Modelling Professional Singers: A Bayesian Machine Learning Approach with Enhanced Real-time Pitch Contour Extraction and Onset Processing from an Extended Dataset

Behnam Faghih

A thesis submitted for the degree of

Doctor of Philosophy

Department of Computer Science

Maynooth University

Supervisor and Head of Department: Dr Joseph Timoney

October 2022

Table of Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to express my profound gratitude to my parents, Khorshid Faghih and Aniseh Attar, for their unwavering love, constant encouragement, and support throughout my life.

Next, a special thanks to my beloved wife, Fatemeh Sadeghi, for her love and support during this remarkable journey. Starting this PhD journey coincides with our marriage and having our first child, Hannah, and my thesis was submitted when we had our second child, Dara. She provided a supportive and peaceful situation that let me focus on my work and study whilst we were busy with our two little loves, who have shown us the most beautiful side of our life, and living apart from our families.

In addition, I would like to express my gratitude to Dr John Keating, Dr Ralf Bierig, and Prof John McDonald for their support during my research by providing me with feedback on my study plan and progress. Additionally, I appreciate Prof Adam Winstanley for his support as the head of the Computer Science department during my first months of study.

Gratitude extends to my viva panel, comprising Prof Ronan Railly, Prof Gerard Lacey, and Dr Andrew Hines, whose invaluable feedback and engaging discussions enriched my thesis.

Moreover, many thanks to Sutirtha Chakraborty, Azeema Yaseen, and Amin Shoari Nejad, whom I had the pleasure of collaborating with as co-authors in a couple of publications derived from my thesis.

Furthermore, thanks to Maynooth University and Higher Education Authority in the Department of Further and Higher Education, Research, Innovation and Science in Ireland for their financial support.

Last but not least, I am extremely grateful to Dr Joseph Timoney, my incredible supervisor and head of the CS department. I could not have undertaken this journey without his support. I have learnt not only a lot about music technology and signal processing from him but also his personality and attitude inspired me that made me a different person from when I started my study. I deeply appreciate his understanding and support during my study, which made my research smooth, relaxed, and enjoyable.

Abstract

Singing signals are one of the input data that computer systems need to analyse, and singing is part of all the cultures in the world. However, although there have been several studies on audio signal processing during the last three decades, it is still an active research area because most of the available algorithms in the literature require improvement due to the complexity of audio/music signals. More efforts are needed for analysing sounds/music in a real-time environment since the algorithms should work only on the past data, while in an offline system, all the required data are available. In addition, the complexity of the data will be increased if the audio signals come from singing due to the unique features of singing signals (such as vocal system, vibration, pitch drift, and tuning approach) that make the signals different and more complicated than those from an instrument.

This thesis is mainly focused on analysing singing signals and better understanding how trained- professional singers sing the pitch frequency and duration of the notes according to their position in a piece of music and the singing technique applied. To do this, it is discovered that by incorporating singing features, such as gender and BPM, a real-time pitch detection algorithm can be found to estimate fundamental frequencies with fewer errors. In addition, two novel algorithms were proposed, one for smoothing pitch contours and another for estimating onset, offset, and the transition between notes. These two algorithms showed better results as compared to several other state-of-the-art algorithms. Moreover, a new vocal dataset that included several annotations for 2688 singing files was published. Finally, this thesis presents two models for calculating pitches and the duration of notes according to their positions in a piece of music. In conclusion, optimizing results for pitch-oriented Music Information Retrieval (MIR) algorithms necessitates adapting/selecting them based on the unique characteristics of the signals. Achieving a universal algorithm that performs exceptionally well on all data types remains a formidable challenge given the current state of technology.

Chapter 1

Introduction

This chapter provides the objectives of this PhD study, a brief literature review, and the structure of the thesis. Some texts of this chapter come from one of our publications listed in the following.

- *Faghih, Behnam & Timoney, Joseph, "Considerations for the Next Generation of Singing Tutor Systems". AES 146th International Convention, Dublin, Ireland.*

1.1 Motivation

Since approximately 25 years ago, during my music classes where I learned to play the Santour, an Iranian instrument, I became fascinated by the variations in musical performances. Despite two musicians playing the same piece at the same tempo, their interpretations differed significantly. Initially, I pondered the influence of note loudness on these variations.

However, it became evident that loudness was not the sole factor modified by musicians during their performances. I discovered that they also altered the duration of notes without affecting the music's rhythm or meter. Additionally, I speculated that musicians, particularly in singing and with instruments capable of altering pitch frequency, might also manipulate notes' pitches. Moreover, I believed these changes were influenced by the position of the notes within a musical composition and the musicians' emotional state during the performance. This curiosity consumed me for many years as I sought to understand how trained professionals adjust pitch frequency, duration, and loudness in relation to note position, music genre, technique, instruments, and emotion. Naturally, I recognized that one study alone could not comprehensively answer this complex question.

To illustrate the problem, consider that a singer wants to sing the music score provided in Figure 1-1.

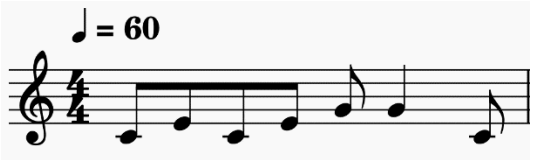


Figure 1-1. Musical scores, as an example of notes, that should be sung by a singer

The MIDI pitch code representation of the notes in Figure 1-1 is provided in Table 1-1.

Table 1-1. The corresponding MIDI pitch code of the notes in Figure 1-1.

Note	MIDI pitch code
C4	60
E4	64
G4	67

In addition, since the BPM of the music in Figure 1-1 is 60, the duration of the eighth and quarter notes are 0.5 and 1 second, respectively. Thus, the black lines in Figure 1-2 depict a visual representation of the notes.

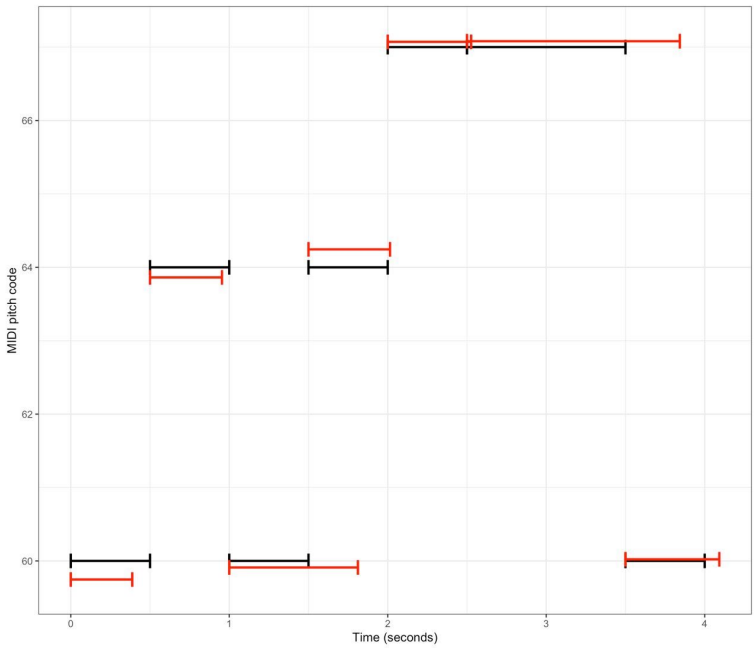


Figure 1-2. A diagram of a representation of the notes in Figure 1-1. The black lines are the theoretical MIDI pitch code and duration of the notes, while the red lines are a hypothetical performance of a singer.

As seen in Figure 1-2, the singer's performance, the red lines, may not fully align with the corresponding ground truth values, the black lines. Thus, the question is which of the notes sung by the singer in Figure 1-2 have a correct pitch and duration as compared to trained-professional singers. In addition, C4, as an example, in Figure 1-1, is repeated three times: at the beginning, middle, and end of the music. That brings another question: whether or not the position of a note in a piece of music affects how trained-professional singers perform the note's pitch and duration.

Consequently, during my master's thesis at Shiraz University, I focused on identifying the acceptable ranges of note durations as perceived by expert musicians. Building upon this research, my doctoral thesis aimed to delve deeper into how musicians manipulate note pitch and duration based on their position within a musical piece. This endeavour brought me closer to unravelling the answer I had sought for so long.

To accomplish this, I chose to analyse singing signals, as singers can modify pitch frequencies, note durations, and loudness, unlike certain instruments like the piano, where

the player lacks control over pitch frequency during a performance. In addition, singing is universal, as every culture has activities associated with music and song.

1.2 An overview of the studies on music tools

This section aims to provide a comprehensive overview of the tools and studies related to music technology, offering a broader perspective on this field of study.

Because of the widespread use of music in daily human life worldwide, researchers and industries are working on developing different software/tools related to music. The purposes of these tools are varied. The most common ones can be tools for listening to music, such as Spotify[1], YouTube Music[2], Apple Music[3], SoundCloud[4], and Amazon Music[5]. Some other ones for writing music notations include Finale[6], Sibelius[7], MuseScore[8], and ABC notation tools[9]. Regarding more advanced musical signal processing, the following list of applications, but not limited to them, can be considered.

