of them. It may serve as an introductory guide to anyone aiming to work in pitch detection.

Introduction

Pitch is one of the core elements in music, responsible for forming melodies and harmonies. There is no strict definition for pitch perception as it is not yet fully understood but it is often described as the perceptual correlate of acoustic periodicity (Oxenham, 2012) or in other words audio frequency perceived by a trained human ear when listening to sound with periodic content [1]. Technically, this corresponds to the identification of the fundamental frequency of a signal. It has been demonstrated though, that human ears may perceive a pitch that corresponds to a fundamental frequency that is missing (Schouten, 1940) or masked by noise (Licklider 1954). In psychoacoustics this is termed as *"missing fundamental"* and although the exact mechanism of human pitch perception is yet unclear it is widely accepted nowadays that perceived pitch is largely affected by harmonic content. Harmonic content refers to the amplitude pattern of the frequencies that are multiple to the fundamental frequency[2]. However, computational estimation of the fundamental frequency of a signal is a topic of research. A good online demonstration of this problem can be found in [Web1] and for a brief description and ongoing research see (Zatorre, 2005).

A tentative workaround for the problem of missing fundamental is to examine all harmonics and infer the pitch by the energy in the partials (i.e. multiples of the fundamental frequency). Even if this approach proves to be efficient for pitch identification of single tones, the complexity of the

[1] Strictly speaking, there are exceptions where pitch is perceived even in absence of periodicity, such as noise filtered by comb or band-bass filters. This special case exhibits interesting properties but is not covered here thoroughly. However, more information can be found in (Heller, 2012; Chapter 23: "Pitch perception") and (Wishart 1994; Chapter 2: "Pitch")

[2] Harmonic content also defines timbre. Timbre is responsible for discriminating two instruments from each other (e.g. guitar and flute) when the same note is played. Even if the fundamental frequency is identical, the sound character is different because of the timbre.

problem increases significantly in case of polyphony, when the signal includes more than one tone. The problem is easier in cases where notes do not share harmonic content (e.g. C3 and A4). Harmonic content is also what defines the character of a sound or a musical instrument (*timbre*) and explains how humans can discriminate across different instruments even if they play the same note, that is, identical fundamental frequency. Strictly speaking, frequency content of higher order partials may still overlap, but in practice this happens in harmonics that are negligible for two reasons. First, natural instruments have a limited range of harmonics and second human hearing stops perceiving periodic sounds as pitches around 4 kHz (Oxenham, 2012). However, the complexity of the problem increases when two or more notes share harmonic content, which occurs often in music as harmonic ratios appear to be aesthetically pleasing and constitute the basis of harmony. Hence, for an undefined number of notes, the computational problem of pitch detection brings the secondary but equally important problem of identifying the number of notes that are present. In case of no notes, which corresponds to silence or noise, a computational model should still be consistent. The observations above summarize the problem of polyphony.

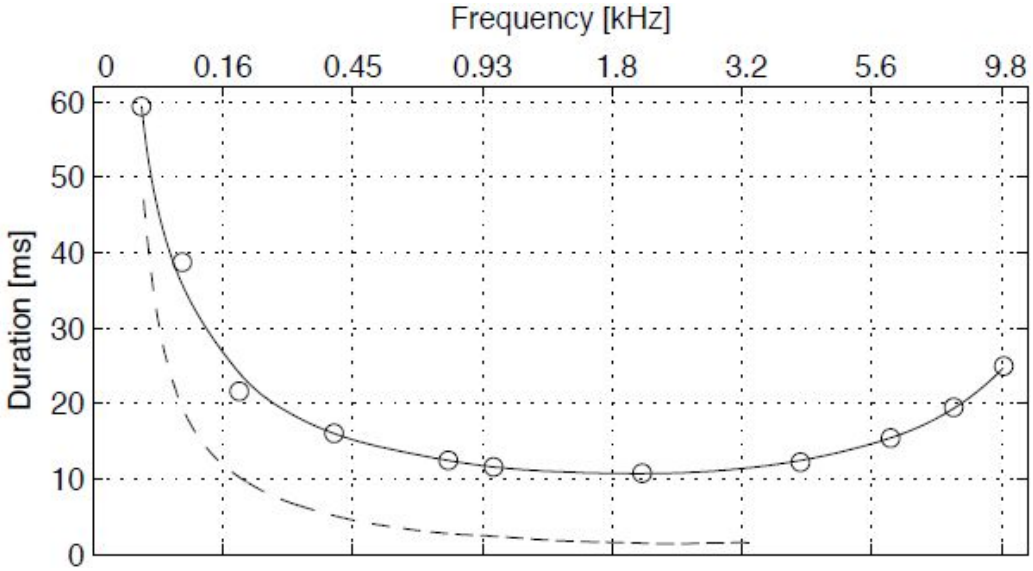

Figure 1. Solid line: Duration for human ear to perceive pitches according to Burck et al. (1935). Dashed line: According to Savart (1840) two cycles are required to detect pitch

Missing fundamental and polyphony are the main challenges of pitch detection. Even if a trained ear might require a certain amount of repetitions to identify the fundamental frequencies of all the notes played, all the information is set in a definite amount of time. The time required to define pitch has been under research and it varies among frequencies. Previous research has shown that a duration of 60 milliseconds is sufficient to perceive pitch even in low frequencies. (see **Figure 1**) and that different frequencies may require different durations to be perceived. This is in analogy with the loudness curves produced by Fletcher and Munson (1933) where perceived loudness is also frequency dependent (see **Figure 2**). From a computational perspective it is important to examine what signal duration is required to examine pitch content.

This may not necessarily coincide with durations required by human hearing and yet may vary between different approaches. Especially when referring to real-time pitch detection the analysis time durations should be negligible. But even if longer durations are required, pitch changes within the analysis interval should either be identified or, in the worst case, neglected so that they do not lead to misleading results.

Digital representation of sound consists of samples acquired at a certain sample rate. Computational approaches typically work on short time windows of the signal, called buffers. Depending on the application and the sample rate, the size of the buffer may vary from a few milliseconds to some tenths of a second. In a similar fashion as human hearing, different frequencies may require different times to be perceived by a computational model. This introduces a trade off between frequency and time resolution that is also demonstrated later in this thesis.

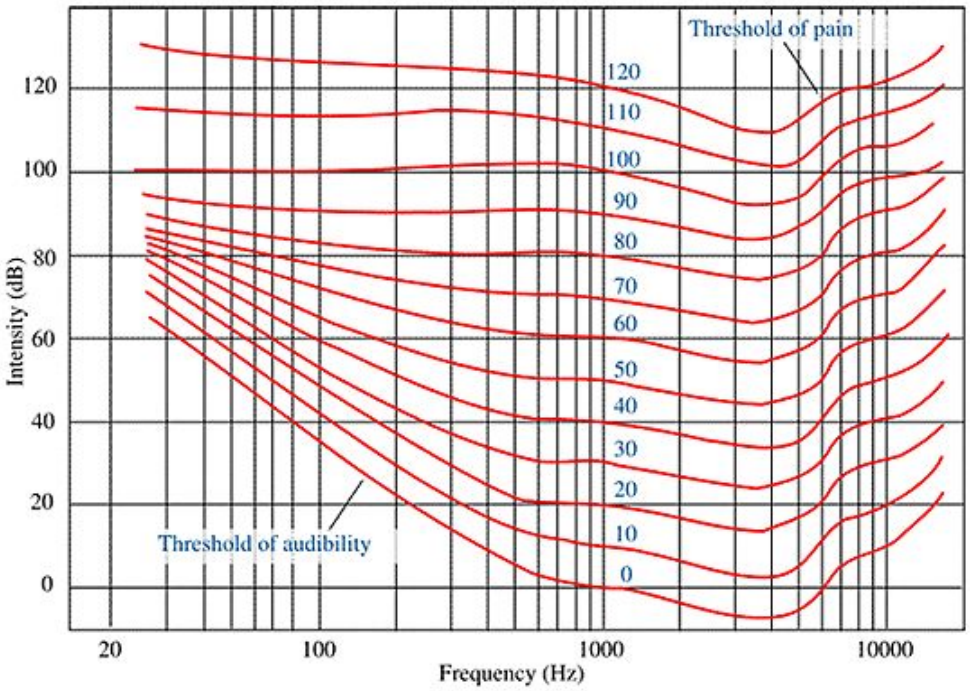

Figure 2: Fletcher-Munson curves show the relation of loudness perception and frequency (1933).