- Source separation (Défossez *et al.*, 2019; Stöter *et al.*, 2019; Hennequin *et al.*, 2020)
- Audio to music notation convertor (Krige and Niesler, 2006; Ryynänen, 2006, 2008; Molina *et al.*, 2014; Dressler, 2016; McLeod *et al.*, 2017; Pesek, Leonardis and Marolt, 2017; Benetos *et al.*, 2019; Liu and Benetos, 2021)
- Noise removal (Berger, Coifman and Goldberg, 1994; Shetty and R, 2014; Abouzid *et al.*, 2019)
- Beat tracking (Goto, 2001; McKinney *et al.*, 2007; Al-Hussaini *et al.*, 2018; Chuang and Su, 2020)
- Music tempo estimation (Schreiber and Müller, 2018a, 2018b; Schreiber, Urbano and Müller, 2020)

[1] https://www.spotify.com/
[2] https://music.youtube.com/
[3] https://music.apple.com/
[4] https://soundcloud.com/
[5] https://music.amazon.com/
[6] https://www.finalemusic.com/
[7] https://www.avid.com/sibelius
[8] https://musescore.org/
[9] https://abcnotation.com/software

- Key detection (Weis, Schreiber and Muller, 2020)
- Query by humming like Shazam [1] and Shortcut for Google Sound Search [2]
- Music genre classification (Karatana and Yildiz, 2017; Pelchat and Gelowitz, 2020; Yu *et al.*, 2020)
- Musical instrument classification (Bhalke, Rao and Bormane, 2016; Racharla *et al.*, 2020)
- Score following (Cont, 2010; Nakamura, Nakamura and Sagayama, 2013, 2016; Dorfer, Arzt and Widmer, 2017)
- Optical Music Recognition (OMR) like SmartScore [3] and Scan-Score [4]
- Music recommender systems (Schedl, 2017; Schedl, Knees and Gouyon, 2017; Baracskay *et al.*, 2022)
- Automatic music generators like SoundRaw [5] and AmperMusic [6]
- Virtual Reality Instruments (Boem and Iwata, 2018; Zhang and Bryan-Kinns, 2022)
- Music education (Creech, 2020; Bharti, Singh and Malik, 2022)

Some of the studies are mainly on singing processing or analysis. The research and tools on singing are in varied areas, as listed in the following, but not limited.

- Pitch correction tools like Auto-Tune Pro [7], Waves Tune Real-Time [8], Melodyne [9], and Logic Pro [10]
- Separating vocal from a piece of music (Rafii and Pardo, 2012, 2013; Cano *et al.*, 2019)
- Singing education (Howard *et al.*, 2004; Nakano, Goto and Hiraga, 2006, 2007; Hoppe, Sadakata and Desain, 2006; Lal, 2006; Mayor, Bonada and Loscos, 2006; Mayor, Oscar., Bonada, Jordi., Loscos, 2009; Cano, Dittmar and Grollmisch, 2011;

[1] https://www.shazam.com/
[2] https://play.google.com/store/apps/details?id=com.rocketsauce83.musicsearch&gl=IE
[3] https://www.musitek.com/
[4] https://scan-score.com/
[5] https://soundraw.io/
[6] https://www.ampermusic.com/
[7] https://www.antarestech.com/product/auto-tune-pro/
[8] https://www.waves.com/plugins/waves-tune-real-time#presenting-waves-tune-real-time
[9] https://www.celemony.com/en/melodyne/what-is-melodyne
[10] https://www.apple.com/uk/logic-pro/

Molina, 2012; Abeßer *et al.*, 2013; Molina *et al.*, 2013; Lin *et al.*, 2014; Schramm, Nunes and Jung, 2015, 2016; Henry, 2015; Yu *et al.*, 2016; Tardón *et al.*, 2018; Gupta, Li and Wang, 2018; Luo *et al.*, 2018), Sing & See [1]

- Singer recognition (Tong Zhang, 2003; Wei-Ho Tsai and Hsin-Min Wang, 2006; Li and Li, 2018)
- Entertainment such as (Lal, 2006), Singstar [2], and Ultrastar [3]

1.3 Key common steps in musical signal processing

There are some typical steps in most musical signal processing applications that this study will follow the same approach, as depicted in Figure 1-3.

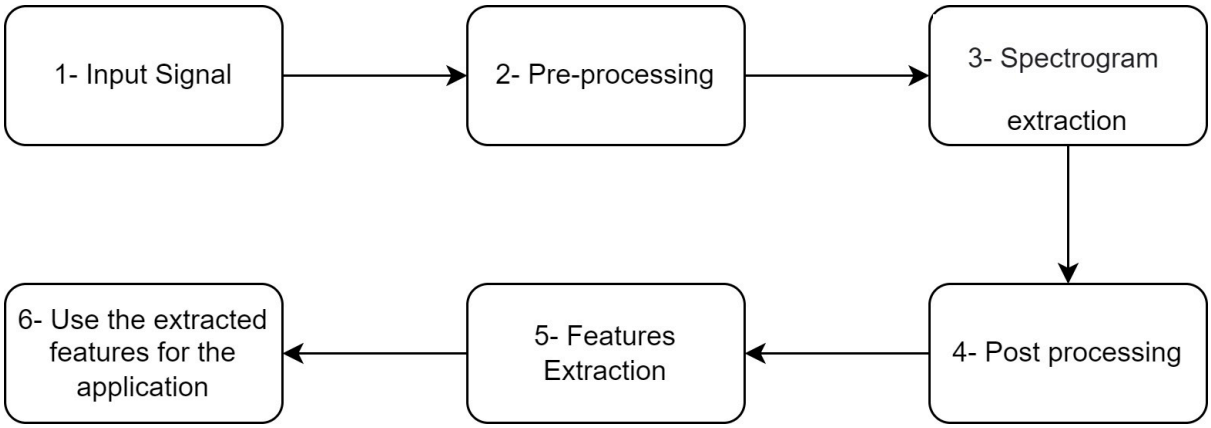


Figure 1-3. The fundamental steps of musical signal processing

1.3.1 Input signal

First, the input signals can directly come from a microphone (real-time) or a recorded file (offline). The input signal is usually raw data that need to be processed, similar to Figure 1-4.

[1] http://www.singandsee.com
[2] https://www.singstar.com
[3] http://ultrastardx.sourceforge.net/

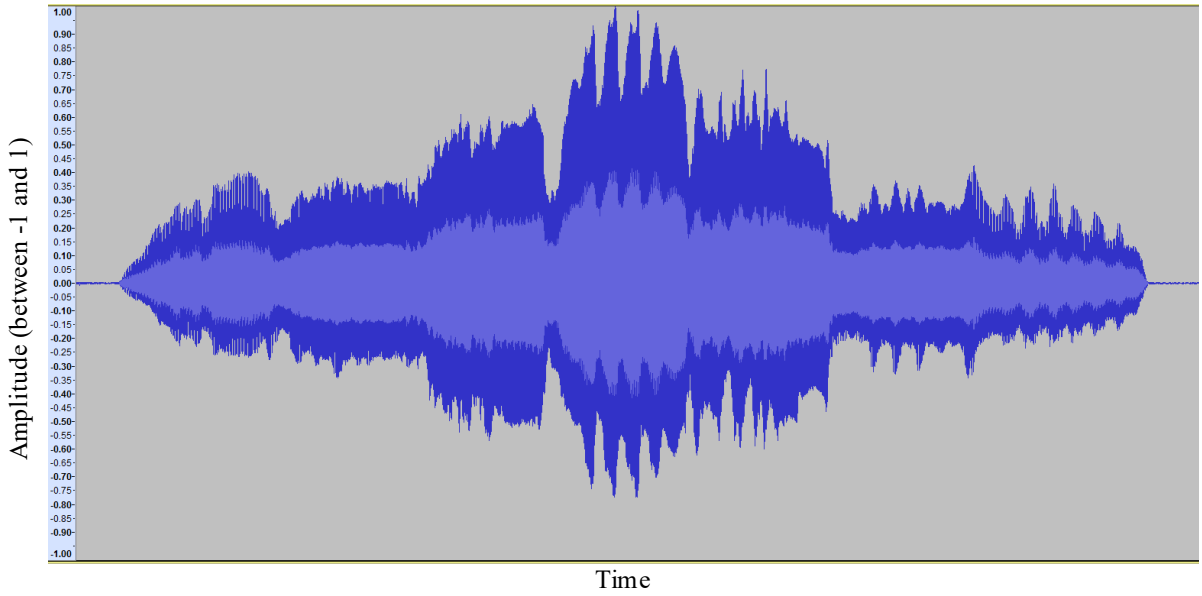


Figure 1-4 Raw audio input signals in time domain representation

1.3.2 Pre-processing

The input signals usually include some unnecessary data, such as noise or too high or low amplitude, that need to be altered in the second step, denoted as pre-processing in Figure 1-3. This thesis mainly analyses singing signals; thus, the source of the signals that will be processed in each step in Figure 1-3 will come from singing.

1.3.3 Spectrogram extraction

The sound spectrogram will be computed in the third step, usually by the Short-Time Fourier Transform, as shown in Figure 1-5(a). According to the applications, some layer(s) of the spectrogram will be used. One of the most crucial spectrogram layers of pitch frequencies is the fundamental frequency (F0), which is usually considered the layer to refer to the pitch frequency of the sound.

The F0 is the lowest frequency component of a periodic waveform. In music, the fundamental frequency represents the perceived lowest pitch of a note, corresponding to the lowest partial, as shown in Figure 1-5(b). The usage of the words “pitch” and “F0” is needed to be clarified. The word “pitch” usually refers to how the human brain perceives sounds, while the F0 refers to the lowest layer, the primary layer, of a sound that is directly related to human pitch perception. Thus, although F0 and pitch are not exactly the same, in this thesis, the difference between them was not counted, and these two words are considered

equivalent. The function of a pitch detection algorithm is to identify the frequency of F0 when it exists in the signal and otherwise flag that it is not present.