Applications of pitch detection include instrument tuning, chord identification and ultimately automatic music transcription (see Klapuri, 2004 for an extensive work on this topic). Ideally, music transcription involves transcription for each instrument separately, which in turn implies the distinction of multiple instruments. Such a problem is though beyond the scope of pitch detection as it could be considered as a separate processing step, employing techniques of source separation. Yet, the quality of the input signal fed to the pitch detection model may include non-musical content. Ideally, an efficient model should be able to ignore such content.

This thesis addresses the problem of pitch detection from a computational perspective. Here we present a variety of approaches for pitch detection listing their strengths and weaknesses.

Methods

In this section we cover a range of basic approaches for pitch detection. First we address methods in time domain, then we move to frequency domain approaches and last we discuss about machine learning methods. We apply the methods in a simple melody under different parameters and examine their strengths and weaknesses. The algorithms were implemented in MATLAB (MATLAB 2015b, The MathWorks, Inc., Natick, Massachusetts, United States).

Zero-crossing rate

The zero-crossing approach detects the time points where the signal crosses the zero value. It is a very simple and fast approach that works well when other frequencies are absent. However this is rarely the case. The zero-crossing rate can be simply estimated by multiplying a signal with a shifted version of itself by one sample. In time points where the product is negative, the signal crossed from a positive value to a negative (or vice versa) and thus the signal crossed zero. Since the method detects both changes, from positive to negative and negative to positive, it corresponds to half period of the signal. In **Figure 3**, an example is presented with an A4 played with virtual piano (SuperQuartet by Edirol) which corresponds to 440Hz.

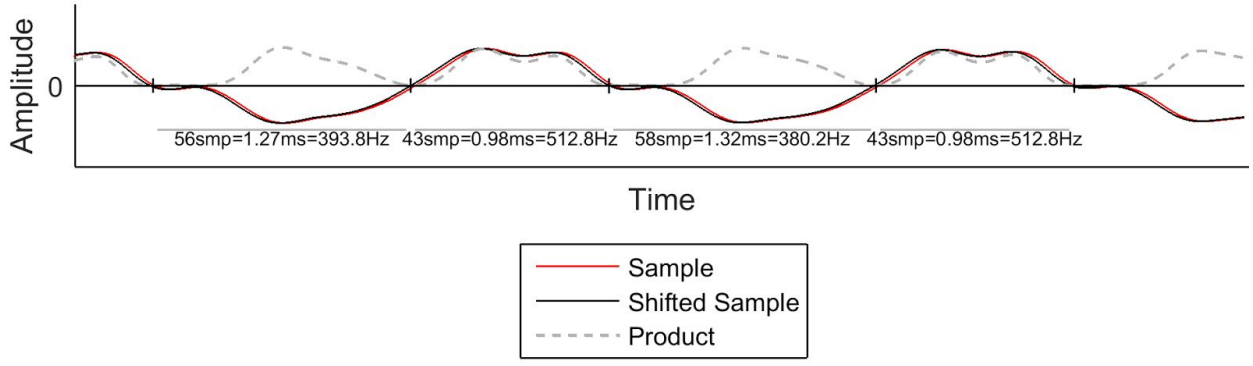

Figure 3: Zero-crossing point of an A4 piano sample. Original sample is shown in black, a shifted version by 1 sample is shown in red and their product with a dashed curve.

Zero-crossing points are detected at distances that correspond to half cycle of the detected frequency. Four zero crossing points were detected within the buffer in **Figure 3**. We observe that frequencies corresponding to positive parts (between a negative-to-positive crossing point and a positive-to-negative) crossing point are higher than the other ones. This is probably due to same offset triggered by some very low frequency component (or even a DC offset) and that is the main drawback of the zero-crossing technique. The average detected frequency in this example is 449.87Hz. Even if averaging produces efficient enough results, in case of polyphony such an averaging is not feasible.

Autocorrelation

Autocorrelation is intuitively explained as the correlation of a signal with a temporally shifted version of itself. Correlation may be any type of similarity measure between. Simple correlation can be formulated as following

$$r(\tau) = \sum_{t=0}^{n} x(t)x(t-\tau) \qquad \textbf{Eq. (1)}$$

Strictly speaking, the most popular similarity measure is Pearson correlation that relates to the equation above when the mean of the signals has been removed and they have been divided by their standard deviation, or in other words, when the signals are *normalized*. Since pitched sounds are periodic signals, their autocorrelation is high when the they are shifted for a number of samples that approximates their period. An example is shown in **Figure 4.**

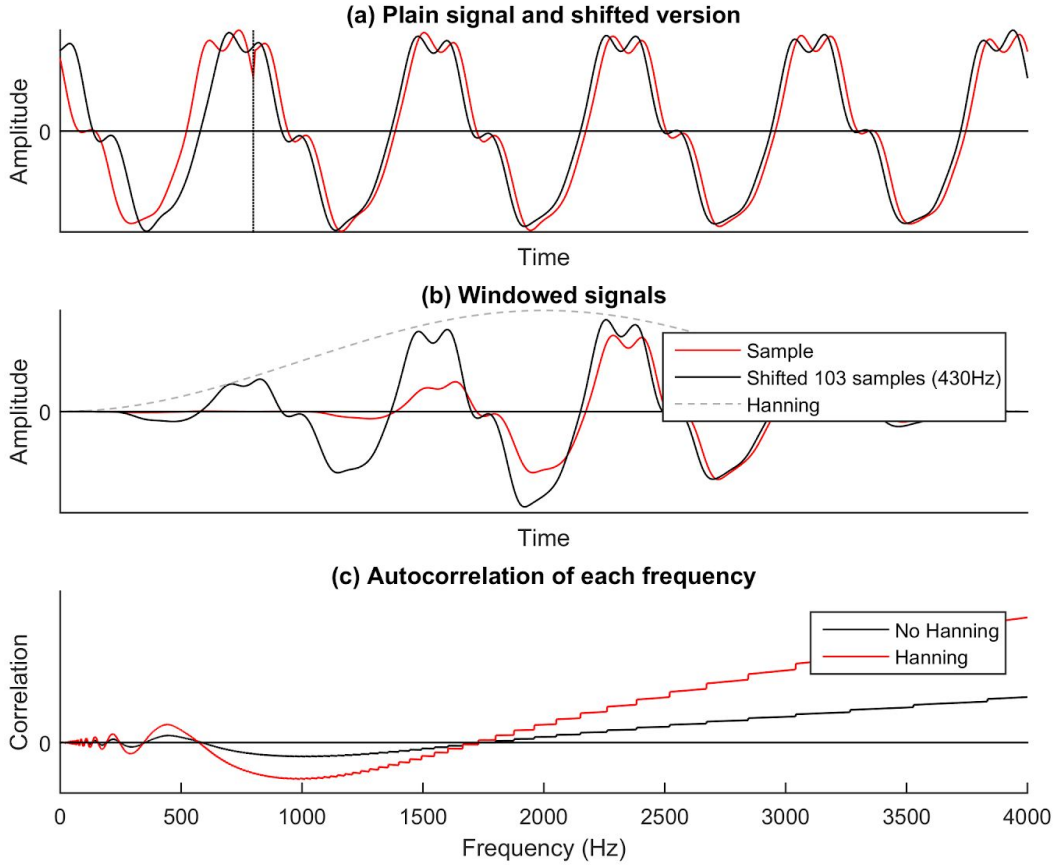

Figure 4: (a): A window of 512 samples from an A4 piano note and a shifted version by 103 samples. **(b)**: same as subfigure a but with a Hanning window applied in both signals, **(c)**: result when repeating the autocorrelation process for sample lengths that correspond to frequencies from 1 to 4000 Hz

There are a few observations that can be made by analyzing **Figure 4**. Subfigure **a** shows a part of a signal (A4 in piano) and its shifted version for 103 samples which corresponds approximately to 430 Hz (buffer size: 512 samples - Sampling frequency: 44.1kHz). In this case

shifting was applied within the same buffer (circular shifting) and this may introduce discontinuities (indicated by the vertical line in **Figure 4a**). In such cases, to avoid the effect introduced by the sharp windowing of the signal, smoother window functions have been proposed such as hanning window. The hanning window is shown in **Figure 4b** (dashed line) and element-wise multiplication has been applied between the two signals and the hanning window.