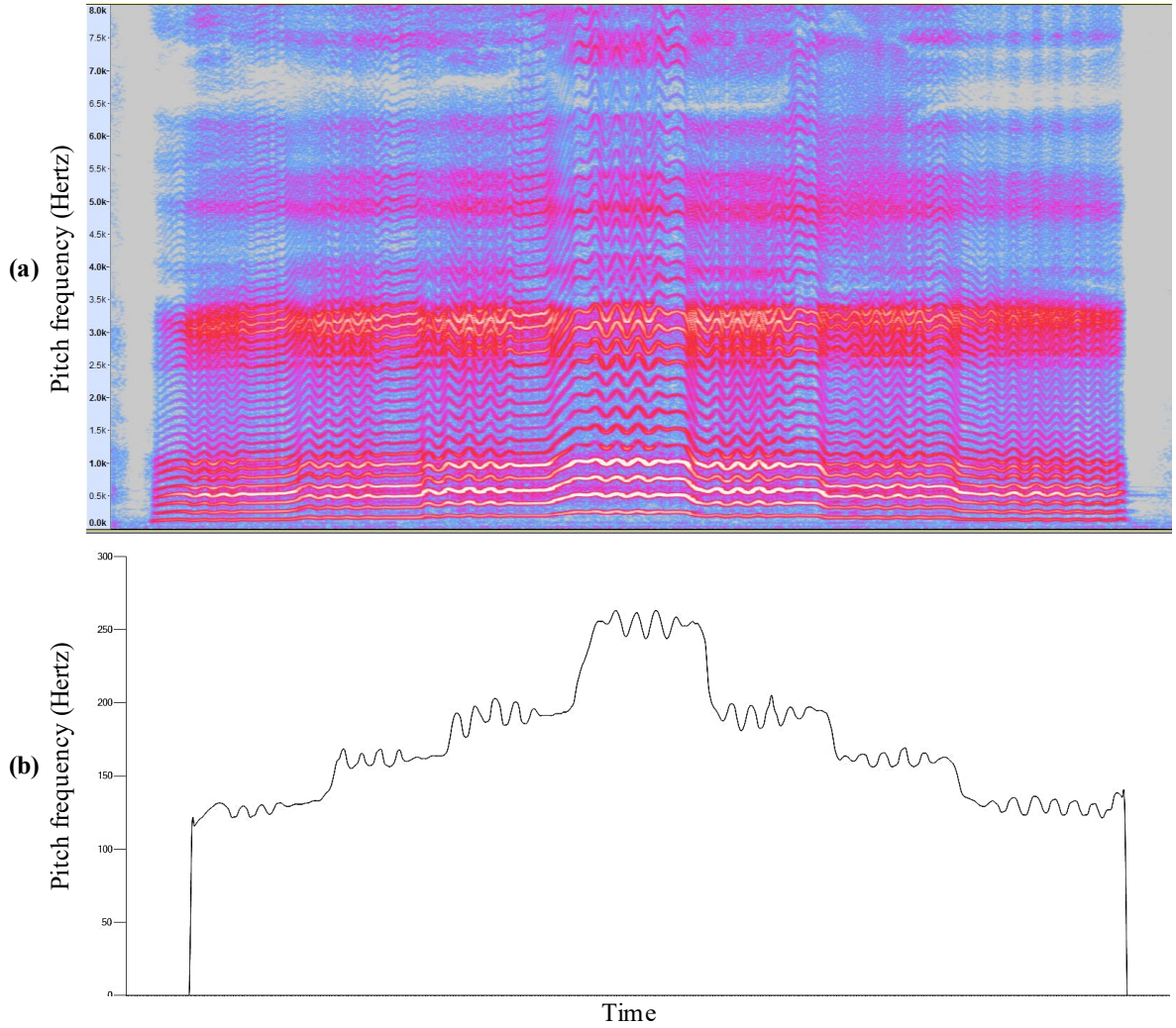


Figure 1-5 Different harmonies of the input signal, from Figure 1-4, represented in a pitch-frequency domain (spectrogram). (a) includes all harmonies in the input signal, (b) is the lowest harmony called fundamental frequency (F0).

1.3.4 Post-processing

There can be some errors in the estimated F0s because there are many harmonics besides the fundamental frequency in musical sounds like the human voice, as shown in Figure 1-5(a), whose energies may be greater than the fundamental frequency. In these cases, pitch detection algorithms are known to have difficulties and instead return an incorrect frequency value as F0. This problem is more common in real-time algorithms that detect a pitch contour.

Therefore, the extracted layers usually need some post-processing, as shown in Figure 1-6(a), such as pitch contour smoothing, as shown in Figure 1-6(b).

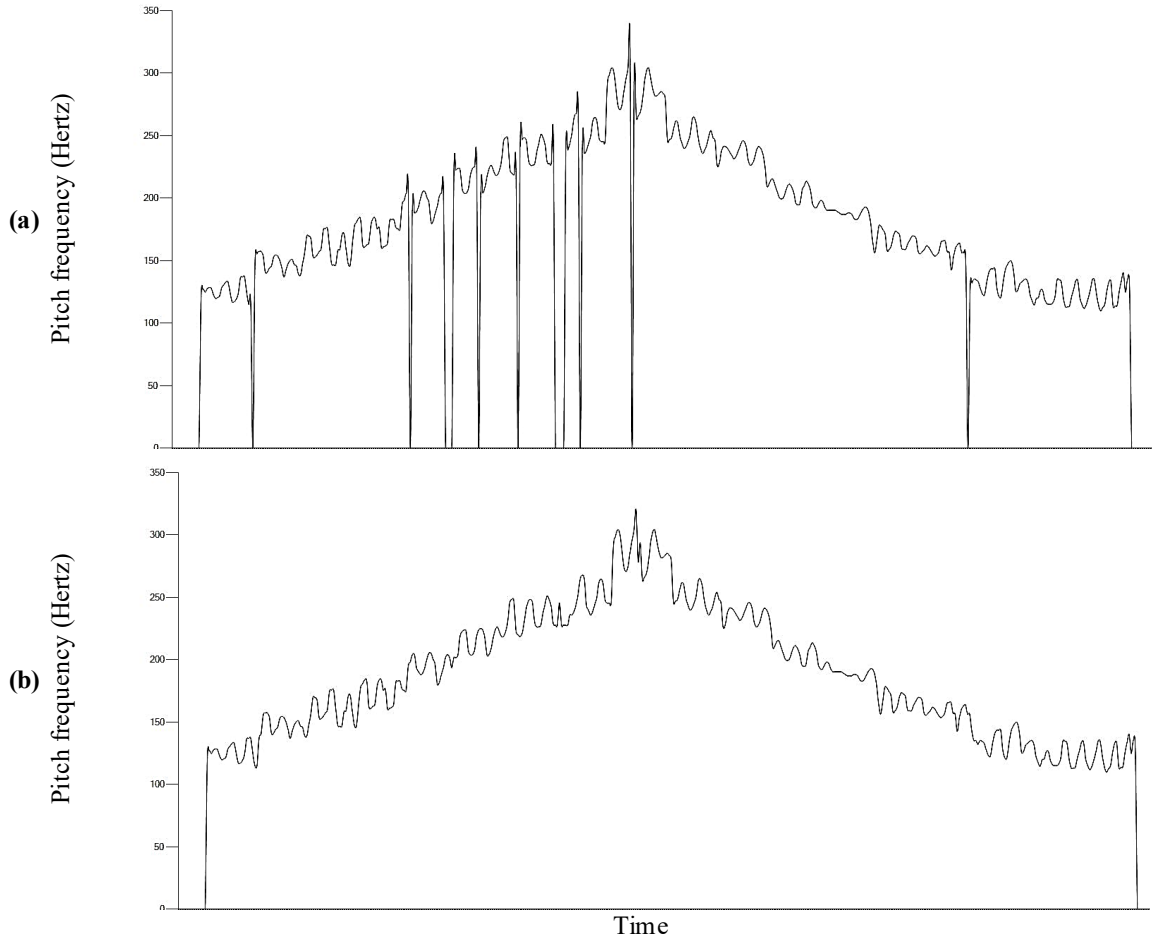


Figure 1-6 An example of post-processing in estimated F0s.(a) shows the output of a pitch estimator algorithm that included several errors. (b) shows the result of a pitch contour smoother on altering the errors in (a)

1.3.5 Features Extraction

The fifth step is extracting the required features, such as musical notes, as shown in Figure 1-7.

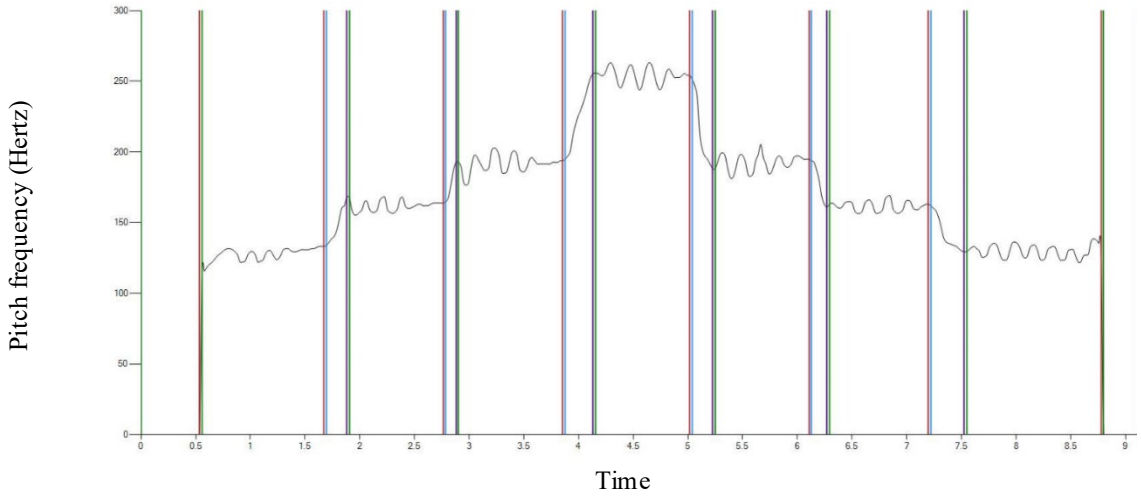


Figure 1-7 Extracted notes from a pitch contour. The vertical lines show onsets, offsets, and transitions between notes.

1.3.6 Processing extracted features

The last step is processing/analysing the extracted musical features from the previous step. The analysis that this thesis will do is with respect to the behaviour of trained-professional singers in changing notes' pitches and frequencies according to their positions in a piece of music. For example, if the music score is similar to Figure 1-8, then the pitch frequencies and duration of the notes, in theory, are shown in Table 1-2. Nevertheless, humans cannot play the notes exactly in these pitch frequencies and duration but in ranges of pitch frequencies and durations. But the question is what are the ranges within which the trained-professional singers would sing the notes? Another question is whether trained-professional singers have different behaviours in singing the same note when its position in a piece of music changes. For example, the note C4 in Figure 1-8 appears several times; thus, will a trained-professional singer change the pitch and duration of C4 according to its position in the music; if so, how? This thesis tries to answer these questions.

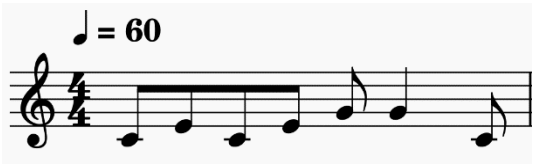


Figure 1-8 An example of a music score

Table 1-2 The theoretically pitch frequency and duration of each note in the example in Figure 1-8

Note	Pitch frequency (in Hertz)	Duration (in second)
C4	261.63	0.5
E4	329.63	0.5
C4	261.63	0.5
E4	329.63	0.5
G4	392.00	0.5
G4	392.00	1
C4	261.63	0.5

Therefore, for analysing the singing signals to answer my question regarding how singers would alter the pitch frequencies and duration of notes according to their positions in a piece of music, I needed to do all these steps to extract the notes' information to analyse them. At the beginning of my study, I found several algorithms for each step; thus, I had a plan to spend

most of my study analysing singers' behaviours in changing pitches, frequencies, and loudness. However, after implementing some of the available algorithms, I realized that they were not error-free, and I could not trust their results. Thus, I tried to design my own algorithms to improve the results.

1.4 Thesis objectives

This dissertation includes five principal objectives listed in the following.

1.4.1 Investigating real-time singing pitch detector algorithms

A principal operation for analysing musical sounds is usually detecting the pitches in the audio waveform because most subsequent analyses are based on identifying the correct notes and their pitches. Therefore, the purpose of this objective is to evaluate several real-time pitch detector algorithms to find the best ones according to the features of the singing signals, such as the singer's gender and the speed of the performance. Chapter 3 concludes with two reliable offline pitch detector algorithms to be used in other chapters. Then, Chapter 4 compares the accuracy of several real-time pitch detector algorithms to achieve this objective.

1.4.2 Real-time smoothing pitch contours generated from singing signals

I realized that the pitch detector algorithms produce some errors when estimating the F0 of singing signals, especially in real-time environments. Thus, these errors needed to be altered. In addition, I realized that the currently available contour smoothing algorithms have some difficulties in smoothing all the errors. Therefore, this objective aims to introduce a novel real-time algorithm to smooth pitch contours generated by pitch detection algorithms from singing signals. This objective will be achieved in Chapter 5.

1.4.3 Real-time onset, offset, and transition extraction from singing signals

Similarly, because of the limitations in the state-of-the-art algorithms in onset detection in real-time singing, this goal is to design a new real-time algorithm for extracting notes from a pitch contour. Chapter 6 discusses this objective in detail.

1.4.4 Generating an annotated singing dataset to analyse professional singers' performances

A fully annotated dataset including several singers and a variety of songs was needed to analyse singers' behaviours in changing pitch frequencies and duration. Thus, this objective aims to create a more precise dataset that is fully annotated to facilitate analysis of the singing notes. The considered annotations are pitch contour, onset, offset, the transition between notes, notes' fundamental frequencies and duration, pitch interval to the previous and following notes, ground truth note name and duration, etc.

1.4.5 Calculating notes' pitch frequencies and duration according to singing technique and their positions in a piece of music

Finally, after preparing all the required information from the previous objectives, this objective examines how trained-professional singers change notes' pitch frequencies and duration according to the position of the note in a piece of music. That is, performing musical notes correctly does not mean that all the performers play the notes at the exact same pitch and duration, but they normally perform the notes within acceptable psychoacoustic ranges, which may vary according to the position of the note in a piece of music. Nevertheless, these ranges are not determined yet, and this objective investigates creating some models to calculate these ranges. This objective will be discussed in Chapter 8.

1.5 Chapters summary

This thesis includes nine chapters. The first chapter, this chapter, provides an introduction to the aims of this study. Chapter 2 reviews related work and discusses several potential applications of the thesis objectives. Next, Chapter 3 compares four offline and real-time pitch detection algorithms and finds two reliable offline ones. Then, Chapter 4 investigates several real-time F0 detector algorithms to find the most accurate ones according to the features of the signing signals. Although the assessment from Chapter 4 results in a list of the best real-time F0 estimator algorithms, the estimated F0 contours still include significant errors. Thus, Chapter 5 proposes a new method for smoothing pitch contours generated from singing signals in real-time and offline environments to reduce errors.

Chapters 4 and 5 are related to each other in the way that Chapter 4 evaluates the performance of the smoother algorithm presented in Chapter 5 to show how it can improve the accuracy of the real-time pitch detection algorithms under different conditions, and Chapter 5 assesses the smoother algorithm according to the pitch detector algorithms presented in Chapter 4 but with a different approach. After correcting the F0 contours, Chapter 6 introduces a new algorithm to estimate the onset, offset, and transition points from the altered pitch contour to identify and extract the notes played by the singer in a real-time environment. After extracting the notes, Chapter 7 introduces a new annotated dataset called the Annotated-VocalSet. Subsequently, Chapter 8 analyses this new Annotated-VocalSet to determine the effect that a note's position in a piece of music has on its pitch frequency and duration. Finally, the thesis will be closed with a conclusion and a number of recommendations for future work in Chapter 9.

Chapter 2

Background and Applications

This chapter aims to review the literature and provide a list of applications of the objectives of this thesis. Some texts of this chapter come from two of our international conference publications listed in the following.

- *Faghih, Behnam & Timoney, Joseph, "Considerations for the Next Generation of Singing Tutor Systems". AES 146th International Convention, Dublin, Ireland*
- *Timoney, Joseph & Faghih, Behnam & Gibney, Anthony & Korady, Benjamin & Young, Gareth, "Singing Blocks: Considerations for a Virtual Reality Game to create chords and Progressions". International Conferences on Computer Simulation of Musical Creativity (CSMC2018), Dublin, Ireland.*

This chapter provides an overview of the related works to each of the objectives of this thesis. In addition, several applications of the objectives will be reviewed in this chapter. Before looking at the literature, some features of the singing signals that make them different from signals produced by musical instruments are discussed to emphasize separate studies on singing signals.

2.1 Features affecting the singing waveform

Although the results of the studies on musical instruments can be helpful in processing/analysing singing signals, separate studies are needed on singing by considering the ramifications of the particular singing features. Thus, this section discusses features of singing that can affect the analysis of the user's performance. These features are tunning pitch, vibrato, pitch drift, transitions between notes, and the vocal system. These features are strongly associated with singing and are not phenomena that occur in the playing of many musical instruments. Therefore, it shows why the signal processing algorithms that work well on instruments may not have the same accuracy and efficiency with singing signals. The features are discussed in the following.

2.1.1 Musical instruments are usually tuned

Most of the instruments can be tuned before playing. For example, a piano will be tuned before a performance, and if the instrument does not have bad quality, the instrument will remain in tune during the performance. In addition, a tuner, a tool for tuning musical instruments, can be used to tune the strings of a stringed instrument to exact frequencies. On the other hand, we cannot tune a human voice before singing. Therefore, it cannot be guaranteed that a person sings in tune or with exact pitch frequencies.

2.1.2 Vibration

"Vibrato corresponds to an almost sinusoidal undulation of F0 and thus can be called frequency vibrato. It can be described in terms of two parameters: (1) the rate, that is, the number of undulations occurring per second, and (2) the extent, that is, the depth of the modulation expressed in cents (one cent is a hundredth of a semitone)" (Sundberg, 2013).

Although it is possible to have a vibration in some instruments like the violin family, humans naturally use vibration when singing a song. Vibration makes the spectrogram appear more complicated.

Based on Prame's research (Prame, 1997), the vibrato rate lies typically between 5.5 and 6.5 Hz but tends to speed up somewhat toward the end of a long sustained tone. The extent of vibrato depends strongly on the singer and the repertoire but typically lies in the range of $\pm$30 cents and $\pm$120 cents, the mean across tones and singers being about $\pm$70 cents. In the Sundberg study (Sundberg, 2013), the vibrato rate was 6.5 undulations per second, and the extent was $\pm$30 cents.

Besouw et al. (Besouw, Brereton and Howard, 2008) presented three-tone ascending and descending arpeggios to musicians. The tuning of the middle tone, which either had or lacked vibrato, was varied, and the listeners were asked to decide which notes were in tune or untune. The results showed that the range of acceptable intonation of the middle tone was, on average, about 10 cents wider when it had vibrato than when it lacked vibrato. In addition, they found that if two voices sing perfectly "straight" (i.e., without vibrato), the demands on accuracy concerning the F0 are higher than if they sing with vibrato (Sundberg, 2013).

Another study conducted by D'Alessandro and Castellengo (D'Alessandro and Castellengo, 1991) measured the perceived pitch when tones are shorter than the duration of a vibrato cycle. They found that when presented alone, the rising half of a vibrato cycle was perceived as being 15 cents higher than the mean F0, while the falling half was perceived as 11 cents below the mean. They concluded that the ending of such short pitch glides is more significant to pitch perception than the beginning.

Therefore, vibrations significantly affect the perceptual pitch frequencies of the notes, and their effect depends on the note's properties, such as duration.

2.1.3 Pitch drift

Pitch drift or intonation drift means changes in tuning throughout a timescale of seconds or more while playing a piece of music (Seaton, Pim and Sharp, 2013). According to some studies (Alldahl, 2006; Ryynänen and Klapuri, 2006), pitch drift mainly occurs in the downward direction, i.e., downward intonation drift. In another study done by Müller et al. (Müller,

Grosche and Wiering, 2010), it is observed that pitch drift is common in unaccompanied solo folk singing. Similarly, Mauch et al. (Mauch, Frieler and Dixon, 2014) also found evidence of pitch drift in solo singing. They (Mauch, Frieler and Dixon, 2014) also realized that the pitch drift extent is often tiny (<0.2 semitones over 50 notes) and not correlated to pitch accuracy, interval accuracy, or musical background. Unlike the other studies, Mauch et al. (Mauch, Frieler and Dixon, 2014) observed that the most significant drifts are upward.

Therefore, singers shift their voice pitch while singing a piece of music, but it rarely happens when playing an instrument.

2.1.4 Transition between notes

In singing, when there is no rest between two consecutive notes, there is a smooth movement from the first note to the second note (Mayor, Bonada and Loscos, 2006). In other words, in some cases, the singer produces a series of frequencies between two consecutive notes to move smoothly from the first to the second, which is defined as the *Portamento* technique in music. On the other hand, playing the *Portamento* technique to ones' playing is impossible with some instruments, like the Piano, and they jump, in terms of frequency, from one note to the next.