The correlation for the two signals in each case is shown in **Figure 4c**. The result exhibits a rippling behavior with the highest peak around 440 Hz, where the actual note is. Autocorrelation is often called quasi-periodic as it exhibits a periodic-like behavior (Huang et al., 2001; figure 6.32). Then, a continuous increase is observed as well a stepping effect. The continuous increase happens as increasing frequencies correspond to very low period, which in turn requires minor shifting of the signal. This leads to an effect like comparing the signal to itself, hence autocorrelation is then driven mainly by the lower frequency components of the signal even if there no higher frequencies involved. The stepping effect that is observed is an effect of the sampling frequency. For example, in a sampling frequency of 44.1kHz, frequency 440 Hz corresponds to 100.22 samples, 441 Hz to 100 samples and 442Hz to 99.77 samples. Since the number of samples for the shifting has to be an integer, the signal will be shifted for the same amount of samples for all of these frequencies, until frequency 444 which corresponds to a 99.32 samples, rounded to 99. This effect increases with the frequency upon study, but decreases when the sampling frequency is increased.

Autocorrelation can be problematic also in the low frequencies (i.e. long periods) so a slightly different equation that penalizes low frequencies is often used to avoid this problem (Huang et al., 2001; Equation 6.159 - “empirical correlation”).

Multiple autocorrelation

Real-time signal processing is typically applied in a buffer of predefined size. Simple autocorrelation approach performs circular shifting only once. However it is possible to perform circular shifting more than once and then correlation can be calculated in each pair of these smaller samples. It is expected that this approach is more robust since the existence of each frequency is tested a few times.

Frequency based approaches

Since PDAs aim to identify the fundamental frequency (or fundamental frequencies) of a melody, it is common to perform analysis in the frequency domain. This is typically done by transforming the signal in the frequency domain using Fourier Transform. The Fourier transform in frequency f for a continuous signal x is calculated as following:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-i2\pi f t} dt \quad \textbf{Eq. (2)}$$

This equation can be simplified using Euler’s formula that provides the relationship between complex exponential functions and trigonometric functions:

$$e^{-ix} = cosx - isinx \qquad \textbf{Eq. (3)}$$

Using Euler's formula the Fourier transform formula becomes:

$$X(f) = \int_{-\infty}^{+\infty} x(t)(cos(2\pi ft) - isin(2\pi ft)))dt = \int_{-\infty}^{+\infty} x(t)cos(2\pi ft)dt - i\int_{-\infty}^{+\infty} x(t)sin(2\pi ft))dt$$

The last equation shows that Fourier transform in practice measures the similarity of a signal to sine and cosine waves of a given frequency. The formulation shows that the frequency domain version is a complex number where the real part corresponds to the multiplication of the signal with a cosine wave of a frequency under analysis f and the imaginary part corresponds to the multiplication of the signal with a sine wave of the same frequency. Intuitively it is explained as the resonance of the signal in a given frequency.

The most popular algorithm for transforming a signal to frequency domain is the Fast Fourier Transform (FFT) due to its computational speed. However, its main drawback is that the analyzed frequency centers are dependent on the buffer size and the sampling frequency and linearly equally spaced. Therefore it does not provide direct information on specific frequencies but rather on frequency bins spaced as

$$\frac{Fs}{N},$$

where Fs is the sampling frequency and N is the size of the buffer, indicating that for the frequency resolution drops for higher sampling frequency and and for lower number of samples in the buffer.

Alternatively, the signal can be analyzed at desired frequencies by examining the signal's product with sine and cosine waves of these frequencies as mentioned above.

Fourier analysis

To examine if a signal contains a certain frequency (e.g. 440Hz) , a sine and a cosine wave can be generated at this frequency and then calculate the product of the two multiplications. If the signal contains a component of that frequency the product should be high, given that the buffer size is large enough to detect this resonance. Similarly, the frequency domain version of the signal can be estimated by testing multiple frequencies. This can be repeated, for example, for all the notes within a given range of notes (e.g. all the notes of the piano). An example of the procedure is presented in **Figure 5**.

A window of 512 samples is shown in (a) from a piano playing A4 (440Hz). A sine wave and a cosine wave (red and blue dashed lines respectively) of 440 Hz are overlaid. The cosine wave appears to be very similar to the actual signal. The sine wave appears to be out of phase. Sine waves and cosine waves are orthogonal signals. This means in practice that they are independent to each other and manage to catch complementary information. This is also demonstrated in **Figure 5**, where the cosine is synchronized better with the piano signal (**Figure 5a**) and hence carries more relevant information (**Figure 5b**). Orthogonality is mathematically expressed as

$$sin(x)cos(x)\,dx = 0$$

This practically means that if one of the two (sine or cosine) fails to catch periodicity because it is out of phase, then the other will catch that information. It is also important to note that in case of negative similarity where signals are similar but with an opposite sign, a negative high similarity which is turned positive when the power spectrum is calculated. The power spectrum is simply the root square of the squared values.

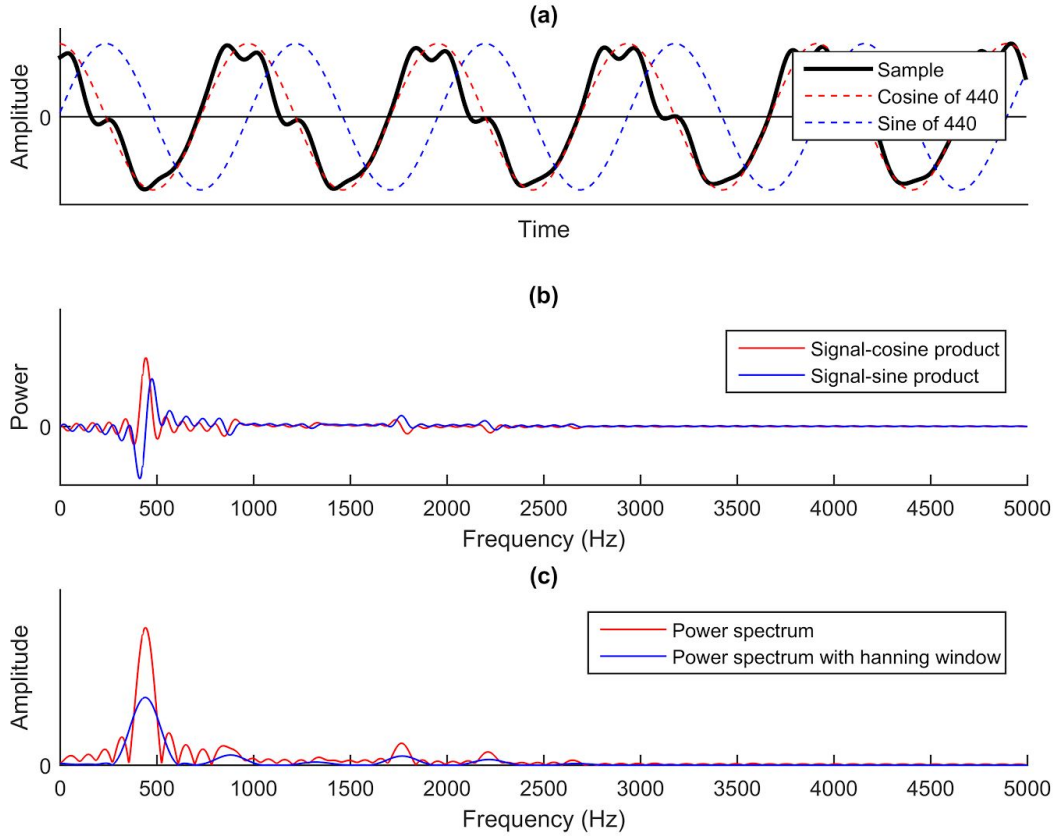

Figure 5: Fourier analysis for a window of 512 samples of an A4 piano. (a) the signal, a sine wave and a cosine wave of 440 Hz. (b): similarity of the sine wave and a cosine wave for each frequency from 1 to 5000 Hz. (c) Power spectrum for frequencies from 1 to 5000 Hz, with (red) and without (red) application of Hanning window

In **Figure 5b** the similarity of the sine and the cosine for each frequency from 0 to 5000 is shown before the calculation of the power spectrum. Hence, negative values are shown (around 400 Hz for sine wave similarity). Most of the correct information is now tracked by the cosine waves but this is purely a coincidence in this case as it only depends on the phase of the signal. We also observe a number of ripples around the peak, this happens due to the windowing effect, in a similar fashion as in autocorrelation analysis. In **Figure 5c** we observe the difference of the power spectrum with and without the application of hanning window to the signal.