Estimating the start and end of a note with a soft onset is more complicated than for one with a hard or sharp onset. A soft onset has a long attack duration or vague envelope shape that becomes a challenge to any peak-picking procedure. Figure 2-1 shows how a soft onset has a long and smooth movement between two consecutive notes, while the movement in a sharp onset is much quicker and is thus more easily distinguishable. The underlying reason for these issues is that the singing voice is classified as a pitched non-percussive (PNP) instrument, and PNP instruments still present a challenge for onset detection (Collins, 2005a).

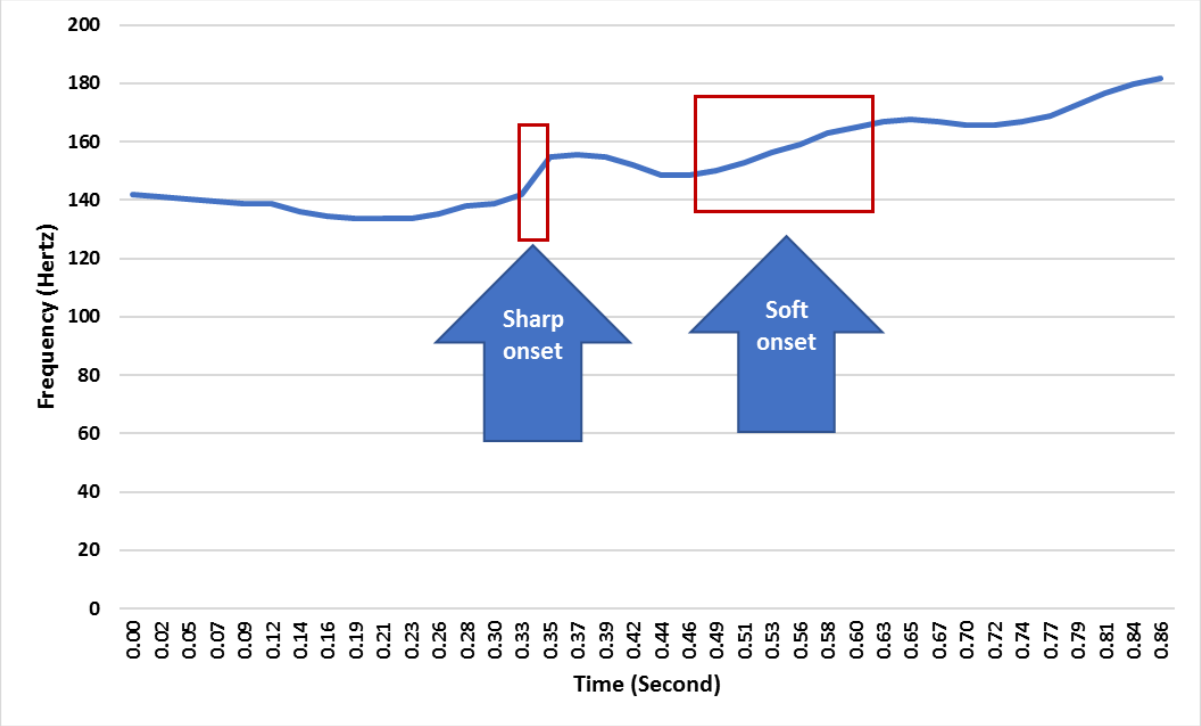


Figure 2-1 illustrations of soft onset as compared to a sharp onset

2.1.5 Vocal system

A singer must breathe in order to sing. Therefore, there is a natural constraint to the duration of notes in legato singing. In addition, there are some limitations to the pitch range and the length of the interval between notes. For example, it is easy for a beginner on the Piano to play A2 and then play A7 immediately afterwards, but this is impossible for a singer.

According to the above observations, there are some unique characteristics to singing signals as compared to signals from musical instruments, and thus, this demands separate studies for singing signals.

2.2 Literature review

This section provides a review of related works. Since this thesis includes several objectives, a review of the relevant work for each objective is separately discussed in the following.

2.2.1 Investigating real-time singing pitch detector algorithms objective

The estimation of the fundamental frequency (F0) of a waveform is known in the literature as the problem of pitch detection. This has been a long-standing task in signal processing, and many different algorithms have been proposed over the years. Up until about 20 years ago, the problem of monophonic pitch detection only was considered, but since then, the much more difficult task of polyphonic pitch detection has been tackled. Although some sample-by-sample detection methods have been proposed, most algorithms first separate the audio signal in short frames, generally of the order of 15-35 ms in length, within which it is assumed that the frequency information is stationary. The pitch is then computed for each frame. The analysis is done either in the time domain, using an algorithm that relies on computing the autocorrelation function or a variant, or in the frequency domain, using an algorithm that applies a type of Fourier transform. The initial algorithm outputs are assumed to be raw estimates that require post-processing. It is this later stage that can really differentiate the effectiveness of an approach.

The idea behind using the autocorrelation function for a time-domain algorithm is that when applying this function to a waveform, it should produce a representation that shows significant peaks at positions related to the period of the waveform, with the largest peak occurring first. A well-known variant is the Average Magnitude Different function, which was introduced as a computationally efficient alternative to the autocorrelation function. In more recent times, the Yin algorithm (de Cheveigné and Kawahara, 2002) has become very popular and uses a related cumulative mean normalized difference function. To enhance the accuracy of the estimate around the detected peak in the computed time-domain function, some form of interpolation is required, for example, parabolic interpolation.

For frequency-based methods, the frame is transformed into the frequency domain, often using the Fourier transform. An early algorithm implemented a further transformation of the Fourier spectrum into what was termed the 'Cepstrum'[1] (Noll, 1967), essentially dividing the spectrum into a fast-varying (because of pitch harmonics) and slowly-varying

[1] The cepstrum, in Fourier analysis, is obtained by performing the inverse Fourier transform (IFT) on the logarithm of the estimated signal spectrum. This method serves as a valuable tool for examining periodic structures within frequency spectra. Specifically, the power cepstrum finds applications in the analysis of human speech. The name "cepstrum" originates from reversing the first four letters of 'spectrum'.

components (because of the spectral envelope). Isolating the fast-varying component facilitated a pitch estimate. Another technique was the Harmonic Product Spectrum (Schroeder, 1968). This attempts to emphasize the harmonic peak of the fundamental component in the spectrum by a succession of decimations of the original spectral representation and then adding them together. More recent algorithms use a template approach where the spectral representation is compared with a template of the known fundamental frequency. The one with the best match signifies the pitch.

Both these techniques benefit from a tracking stage that follows. All algorithms can produce incorrect pitch estimates, particularly if the second harmonic or a subharmonic is too strong, leading to the problem of octave-doubling or octave-halving, or if the harmonicity of the signal is weak, leading to erroneous values. Thus, the pitch estimates need to be tracked and refined to remove any unexpected jumps from a 'smooth' contour. Tracking can be done in a forward manner, that is, as the estimates are produced and then it is determined how well they fit with previous values. It can also be done in a backward manner, using an algorithm such as dynamic programming, where estimates are obtained from the start to the end of the signal, and then the best possible contour is traced out from the end to the beginning. A good example of this is the pitch detector in the Pratt software package (Boersma and van Heuven, 2001). Another recent approach is PYIN (Mauch and Dixon, 2014), which has included a Hidden Markov Model (HMM) with the Yin algorithm (de Cheveigné and Kawahara, 2002). The HMM uses the Viterbi algorithm, which is a dynamic programming technique.

As an recent study on offline pitch detection is undertaken by de Obaldía and Zölzer (de Obaldía and Zölzer, 2019). They proposed an algorithm for improving the determination of correct pitch candidates by applying the autocorrelation function and simple heuristics. They evaluated their algorithm with a dataset of musical instruments files and a dataset of vocals. They found that the performance of their proposed algorithm was better than that of Yin (de Cheveigné and Kawahara, 2002), RAPT (Talkin, 1995) and PEFAC (Gonzalez and Brookes, 2014) algorithms. However, their proposed algorithm could not correctly estimate F0 in all cases. For example, errors occurred both in detecting the moments where transitions happened and for accurate voiced segment determination. In total, they obtained a correct pitch estimation around 90% of the time for the vocal sounds in the dataset they used.

Moreover, there are several studies, such as Drugman et al. (Drugman *et al.*, 2018) and Khadem-hosseini et al. (Khadem-hosseini *et al.*, 2020), on offline pitch detection by machine learning algorithms, which in essence apply forms of statistical methods to all the spectrogram channels of detected frequencies and are enhanced with training sets to estimate F0.

As expected, offline detection techniques are superior in accuracy as it is possible to detect the whole contour and then iteratively refine the result until it matches some criterion of optimality. On the other hand, real-time detection is more difficult as only previous pitch values are available for verifying the current value.

Recently, some sample-by-sample methods for pitch detection have appeared. This obviates the need for explicit tracking following the estimation as it is built into the algorithm. These use techniques from other areas of signal processing, that are the Phase-locked loop (PLL) (Zolzer, Sankarababu and Moller, 2012), a communications tool, and the Extended Kalman filter (Das, Smith and Chafe, 2017), which are more familiar in statistical signal detection. These take as input the audio signal and provide a value for the pitch at every sample. The initial PLL method was augmented to have a set of PLLs to track the pitch and select the most likely pitch value (Zolzer, Sankarababu and Moller, 2012). The Extended Kalman Filter can produce good results, according to (Das, Smith and Chafe, 2017), but care is required when setting the parameters of the signal model. Another recent work is the Harmonic locking Loop (Bittner, Wang and Bello, 2017), which extends the tracking idea to all harmonics to produce an improved estimate. The difficulty again is that parameter values need to be set. It is noteworthy that PLL and Kalman Filter have been evaluated mainly with instruments, that is, PLL with the cello (Zolzer, Sankarababu and Moller, 2012) and the Guitar (Böhler and Zölzer, 2016), and Kalman Filter with the Guitar (Das, Smith and Chafe, 2017).

One of the recent studies on pitch detection of the human voice in real-time is by Makov et al. (Makov *et al.*, 2019), and was based on the spectral approach. They compared their proposed algorithm with a ground truth assessed offline using the algorithm built into the Praat software (Boersma and van Heuven, 2001) and determined that their algorithm estimated F0 correctly, with less than a 1% difference on average with Praat. However, since the details of their evaluation, such as their dataset, are not provided, it is impossible to figure out precisely under which conditions their algorithm will work as claimed.

Once the pitch contour is found, the next stage is to convert this into a melodic representation. In the case of singing, it has to be recognized that singers use many techniques, such as portamento and vibrato, in their style, so a true description needs to retain these qualities (Besouw, Brereton and Howard, 2008; Anand *et al.*, 2012; Sundberg, 2013).

To sum up, as was mentioned, several offline pitch detector algorithms work well on both singing and instrumental signals, such as PYIN (Mauch and Dixon, 2014) and the Boersma algorithm (Boersma, 1993) implemented in the Praat tool (Boersma and van Heuven, 2001). However, a reliable real-time pitch detection algorithm for singing that works accurately in different circumstances has not been found. Most algorithms show reasonable results with speech, such as (Makov *et al.*, 2019) and a comparison between them is provided by Jouvet and Laprie (Jouvet and Laprie, 2017). In addition, several studies worked on offline pitch detection algorithms for singing signals, and a comparison across a selection of these algorithms was discussed in Gawlik and Wszołek's study (Gawlik and Wszołek, 2018). However, the real-time experiences in detecting F0 from singing signals have some limitations. That is, their accuracy is not high enough, as described in studies such as Das et al. (Das, Smith and Chafe, 2017, 2020) and reported by (Faghih and Timoney, 2019a).

Therefore, it is necessary to introduce a new algorithm or evaluate the current algorithms to find the best one for singing signals with respect to their different features, such as note duration, range of pitch frequencies, and the interval between notes. Thus, Chapter 3 compares some offline and real-time pitch detection algorithms on singing signals and identifies a trustworthy offline algorithm to be used for generating ground truth data that will be helpful for later assessments. Subsequently, Chapter 4 evaluates several real-time pitch detection algorithms based on the different singing features to determine the best real-time F0 detection algorithms for singing signals.

2.2.2 Real-time smoothing pitch contours generated from singing signals objective

It has been observed that none of the investigated real-time pitch detector algorithms could work perfectly on singing signals, and all of them return some errors in their output. Thus, the estimated pitch contours must be smoothed to minimise the errors/outliers.

These errors are often due to doubling or halving estimates of the true pitch value, and are therefore impulsive in appearance rather than random (Zhao, O'Shaughnessy and Nguyen, 2007; So, Jia and Cai, 2012; Faghih and Timoney, 2019a, 2022a; Ferro and Tamburini, 2019). Furthermore, incorrect pitch estimation often happens in real-time pitch detection, especially when the sound source is a human voice (Faghih and Timoney, 2022a). Therefore, a contour-smoother algorithm is necessary to filter the incorrectly estimated F0 before further analysis.

Generally, contour smoothers can be divided into two categories: 1—contour smoothing to show the data trend; and 2—contour smoothing to remove errors, noise, and outlier points.

There are several algorithms for showing a contour trend, such as polynomial (Luers and Wenning, 1971), spline (Craven and Wahba, 1978; Hutchinson and de Hoog, 1985), Gaussian (Deng and Cahill, 1994), Locally Weighted Scatterplot Smoothing (LOWESS) (Cleveland, 1979, 1981), and seasonal decomposition (Wen *et al.*, 2020). One of the applications of trend detection using pitch contours is to find out how two melodies are similar to each other (Lin, Wu and Kao, 2008; Wu, 2013; Chatterjee *et al.*, 2018; Sampaio, 2018). Other contour smoothers, such as moving average (Smith, 1999) and Median filter, function by attenuating or removing outliers in the contour (Faghih and Timoney, 2022a). None of these contour-smoother algorithms was explicitly designed for smoothing pitch contours; they can be used for any contour from any data series. They have been applied to smoothing pitch contours, such as in the study by Kasi and Zahorian (Kasi and Zahorian, 2002) that used the Median filter. However, there are certain adjusted versions of these algorithms for smoothing estimated pitches; for example, Okada et al. (Okada, Ishikawa and Ikegaya, 2016) and Jlassi et al. (Jlassi, Bouzid and Ellouze, 2016) introduced pitch contour algorithms based on the Median filter. In the following, some of these adjusted algorithms are discussed.

Zhao et al. (Zhao, O'Shaughnessy and Nguyen, 2007) introduced a pitch smoothing method for the Mandarin language based on autocorrelation and cepstral F0 detection approaches. They first used two pitch estimation techniques to determine two separate pitch contours, and then both were smoothed. Finally, combining the two smoothed pitch contours created a final smoothed contour. Generally, their approach was very similar to the idea of this study, moving through a pitch contour to identify noisy estimates by comparing each point

to its previous and succeeding points, and finally editing out the noise. However, their approach involved altering some correct parts of the data, which impacted peaks that were not incorrect. Moreover, in their evaluation, they only checked the error reduction capability of their algorithm for removing octave-doubling and sharp rises in estimated F0s. It would have been preferable to compare their smoothed contours with a ground truth to realize how well their algorithm could adjust the estimated contour to make it similar to that of the ground truth.

Liu et al. (Liu *et al.*, 2013) introduced a pitch-contour-smoother algorithm for Mandarin tone recognition. They used several thresholds for finding half, double, and triple errors by comparing each point with its previous point. Then, an incorrect frequency was doubled, halved, or divided by three, according to the type of error detected. They indicated that experiments should be carried out to determine the threshold values, but did not provide any guidelines for selecting or adjusting these. In addition, the threshold values they used were not revealed. Therefore, how one could change the thresholds to optimize the result is unclear. In addition, they tested their algorithm only on isolated Mandarin syllables, although realistically, they should also have tried their approach on continuously spoken language. Moreover, they did not compare the accuracy of their algorithm with other contour-smoother algorithms to show how well their method performed in relation to others.

The smoothing approach presented by Jlassi et al. (Jlassi, Bouzid and Ellouze, 2016) was designed for spoken English. Their smoothing system was based on the moving average filter. However, they only calculated the average of the two immediately previous F0 points for those points that showed more than a 30 Hz difference from their previous and following points. They compared their algorithm with the Median filter and Exponentially Weighted Moving Average (EWMA), and found improved accuracy using their approach. However, the dataset (Plante, Meyer and Ainsworth, 1995) used in their study was small, i.e., 15 people reading a short phonetically balanced text; that is, "The North Wind Story". Thus, their dataset included only 15 short files. Their results would have been much more convincing if they had evaluated their algorithm with a more extensive dataset generated by various pitch-detector algorithms. Moreover, several metrics could have been employed to measure how well they smoothed the errors. Furthermore, their algorithm considered a difference of more than 30 Hz from both the immediately previous and following points as an error; therefore, it was unable

to identify and smooth any errors existing over more than one point on the contour. Therefore, a dataset including pitch contours generated by different pitch detection algorithms to produce a variety of errors in a good balanced of human singing pitch frequencies range is needed to evaluate the smoother algorithms.

Ferro and Tamburini (Ferro and Tamburini, 2019) introduced another pitch-smoother technique for spoken English, based on Deep Neural Networks (DNN) and implemented explicitly as a Recurrent Neural Network (RNN). However, they did not compare the improvement offered by their approach and that of any other method. In addition, a comparison of their datasets and the mixture of datasets suggests that their DNN architecture may not work well with a new dataset.

As exemplified above, many pitch detection algorithms have been designed for and tested on speech. However, although both speech and singing are produced with the same human vocal system, because of the differences between speaking and singing, separate studies are required for the pitch analysis of singing (Gawlik and Wszołek, 2018). In addition, in real-time environments, the smoother algorithm should alter the contour within a reasonable delay, mainly based on previous data, because there is no future data. The reasonable delay can be varied according to the application of the pitch contour being used. For example, a shorter delay is needed in real-time to reproduce what the user is singing compared to a learning system that wants to show the user errors after singing each note.

The smoother algorithm should be based on the features and applications of the contour, similar to the approach taken by Ferro and Tamburini (Ferro and Tamburini, 2019) and the studies by So et al. (So, Jia and Cai, 2012) on smoothing contours generated from speech. In other words, expected error types in the pitch contours for the specific data type should be identified. Then, an investigation for a targeted contour-smoother algorithm to solve these errors should be made. In addition, the applications of the smoothed contour should also be considered. For example, when a highly accurate estimate of the F0 value at each point is required, the smoother algorithm should not change any data except those points identified as having been incorrectly estimated. Moreover, the smoother algorithm should not have a significant delay in real-time environments. Therefore, several

considerations have been ascertained that a real-time smoother algorithm should consider for singing signals as follows.

2.2.2.1 Considerations for smoothing pitch contour of singing signals

This section lists the considerations that need to be adhered to when designing a real-time smoother algorithm for signing signals.

1) Only the incorrectly estimated pitches need to be changed. Therefore, it is necessary to decide which jumps in a contour are incorrect.
2) Some of the estimated pitches around the incorrectly detected F0 should be selected to alter their values. This represents the window length for the calculation. Therefore, the decision on the number of estimated pitches before and/or after the erroneously estimated pitches requires to define a window length. Thus, a buffer delay is required in real-time scenarios to ensure that sufficient successive pitch frequencies are available when correcting the current pitch frequency.
3) There is a minimum duration for which a human can sing.
4) There is a minimum duration for which a human can rest between singing two notes.
5) There is a maximum frequency that a human can sing.
6) There is a maximum interval during which humans can move from one note to another when singing.
7) To sing a large pitch interval in a very short time is impossible.

Table 2-1 shows a list of the contour smoother algorithms demonstrating which of the considerations mentioned above they cover. As the table shows, none of the algorithms have addressed all the considerations.