Hanning[3] window suppresses the ripples. Its drawback though is that it also reduces and widens the actual peaks of the signal, since some information from the edges are lost. The peaks are not very sharp in this example. Peaks become sharper when larger buffer size is used. As mentioned also earlier, this is the frequency-time trade-off where better time accuracy causes lower frequency resolution and vice versa. Peaks are also present for higher partials, multiples of the 440Hz. This reveals the limitation of the simple frequency analysis directly on the fourier transform since pure sine waves are rarely encountered in music. Alternatively, harmonics (i.e. multiples of the frequency under analysis) can be also examined and taken into account.

Harmonic Product Spectrum analysis

Since timbre is generated by different amplitudes of harmonics, instead of analyzing a signal with a single sine wave, it can be analyzed by a second signal that contains more than one frequency. More specifically a signal can be generated for a specific frequency that also contains a number of harmonics, or in other words calculate the Harmonic Product Spectrum (HPS). This is mathematically expressed as

$$Y(f) = \prod_{r=1}^{R} |X(f)| ,$$

where r denotes the index of the harmonic. An example of this approach can be found in [Web4]. Since the timbre is unknown for a random musical source, the amplitude of harmonics have to be defined *a priori*. However a weighted product can be calculated in case of a source with known timbre. A serious drawback of this approach relies in polyphony; if harmonic notes also exist in the musical content, such as octaves and fifths, they might be mistakenly estimated as content of the lowest note.

Machine Learning & Neural Networks

As mentioned in the introduction, pitch detection is a challenging computational problem since a trained human ear can approximately identify the notes in a given melody, independently of the source's timbre. This suggests that the problem can be solved under certain conditions but it is rather unclear what these conditions are. For such complex problems that need to be solved but they can not be mathematically formulated precisely, it is common to apply machine learning methods that try to find solutions to complex problems through optimization procedures. If these methods work well they not only succeed in solving the problem but also provide insight on how this was achieved. **Artificial neural networks (ANNs)** constitute a field of machine learning that has lately displayed remarkable performance in complex tasks in several fields such as image classification (Ciresan et al., 2012; Bach et al., 2015), image enhancement and generation (Gatys et al., 2015; Zhu et al., 2017), clinical diagnosis (Amato et al., 2013), singing voice classification (see [Web2]) and audio generation (Bengio et al. 2016; Oord et al., 2017). Often shortened to neural networks (NNs), their original concept was to model or imitate neuronal activity in human brain; a large number of neurons interacting with each other and propagating

[3] Hanning window or Hann window: a window function often used to avoid aliasing in the frequency domain. Although highly similar to Hamming window, a different window function, they are not identical.

information in several layers. In a similar fashion as the human brain neurons that either fire or not, neurons in ANNs are typically modeled in a digital fashion with output 0 or 1, as well as the effect of training; neuron connections are adjusted according to previous knowledge in order to efficiently process new information. Since their conception, ANNs have developed and the aforementioned properties do not always hold, depending on the task and model. Computationally, the efficiency of NNs, can be summarized to their ability to detect complex relationships and features that describe the input data efficiently.

In the most common neural network structure, called Multilayer Perceptrons (MLP), the models consist of an input layer, an output layer and optionally a number of hidden layers in between. The input data is fed through the input layer and propagates towards the output layer while being subjected to a number of operations. These operations are typically multiplications with the neurons' weights and nonlinear functions in the output of each layer.

Neural networks are functional after proper training. Training is the most time-consuming process and requires several implementation decisions such as the type of input data, the type of output, the training set, its structure and a number of parameters that affect its optimization. However, once trained, a neural network is very fast.

In general, machine learning can be divided into two categories: unsupervised learning and supervised learning. Unsupervised learning refers to techniques that analyze data without any prior or extra knowledge regarding the data. A simple example of unsupervised learning is clustering, where the task is to divide the data into two or more categories without any knowledge on how the data were generated. On the other hand, supervised learning, refers to tasks where more knowledge is provided such as labels and tags for the data. In case of pitch detection, the data consist of sounds and their corresponding note names; this forms a classification problem, which is a subcategory of supervised learning. Although there are both unsupervised and supervised approaches for Neural Networks, here we use supervised models to classify the pitch of sound.

Types of input data

The simplest type of input data for a neural network addressing pitch detection would be to feed directly the samples. However, the inputs have to be consistent from sample to sample whereas temporal information is not, since for example the 2nd time point of a buffer does not have necessarily anything in common with the 2nd time point of the next buffer. This means that they they do not share the same information. Alternatively, frequency information of each buffer can be fed since each input will then correspond to a specific frequency. For example the power spectrum of the FFT is a meaningful input. However, many frequency bins of FFT fall out of the frequency range responsible for pitch and there they are redundant. An alternative approach is to use as inputs, information from the frequencies corresponding to the notes under analysis and a certain number of harmonics for each of them. Although conceptually phase is not expected to play any role in pitch perception, phase information is used to improve the frequency/time resolution through techniques like *phase unwrapping* (see for example [Web5]).

Training of the Neural Network

In machine learning, the efficiency of the model emerges from proper training that adjusts the model's parameters to suitable values. In the case of neural network, these parameters are the weights of the connections between neurons. A suitable training data set is required that is a sufficiently large number of examples with pre-known results. In case of pitch detection, a large number of sounds with known labels (i.e. pitches) is required. The process of obtaining a sufficiently large number of real examples is rather cumbersome so simulations were used here to generate sounds with random timbres and known pitches. The sounds were obtained by generating sounds with length equal to the buffer size and random harmonic amplitudes. We generated 1000 sounds for each note under study. It is necessary to produce an equal number of examples for each pitch since otherwise the model may successfully guess just by choosing the most common example even if no features are learnt. Without getting into further mathematical details, since they are beyond the scope of the thesis, the settings of the network are provided in brief as follows:

- **Number and size of layers**: Two hidden layers. The size of the first equals the number of harmonics and notes and analysis (e.g. for three octaves and 10 harmonics it would be 36x10=360 nodes in the first hidden layer) and the size of the second hidden layer was equal to the number of notes under analysis.
- **Optimization algorithm**: stochastic gradient descent
- **Minibatch size**: 50 samples
- **Learning rate**: 0.01
- **Activation function**: Rectifiers in the hidden layers and Softmax in the output layer.
- **Error function**: Least Absolute Error

Results

In this section we provide simple demonstrations of the algorithms in close to real applications. We show how they behave in a simple monophonic melody playing a C major scale downwards from C3 to C2 , in quarters at 180BPM. This results in 8 notes, with a duration of 0.33 seconds for each one. First we use simple sine waves for the melody and then a virtual piano (Edirol Super Quartet) for the exact same melody. We analyzed wave files with sample rate of 44.1kHz.

Zero crossing results

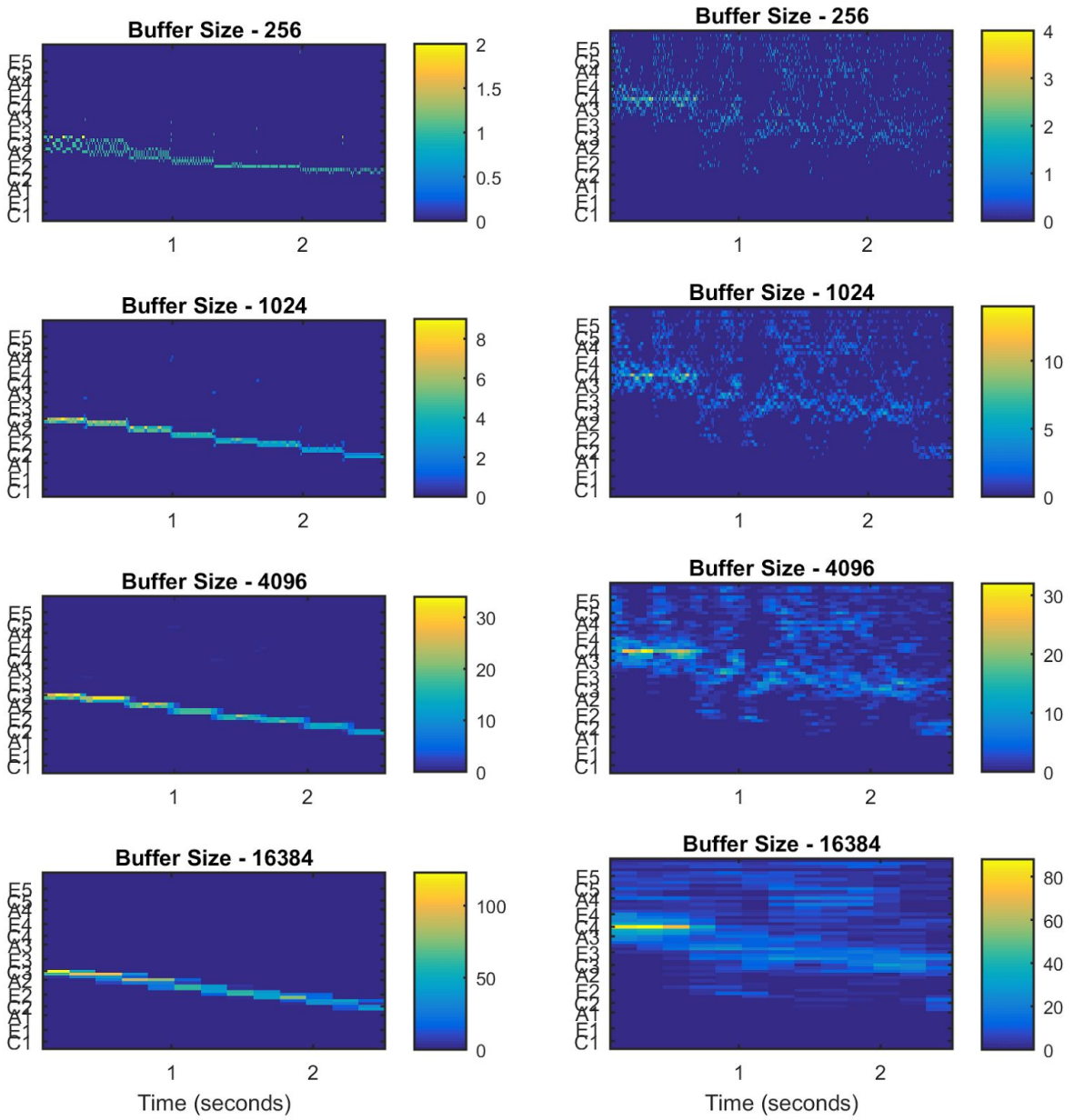

Figure 6: Zero crossing results for different buffer sizes for simple sine wave melody (left column) and simple piano melody (right column). Simple sine waves can be detected efficiently using zero crossing but its efficiency drops significantly in more complex sounds. In small buffer sizes there is large inaccuracy in the detected note, while in larger buffer sizes, there are overlapping areas close to the note transitions. The piano melody fails to be detected. The first harmonic of the first note (i.e. C4) is detected instead. As it will be shown later, in the fourier section, the first harmonic of the piano happens to be louder than the fundamental. Later notes complicate sound and zero crossing does not perform well although it maintains a downward trend.

Autocorrelation results

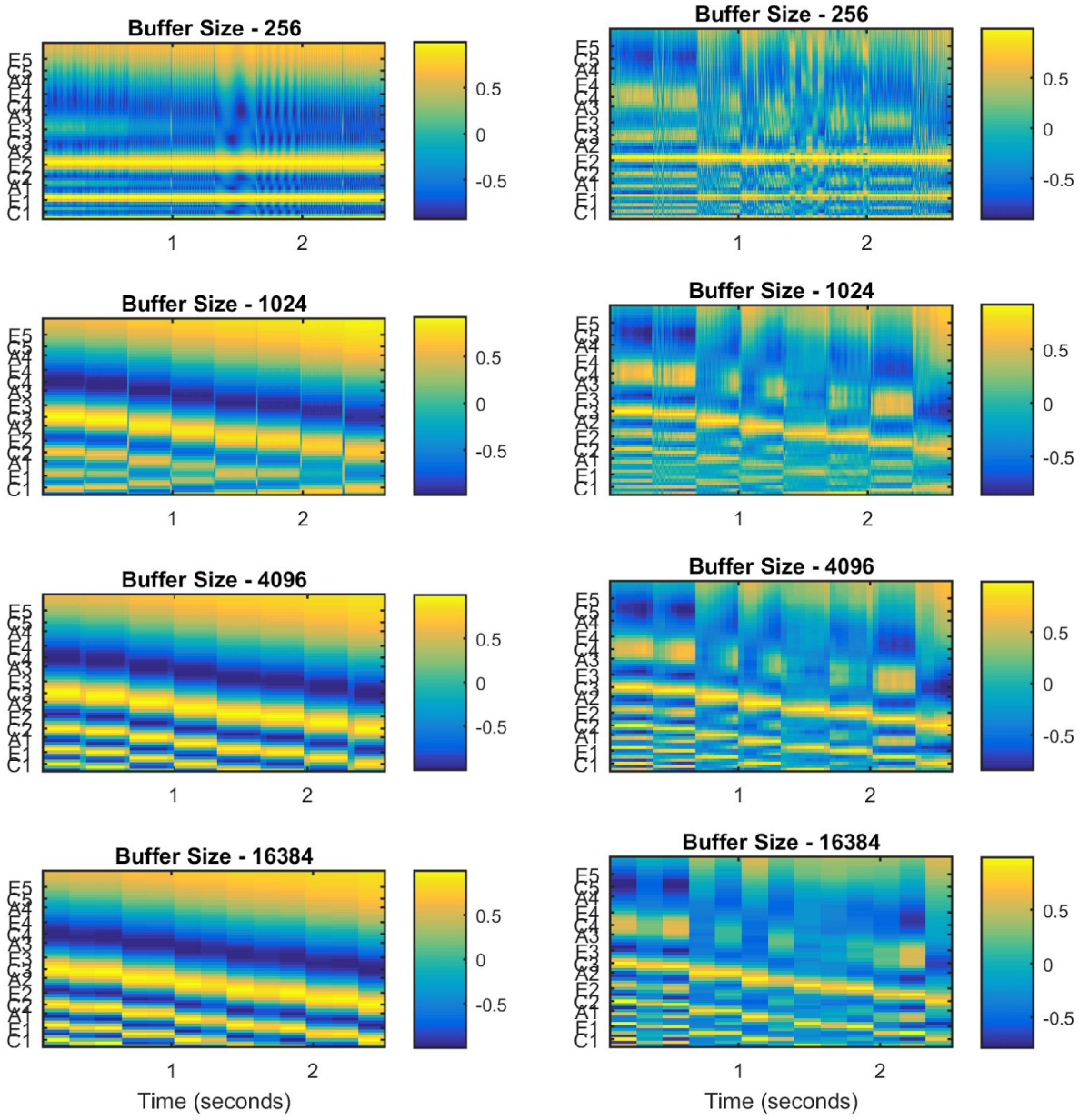

Figure 7: Autocorrelation results for different buffer sizes for simple sine wave melody (left column) and simple piano melody (right column). The results are similar to the demonstration in **Figure 4**, showing a high trend in the higher frequencies (i.e. higher notes) as well as rippling behavior in the lower frequencies. Again, too small buffer or too large buffer sizes hinder the results, by losing frequency and temporal accuracy respectively. In this case, the precision is better in lower frequencies, as higher notes have smoother patterns over different notes.