Table 2-1. A list of the contour smoother algorithms with indicating the code(s) of their considerations according to the list in section 2.2.2.1

Algorithm	The Considerations
Gaussian (sigma = 1)	2
Savitzky–Golay filter	2
Exponential	2
Window-based (window_type = ‘rectangular)	2
Window-based (window_type = ‘hanning’)	2
Window-based (window_type = ‘hamming’)	2
Window-based (window_type = ‘bartlett’)	2
Window-based (window_type = ‘blackman’)	2
Direct Spectral	2
Polynomial	2
Spline (type = ‘linear_spline’)	2
Spline (type = ‘cubic_spline’)	2
Spline (type = ‘natural_cubic_spline’)	2
Gaussian (sigma = 0.2, n_knots = 10)	2
Binner	2
LOWESS	2
Decompose (type = ‘Window-based’, method = ‘additive’)	2
Decompose (type = ‘lowess’, method = ‘additive’)	2
Decompose (type = ‘natural_cubic_spline’, method = ‘additive’)	2
Decompose (type = ‘natural_cubic_spline’, method = ‘multiplicative’)	2
Decompose (type = ‘lowess’, method = ‘multiplicative’)	2
Decompose (type = ‘natural_cubic_spline’, method = ‘multiplicative’)	2
Kalman (component = ‘level’)	2
Kalman (component = ‘level_trend’)	2
Kalman (component = ‘level_season’)	2
Kalman (component = ‘level_trend_season’)	2
Kalman (component = ‘level_longseason’)	2
Kalman (component = ‘level_trend_longseason’)	2
Kalman (component = ‘level_season_longseason’)	2
Kalman (component = ‘level_trend_season_longseason’)	2
Moving Average (simple = True)	2
Moving Average (simple = False)	2
Median Filter	2
Okada Filter	1, 2
Jlassi Filter	1, 2, 7

In summary, most of the contour smoother algorithms available in the literature were not explicitly designed for musical signals, but they have been used in a number of studies for smoothing pitch contours, such as the study by Kasi and Zahorian (Kasi and Zahorian, 2002) that uses the Median filter. Furthermore, some of them have only been tested on smoothing the pitch contour for speech signals, such as the Jlassi et al. (Jlassi, Bouzid and Ellouze, 2016) study designed for smoothing pitch contour for spoken English. However, employing them to smooth the F0 contours derived from singing signals may not be appropriate due to the different features associated with singing, as discussed in section 2.1. Therefore, it is necessary to design a novel algorithm for smoothing the F0 contours from singing signals in real-time and offline environments, as done in Chapter 5.

2.2.3 Real-time onset, offset, and transition extraction from singing signals objective

One of the fundamental processes of analysing audio signals is finding the start and endpoint of the notes, which are called the onset and the offset, respectively. Onset and offset are not exact points/times universally agreed as the starting and ending of a note but exist within an acceptable range (Hoon Heo, Dooyong Sung and Kyogu Lee, 2013; Choi *et al.*, 2020; Rosenzweig, Scherbaum, *et al.*, 2020; Bittner *et al.*, 2021).

Several applications need the results of onset/offset detection, such as tempo and pitch estimation, beat tracking, score following, automatic music transcription, and analysis of recorded music. Real-time music applications demand almost instantaneous results, i.e., real-time onset detection for systems such as the interactive music systems explained in Müller-Rakow (Müller-Rakow and Flechtner, 2017) and Malloch (Malloch *et al.*, 2019), or for music transcriptions as discussed by Kroher and Díaz-Báñez (Kroher and Díaz-Báñez, 2019). Therefore, minimising the time delay between the onset or offset and their detection in real-time environments is vital.

Over the years, many research contributions have been made for onset detection, but most work offline. If the onset detection function has been appropriately created, then onset events will give rise to well-localized recognizable features, e.g., a peak, in the detection function (Bello *et al.*, 2005). Several common approaches for detecting onsets, such as spectral difference, phase deviation, wavelet regularity modulus, negative log-likelihood, and high-

frequency content, are well explained in the Bello et al. (Bello *et al.*, 2005) study and then compared by Collins (Collins, 2005b). Moreover, Dixon (Dixon, 2006) has proposed multiple future enhancements for some of these methods.

In addition, Lacoste and Eck (Lacoste and Eck, 2006) propose an offline music onset detection algorithm using single and combined versions of Artificial Neural Networks (ANN) trained with different hyperparameters, and Eyben et al. (Eyben *et al.*, 2010) employ a Recurrent Neural Network (RNN) based on Mel spectrograms. Furthermore, after pre-processing with a time-variant filter, a method using Hidden Markov Models (HMMs) was proposed by Degara et al. (Degara *et al.*, 2011) for offline onset detection. Schlüter and Böck (Schluter and Bock, 2014) refined the model proposed by Eyben et al. (Eyben *et al.*, 2010) and trained Convolutional Neural Networks (CNNs) with mini-batch gradient descent (this splits the training dataset into small batches) to reduce model error, and the input to their model was two log Mel-spectrograms. Their approach outperformed other traditional methods and required less additional processing. However, the peak-picking approaches used for CNN and RNN-based methods rely on future information (not probabilistic) to detect an event; thus, they cannot work for real-time music onset detection.

Some of the studies are focused on detecting onsets from singing signals. For instance, the singing onset detection method of Toh et al. (Toh, Zhang and Wang, 2008) is based on audio features such as Mel Frequency Cepstral Coefficients, Linear Predictive Cepstrum Coefficients, pitch stability zero-crossing rate, and signal periodicity. First, the extracted audio features are classified into onset and non-onset frames using Gaussian Mixture Models (GMM). After GMM scoring, the feature evaluation is preceded by a dual detection function (feature level and decision level fusion) for higher accuracy in selecting the most optimal features. This method resulted in an 86.5% precision, 83.9% recall, and an F-measure of 85.2%. The recall shows the proportion of the cases correctly predicted positive. Use of positive precision implies the fraction of predicted positive cases determined to be real positive. In binary classification, the F-measure calculates a test's accuracy. It is calculated from the precision and recall of the test. The F-measure is the harmonic mean of the precision and the recall. The value of an F-measure is between 0 and 1. The highest value specifies perfect precision and recall, while the lowest shows whether the precision or the recall is zero

(Powers, 2020). However, despite the high F-measure score, it was still possible that their result could contain bias because of the dataset they used. The training and test set came from a tiny dataset comprising 18 singing recordings from four singers with 1127 onsets. This amount of files may not generate a variety of conditions to sufficiently challenge an onset detector algorithm in different situations, e. g., different sequences of pitch intervals and singing techniques.

In the study conducted by Gong and Serra (Gong and Serra, 2018), a deep learning model was trained for musical onset detection in solo singing, and the authors discussed how their algorithm could lead to improve live onset detection models. They used two datasets, one of which contains more than 25,000 onsets, mostly complex mixtures or solo instrumental excerpts, and only three excerpts are of a solo singing voice, and the other dataset is a subset of a solo Jingju singing voice that contains 100 recordings. They employed seven deep learning-based architectures.

In the Gong and Serra (Gong and Serra, 2018) study, it was preferred to use the score-informed method if the musical score information was available. Score-informed approaches evaluate the data with the assistance of musical scores. Based on the results, score-informed HMM outperformed peak picking for all of the architectures used in this experiment (Gong and Serra, 2018). The reported F-measure for the combination of the peak picking method and a no-dense neural network architecture was 73.88%, with a *p*-value of 0.002. For the score-informed HMM method, a nine-layer CNN architecture worked best, giving an F-measure of 80.90% and a *p*-value of 0.001. Learning strategies for inter-dataset knowledge transfer were also studied, but due to the features of different musical patterns, the authors claimed that when the musical patterns from the two datasets used to train their model were different, the onset prediction was not accurate.

Despite these studies, the onset detection of a musical note remains a challenge, primarily for the singing voice. Chang and Lee (Chang and Lee, 2014) explain several reasons, including articulation inconsistency, singer-dependent tonal quality, and gradual variation in onset envelopes over time. In other words, the time-varying spectral envelope and the inconsistency of vocal tracks may produce fake maxima (i.e., peaks) in an onset detection function that can lower the precision rate for onset detection. Therefore, detecting onsets

from the singing voice is still an active area of study because of waveform unpredictability and the occurrence of many noisy segments. Moreover, most methods are only suitable for recorded singing and are designed to work offline.

According to the previously published results, most existing approaches do not work well for soft onsets, including singing music. A soft onset has a long attack duration or vague envelope shape that becomes a challenge to the peak-picking procedure. The underlying reason for these issues is that the singing voice is classified as a pitched non-percussive (PNP) instrument, and PNP instruments still present a challenge for onset detection (Collins, 2005b). The nature of the singing voice adds further complexity due to its natural inconsistency with respect to pitch and time dynamics. Unlike some instruments, whose timbre is usually consistent throughout a note, the singing voice inherently can produce more variations of formant structures (for articulation); sometimes, it may even vary within the duration of a single note (Lindblom and Sundberg, 2007). While most onset detection algorithms are based on detecting spectral changes, they can fail to differentiate such variations in a singing voice because of singing features such as vibration and soft onset.

Relevant challenges for onset detection in solo singing voices were identified in a report from the Music Information Retrieval Evaluation eXchange 2012 (MIREX 2012). According to this report, the best-performing detection method gives an F-measure of only 55.9% (Hoon Heo, Dooyong Sung and Kyogu Lee, 2013), which becomes even lower for solo sustained strings with an average F-measure of 52.8%. In addition, training datasets for dynamically changing patterns in a singing voice is still a challenge (Gong and Serra, 2018; Schindler, Lidy and Böck, 2020).

One of the missing parts of most of the onset detection algorithms is consideration of the actual singing style features. In the Mayor et al. (Mayor, Bonada and Loscos, 2006) study, it is shown that one of the crucial features that should be taken into account in onset detection is the transition from one note to another note where there is no intervening silence, i.e., the legato singing (Mayor, Bonada and Loscos, 2006). The transition means that a singer will take a while to reach the target note. If the time for the transiting is not incorporated, the onset detector cannot find the correct times for onset and offset events. These transitions are categorized as a soft onset.

Therefore, as discussed in section 2.1, singing signals have some unique features that significantly impact the approaches to detecting onsets. Thus, it was found to be necessary to design a new onset detector algorithm according to the singing features. Thus, Chapter 6 explains the details of a novel real-time onset detection algorithm for singing sounds.