Multiple autocorrelation results

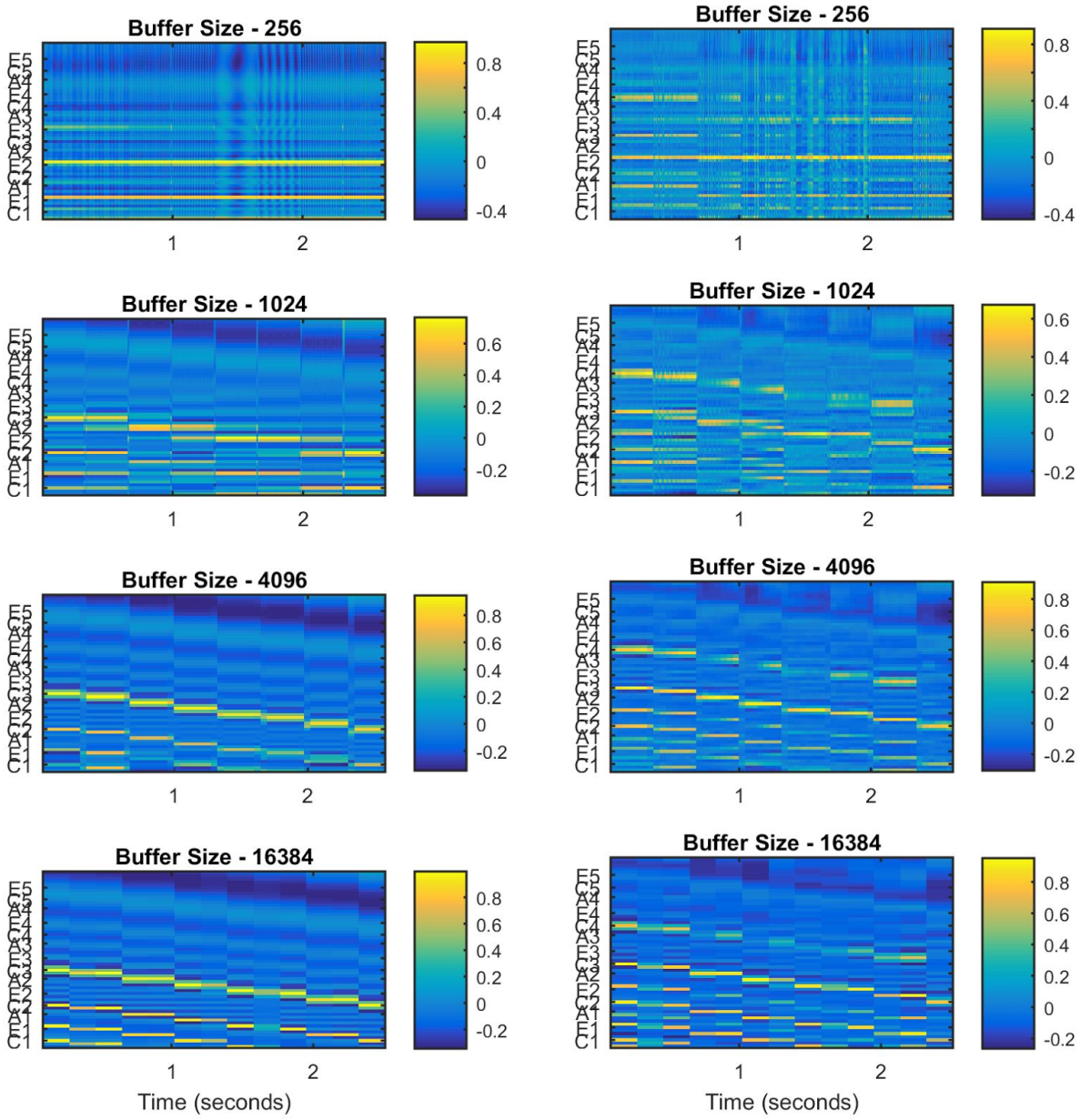

Figure 8: Multiple autocorrelation results for different buffer sizes for simple sine wave melody (left column) and simple piano melody (right column). Five autocorrelation samples were collected in this case (the sample was shifted circularly 5 times and the average correlation was calculated). The results are less noisy than simple autocorrelation results but the lower frequencies keep being present.

Fourier analysis results

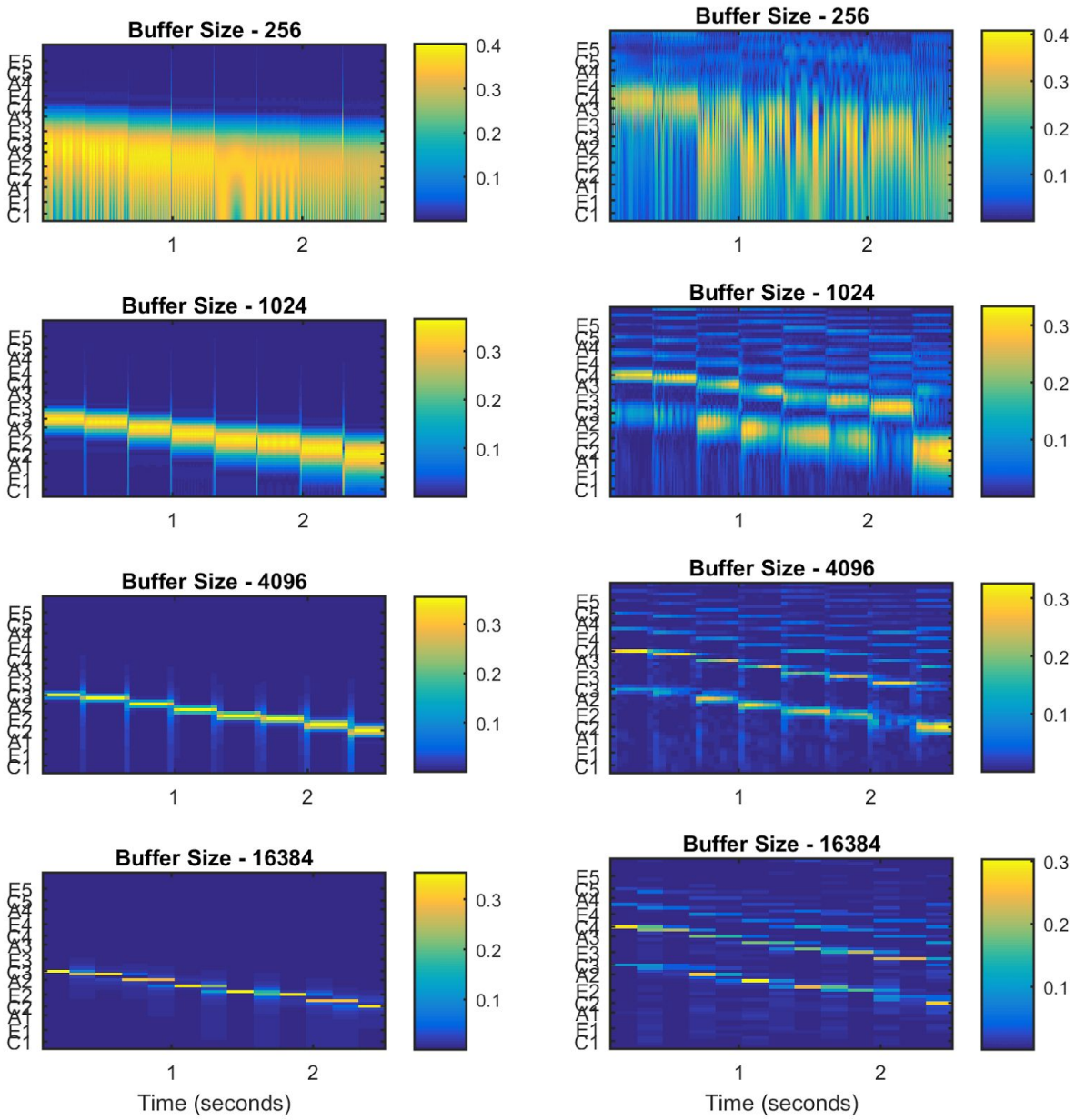

Figure 9: Simple Fourier analysis results for different buffer sizes for simple sine wave melody (left column) and simple piano melody (right column). Low buffer sizes lack frequency accuracy, large buffer sizes hinder temporal resolution. This approach works perfect for the sine wave melody. In the piano melody it detects all the harmonics. It is important to note that this example in practice is identical to what is often called *spectrogram*.

Harmonic Product Spectrum analysis results

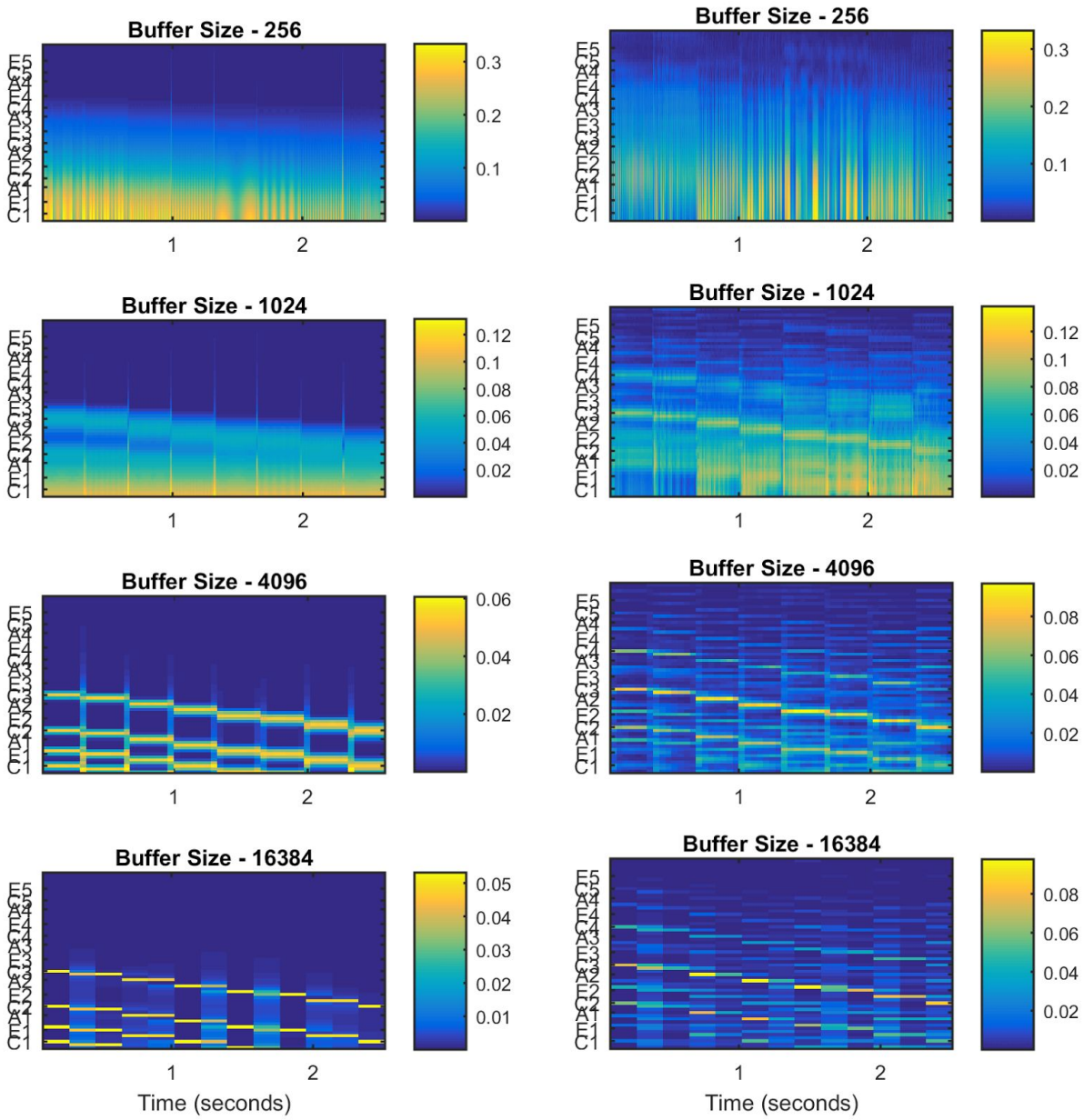

Figure 10: Harmonic Product Spectrum analysis results for different buffer sizes for simple sine wave melody (left column) and simple piano melody (right column). This sums in practice the energy of each note and its 5 harmonics. In the case of sine wave melody, lower notes seem active as they are subharmonics of the actual note present. Again, low buffer sizes lack frequency accuracy, large buffer sizes hinder temporal resolution.

Neural Network results

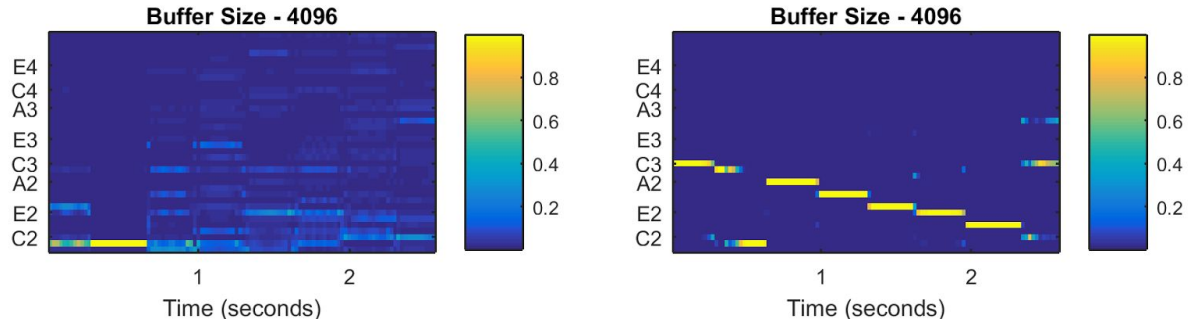

Figure 11: Neural network results for simple sine wave melody (left) and simple piano melody (right). Only one buffer size is demonstrated since training and testing of neural networks is laborious. Surprisingly, the neural network fails to detect the simple sine wave melody but detects the piano melody better than any other method. A tentative explanation for this result is that the training set consisted of spectrally complex sounds only. This implies that the generated training set was not descriptive enough to also cover simple cases, like sine waves.

Discussion

Monophonic versus polyphonic pitch detection

While monophonic pitch detection might seem as a subcase of polyphonic pitch detection, this is not always the case. In many implementations, under the assumption of only one existing pitch, methods may rely on minimizing some error function and locating the minimum. Even if they perform well for single pitch detection (see YIN pitch detection algorithm; de Cheveigné & Kawahara, 2002), generalization to polyphony may be intractable and require fundamental changes in the algorithm. The authors claim that the extension to music and polyphony is feasible by applying comb filters to remove irrelevant but periodic elements, but they state that this works well "*except in the unlucky event that the periods are in certain simple ratios*" which reflects exactly musical harmonies.
Other methods, such as zero-crossing, improve their robustness by averaging over a number of noisy observations. As shown also in **Figure 3**, a close to real estimation is achieved by averaging four zero-crossing points. However averaging results to a single number that ignores the possibility of more than one note being present.

Timbre and pre-defined frequency maps

In **Figure 5c**, the fourier analysis of an A4 piano note revealed amplitudes also in higher partials than the fundamental. This specific pattern of amplitudes of each harmonic defines the character of the instrument often termed as timbre. For example if a violin would play an A4, the fundamental (at 440Hz) should appear again but the peak amplitudes should be different. By studying these frequency patterns repeatedly and among different instruments, one may derive frequency maps that provide information for each instrument. This can assist when the sound

source is known, for example a piano recording and then perform weighting in a similar fashion as the harmonic series sine detection. Knowing the source can provide even more information, such as the frequency range of an instrument so that any content out of this range can be safely filtered or discarded.

Output of PDAs

Each PDA makes decisions regarding pitches through a specific measure. Optimally, related pitches yield high numbers while non-existant pitches yields low or zero numbers. In our implementation of zero crossing, the algorithm output reflects the number of times a frequency close to a certain note was detected. Using the example of **Figure 3**, durations that corresponds to 380.2, 393.8 and twice 512.8 Hz. These numbers correspond to detecting one F#4, one G4 and one C4. This also shows the lack of accuracy of the zero-crossing algorithm. In autocorrelation algorithms, the output is a number between -1 to 1. Values close to -1 imply negative correlation or in other words high similarity but phase difference of π. Values close to 1 show direct high similarity. Values close to 0 show no relation between the two signals under comparison. Frequency based approaches have the benefit of quantifying the energy of each spectral component. In other words, the higher amplitude a sinusoid has, the higher value the output will get. It is a desirable property of a PDA to be able to quantify the amplitude of a signal, e.g. tell the volume of a piano note detected. This is not possible with zero-crossing and autocorrelation methods. An alternative is to keep track of the local volume of the signal (buffer) through a low pass filter (i.e. envelope) in order to quantify the volume of the detected notes, but this approach does not allow volume quantification of multiple notes at the same time.

The output of the neural network varies, depending on the implementation of the network. Typically, neural network outputs range from 0 to 1, indicating the probability of an output to be true.

Preprocessing

Certain preprocessing steps have shown to improve results of PDAs. We demonstrated the benefits of windowing functions, such as Hann window, which help in removing edge effects. Filtering of the source signal is also a common process before applying PDAs. Removing spectral components that are out of the analysis frequency range may simplify the problem. Typically, band-pass filters are applied to remove spectral components that do not convey pitch information. In certain methods, such as zero-crossing method, band-pass filtering is extremely crucial as low or high components may cause zero-crossings that are not related to periodicity.

Preprocessing steps deriving from psychoacoustics and neuroscience of hearing also exist. These mostly take into account properties of the ear canal, cochlea and basilar membrane. For example it has been shown that hair cells in the human ear perform a half-wave rectification and hence pitch perception relies mainly on the positive part of the signal (Pulkki & Karjalainen, 2015; section 10.1.9). In a similar fashion various approaches perform band-pass filters similar to the ones that human ears perform (critical bands; see Fletcher, 1940). There have also been

attempts to approximate the problem of pitch detection through simulations of the cochlea and human hearing system in general (de Cheveigné, 2005).

State of the Art & future directions

Current state-of-the-art approaches for pitch detection are basically sophisticated extensions and combinations of the approaches mentioned in this thesis. An extension of the zero-crossing method that has shown remarkable results is PLL-tracker (Zölzer et al., 2012). The YIN algorithm (de Cheveigné & Kawahara, 2002) is an autocorrelation based method that has demonstrated notable results. An approach that incorporates both Harmonic product spectrum and neural networks has been presented by Guerrero-Turrubiates et al. (2014). A frequency-domain approach that also incorporates phase information has been introduced by Brown & Puckette (1993). Since neural networks have proven to be very efficient in a wide range of applications including audio with impressive results (Bengio et al., 2016; Oord et al., 2016), it is reasonable to attempt to use them for pitch detection. However, there are certain implementation decisions that must be taken, ranging from the type of input, the structure and the size of the network and the interpretation of the output. A Recurrent Neural Network (RNN) approach for pitch detection in piano has been recently introduced (Bock & Schedl, 2012) as well as a machine learning based toolbox in Python for music information retrieval (Korzeniowski et al., 2016).

A certain amount of research in pitch perception focuses on its neuroscientific perspective. This is a top-down approach aiming to understand each step in the process of human pitch perception. On the other hand, machine learning algorithms seem to be quite the opposite; building complex models that can effectively solve the problem and then study the model to unveil its mechanism. The second approach does not guarantee though that an efficient approach would resemble human hearing, but in musical applications this is not necessary.

Limitations

Since the purpose of this thesis is mainly demonstrative of the approaches and challenges faced by computational pitch detection, there are aspects that are not covered as well as aspects that are not optimized. The effect of bit depth and sample rate in the analysis are not shown here, neither the effects of various filters. However the code is available for download and further analysis.
In terms of neural networks, convolutional structures have be shown to allow signal input in the time domain (Su et al., 2016) and recurrent neural networks have shown to exploit previous time points to infer the current pitch (Bock et al., 2012). Furthermore, softmax as an output activation function may be limiting since it favors one output and hence is not optimal for polyphonic outputs.

Searching for a training set

A major implementation decision in supervised machine learning algorithms is what data to use as training data set. Data collection can be a tedious process, especially considering the

amount of samples required to train complex relations. Therefore, data collection is often performed in an automated manner, such as using web crawling. Web crawlers are programs that search the web and analyze data they find. Depending on the application, the web crawler can collect text, images or detect links that lead to sound files. Labeling of large amounts of data is also laborious, website often include tags related to the files. Online sound platforms such as freesound.org provide sounds for downloading, accompanied by related tags.

Another option is to use sound libraries that are used by synthesizers and samplers. In certain libraries, there is enormous detail in the data acquisition such as collecting sounds of all notes and in many different velocities.

Yet another way would be to naturally produce a training set while playing music through MIDI notes and audio output. This should be rather straightforward to implement but requires further work that is beyond the scope of this thesis.

Conclusions

We summarized the main challenges faced when dealing with pitch detection algorithms. A few fundamental pitch detections were implemented and tested in simple cases. A neural network implementation was also demonstrated using a simulated data set from scratch, showing interesting properties. Aims for this thesis were:

- To provide an introductory guide for anyone who wishes to elaborate further on pitch detection.
- To experiment with different pitch detection algorithms and underline their strengths and weaknesses.
- To train and test a machine learning algorithm without any prior sound library.
- To implement algorithms that will be publicly available. The code for the thesis can be found here: https://github.com/gostopa1/PDA

Based on the aforementioned goals, this thesis has covered a wide range of problems related to pitch detection. Further elaboration on pitch detection could easily be a topic for a whole research plan of a graduate student and hence beyond the scope of a Bachelor's thesis.

References

[1] Amato, F., López, A., Peña-Méndez, E. M., Vaňhara, P., Hampl, A., & Havel, J. (2013). Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine, 11(2), 47–58. https://doi.org/10.2478/v10136-012-0031-x

[2] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE, 10(7), 1–46. https://doi.org/10.1371/journal.pone.0130140

[3] Bengio, Y., Courville, A.C., Gulrajani, I., Jain, S., Kumar, K., Kumar, R., Mehri, S., & Sotelo, J. (2016). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. CoRR, abs/1612.07837.

[4] Bock, S., & Schedl, M. (2012). Polyphonic piano note transcription with recurrent neural networks. In ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (pp. 121–124). IEEE.

[5] Brown, J.C. & Puckette, M.S. (1993). A high resolution fundamental frequency determination based on phase changes of the Fourier transform. J. Acoust. Soc. Am. Volume 94, Issue 2, pp. 662-667 https://doi.org/10.1109/ICASSP.2012.6287832

[6] Bürck, W., Kotowski, P., Lichte, H. (1935). Elek. Nachrech., 12, 326 et 355.

[7] Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In 2012 IEEE Conference on Computer Vision and Pattern Recognition (pp. 3642–3649). IEEE. https://doi.org/10.1109/CVPR.2012.6248110

[8] de Cheveigné, A., & Kawahara, H. (2002). YIN, a fundamental frequency estimator for speech and music. The Journal of the Acoustical Society of America, 111(4), 1917–1930. https://doi.org/10.1121/1.1458024

[9] de Cheveigné, A. (2005). Pitch perception. Oxford Handbook of Auditory Science: Hearing, 71–104. https://doi.org/10.1093/oxfordhb/9780199233557.013.0004

[10] Fletcher, H. (1940). Auditory Patterns. Reviews of Modern Physics, 12(1), 47–65. https://doi.org/10.1103/RevModPhys.12.47

[11] Fletcher, H. & Munson, W.A. (1933). Loudness, its definition, measurement and calculation, Journal of the Acoustic Society of America 5, 82-108

[12] Gatys, L. a., Ecker, A. S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style, 3–7. Retrieved from http://arxiv.org/abs/1508.06576

[13] Guerrero-Turrubiates, J. d. J. , Gonzalez-Reyna, S. E., Ledesma-Orozco, S. E., & Avina-Cervantes, J. G. (2014) Pitch estimation for musical note recognition using Artificial Neural Networks, *2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, Cholula, 2014, pp. 53-58.

[14] Heller, E. J. (2012). Why You Hear What You Hear. *Princeton University Press* (Chapter 23; pp. 437-504)

[15] Huang, X., Acero, A., & Hon, H.W. (2001). Spoken Language Processing. Prentice–Hall, Upper Saddle River, NJ. Chapter 6

[16] Klapuri, A. P. (2004). Signal Processing Methods for the Automatic Transcription of Music. Doctoral Thesis. Tampere University of Technology. http://www.cs.tut.fi/sgn/arg/klap/phd/klap_phd.pdf
[17] Korzeniowski, F., Krebs, F., & Widmer, G. (2016). madmom: a new Python Audio and Music Signal Processing Library. https://arxiv.org/abs/1605.07008v1
[18] Licklider J.CR. “Periodicity” pitch and “place” pitch. J. Acoust. Soc. Am. 1954; 26:945.
[19] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. Retrieved from http://arxiv.org/abs/1609.03499
[20] Oxenham, A. J. (2012). Pitch Perception. Journal of Neuroscience, 32(39), 13335–13338. https://doi.org/10.1523/JNEUROSCI.3815-12.2012
[21] Pulkki, V., & Karjalainen, M. (2015). An introduction to speech, audio and psychoacoustics. Wiley Publishers.
[22] Rossing T.D., Moore R F and Wheeler P A (2002). The Science of Sound 3rd edition (Reading, M.A.: Addison-Wesley) pp 178–180
[23] Savart, F. (1840). Annalen der Physik und Chemie 53 555-561
[24] Schouten J.F. (1940). The residue and the mechanism of hearing. Proc. Kon. Akad. Wetenschap. 43:991–999
[25] Su, H., Zhang, H., Zhang, X., & Gao, G. (2016). Convolutional neural network for robust pitch determination. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 579–583). IEEE. https://doi.org/10.1109/ICASSP.2016.7471741
[26] Wishart, T. (1994). Audible Design. Orpheus The Pantomime Ltd.
[27] Zatorre, R. J. (2005). Neuroscience: finding the missing fundamental. Nature, 436(7054), 1093–4. https://doi.org/10.1038/4361093a
[28] Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. Retrieved from http://arxiv.org/abs/1703.10593
[29] Zölzer, U., Sankarababu, S. V. & Möller, S. (2012) PLL-based Pitch Detection and Tracking for Audio Signals. *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Piraeus, 2012, pp. 428-431. doi: 10.1109/IIH-MSP.2012.110

Web sources

[Web1] https://auditoryneuroscience.com/topics/missing-fundamental
[Web2] http://www.multimed.org/singing/
[Web3] http://whyyouhearwhatyouhear.com/TextandPDFFiles/pitch23.pdf
[Web4] http://musicweb.ucsd.edu/~trsmyth/analysis/Harmonic_Product_Spectrum.html
[Web5] https://www.surina.net/article/time-and-pitch-scaling.html