2.2.4 Generating an annotated singing dataset objective

Datasets are fundamental for analysing and understanding relationships and causes. Gathering an adequate store of data is the primary step before considering the development of signal processing or machine learning analytical tools for audio. Despite there being a shortage of singing datasets, recently, several singing datasets, such as (Cuesta *et al.*, 2018; Choi *et al.*, 2020; Rosenzweig, Cuesta, *et al.*, 2020; Rosenzweig, Scherbaum, *et al.*, 2020; Bittner *et al.*, 2021), have been published. However, many more datasets are needed representing different categories, such as techniques, genres, countries, traditions, and languages.

One recently released dataset that covers a wide range of singing techniques and expressions is VocalSet (Wilkins *et al.*, 2018). This dataset is already described and used in the previous chapters.

Although the VocalSet offers a variety of audio files of singing, the notes sung by the singers were not annotated. Therefore, the original VocalSet is a suitable resource for evaluating machine learning algorithms for problems such as singer classification, vowel classification, singing technique classification, and melody classification. Nevertheless, to use the VocalSet for other purposes, such as pitch detection, pitch contour smoothing, onset/offset estimation, note extraction, lyric estimation, and automatic transcription, it would be necessary to annotate it. Therefore, this study aims to annotate the original VocalSet to render it appropriate for the possible additional studies mentioned above. The annotations added to the VocalSet include the fundamental frequencies (F0), amplitude, onset, offset, transition, MIDI pitch, average/median F0 of each note, each note's duration, and the lyric.

Table 2-2. A comparison of existing singing datasets.

Dataset	Solo/ Mix	No. of Files	Total Duration (min)	Annotations	Scripts	Number of Singer	Professional/Amateur
MIR1K (Chao-Ling Hsu and Jang, 2010)	Both	1000	133	F0, unvoiced sounds and vocal/non-vocal segments, lyrics	No	8 female (f), 11 male (m)	Amateur
TONAS (Mora *et al.*, 2010; Gómez and Bonada, 2013; Team, 2013)	Solo	72	20.6	F0, onset, note F0	No	>40	Professional
SVNote1 (Hoon Heo, Dooyong Sung and Kyogu Lee, 2013; Chang and Lee, 2014)	Solo	30	16.6	Onset, offset, MIDI pitch	No	7 m, 3 f	NI *
Evaluation Framework (Molina *et al.*, 2014)	Solo	38	19.2	MIDI pitch	No	8 Child, 8 m, 5 f	Both
iKala (Chan *et al.*, 2015)	Solo	252	126	F0, lyrics	No	6	Professional
MedleyDB (Bittner *et al.*, 2014, 2016)	Both	28	255	F0, meta data	No	NI	NI
MASTmelody (Bozkurt, Baysal and Yüret, 2017)	Both	1018	90	F0	Yes	NI	Pupils
Dzhambazov (Dzhambazov *et al.*, 2017)	Solo	13	7	F0, amplitude, note MIDI pitch	No	NI	NI
Choral Singing (Cuesta *et al.*, 2018)	Choir	48	115.5	MIDI file	No	16	Semi-professional
VocalSet (Wilkins *et al.*, 2018)	Solo	3560	606	NA +	Yes	11 m, 9 f	Professional
CSD (Choi *et al.*, 2020)	Solo	200	291.7	Onset, offset, lyric, MIDI pitch, MIDI	No	1 f	Professional
Dagstuhl ChoirSet (Rosenzweig, Cuesta, *et al.*, 2020)	Choir	81	55.5	MIDI, F0, beats	No	NI	Amateur
Erkomaishvili (Rosenzweig, Scherbaum, *et al.*, 2020)	Solo	101	424.5	F0, segmentation, onset, lyric	Yes	1 m	Professional
Vocadito (Bittner *et al.*, 2021)	Solo	40	13.62	F0, lyric, note	No	29	Varying levels of training
DALI (Meseguer-Brocal, Cohen-Hadria and Peeters, 2018)	Mix	5358	NI	Note, lyrics	NA	NI	Amateur
Annotated-VocalSet	Solo	2688	406.7	F0, onset, offset, note, lyric, MIDI pitch	Yes	11 m, 9 f	Professional

* NI = Not indicated, + NA = Not applicable.

2.2.4.1 A Review of Published Vocal Datasets

Several singing datasets have been published and made available to researchers. Table 2-2 lists these singing datasets with their properties, such as the number of files, total duration, and annotations. As shown in Table 2-2, the Annotated-VocalSet dataset, which I have generated and will be discussed in detail in Chapter 7, includes a broader range of annotations and singers than the listed datasets. However, some other datasets include properties, such as the singer's amateur status or whether they are currently studying, which the Annotated-VocalSet does not have.

2.2.4.2 A Review on Annotating Methods

This subsection reviews the other researchers' approaches to annotating vocal datasets. Generally, they use three main annotation approaches: manual, automated, and semi-automated, as described in the following.

2.2.4.3 Manual Annotation

Some datasets, such as MIR-1K (Chao-Ling Hsu and Jang, 2010), iKala (Chan *et al.*, 2015), Dzhambazov (Dzhambazov *et al.*, 2017), and Erkomaishvili (except for the F0 annotation) (Rosenzweig, Scherbaum, *et al.*, 2020), were annotated manually. To reduce human errors in creating annotations, some researchers, such as TONAS (Team, 2013), after generating annotations by the first person, asked some experts to double-check the initial annotations. In addition, other researchers, such as the providers of the SVNote1 dataset (Hoon Heo, Dooyong Sung and Kyogu Lee, 2013; Chang and Lee, 2014), asked more than one person to generate manual annotations. The Erkomaishvili dataset (Rosenzweig, Scherbaum, *et al.*, 2020) used Sonic Visualiser (Cannam, Landone and Sandler, 2010) to include the onset annotations manually. In addition, they added the musical scores along with the lyrics by hand with the aid of the software tools named Finale ('Finale', no date) and Sibelius ('Sibelius', no date).

2.2.4.4 Automatic Annotation

The fundamental frequencies in the MedleyDB dataset (Bittner *et al.*, 2014, 2016) were annotated with the PYIN algorithm (Mauch and Dixon, 2014).

To annotate the Choral Singing dataset (Cuesta *et al.*, 2018), they used the spectral autocorrelation (SAC) method proposed by Villavicencio et al. (Villavicencio *et al.*, 2015) to

estimate F0, and they mentioned that the result of the SAC method contained some errors. Therefore, to calculate the mean of the notes' pitch frequencies, they considered a threshold on the F0 values obtained for each note before computing the average. The threshold was set to 60 cents to reduce the effect of outliers. Moreover, they used a Python library, pretty_midi (Raffel and Ellis, 2014), to extract the note onsets and offsets from the synchronised MIDI files and segmented the F0 array.

The DALI dataset (Meseguer-Brocal, Cohen-Hadria and Peeters, 2018) was generated using the teacher-student machine learning paradigm. They synchronised the audio files from karaoke games with lyrics and notes by applying machine-learning techniques to assist them.

2.2.4.5 Semi-Automatic Annotation

The fundamental frequencies in the Erkomaishvili dataset (Rosenzweig, Scherbaum, *et al.*, 2020) were automatically computed within the user-specified regions using an F0 estimation algorithm similar to Melodia (Salamon and Gomez, 2012). Then, the annotator could guide the estimation process. Moreover, the tool's audiovisual feedback mechanisms helped the annotator validate and correct the computed F0-trajectories.

For annotating F0 in the MASTmelody dataset (Bozkurt, Baysal and Yüret, 2017), a software tool, Melodia (Salamon and Gomez, 2012), was used. Then, since the pitch contours were not error-free, they manually altered the pitch contours.

To manually annotate beats in the Dagstuhl ChoirSet dataset (Rosenzweig, Cuesta, *et al.*, 2020), the Sonic Visualiser tool (Cannam, Landone and Sandler, 2010) was employed. An expert annotator corrected the annotations that a non-professional musician had created. To synchronise notes with MIDI files, they employed the DTW approach presented by (Muller, Kurth and Röder, 2004; Ewert, Muller and Grosche, 2009), using the beat annotations as anchor points for the alignment. Therefore, they had each note's onset, offset, and MIDI pitch after the synchronisation. Regarding F0 annotation, they applied the PYIN (Mauch and Dixon, 2014) and CREPE (Kim *et al.*, 2018) to estimate them. In addition, they also used a tool, Tony (Mauch *et al.*, 2015), to edit pitch contours manually.

Similarly, the F0s in the Vocadito dataset (Bittner *et al.*, 2021) were estimated by Tony (Mauch *et al.*, 2015), which is based on the PYIN (Mauch and Dixon, 2014) algorithm, and then an expert edited the estimated pitch contours. They used a similar approach for extracting

notes; Tony was used first to estimate the onset and offset of the notes. Then, two experts corrected any errors identified in these extracted notes. Finally, for the lyrics, they manually added the words that the singers sang without considering the timing of the words.

As can be seen from the related work in annotations, the current automatic tools are not error-free, and humans need to review their results and alter any incorrect annotations. Thus, we used the same approach; first, a tool automatically annotated the dataset, and then manual intervention was used to alter any incorrect annotations.

2.2.5 Calculating notes' pitch frequencies and duration according to singing technique and their positions in a piece of music objective

Based on the psychoacoustic studies (Bjørklund, 1961; Seashore, 1967; Sundberg, Prame and Iwarsson, 1995; Dalla Bella *et al.*, 2007; Stables, Athwal and Bullock, 2011; Sundberg, 2011, 2013; Sundberg, Lã and Himonides, 2013; Mauch, Frieler and Dixon, 2014) the perception of intonation and duration of a note are affected by how the brain processes sound. It could be the case that a subtle drift in the previous or the following notes can influence the acceptable ranges within which the note is judged to be correct by the listener.

Sundberg et al. (Sundberg, Prame and Iwarsson, 1995) studied what mean F0s were accepted as being "in tune" and "out of tune". The results showed that most of the tones deemed to be in tune had an average F0 that varied within a narrow band of about ±7 cents, whereas most tones judged as being out of tune were outside this frequency band. Moreover, Sundberg et al. (Sundberg, Prame and Iwarsson, 1995) found that singers exhibited the same patterns of changing intonation when performing the same notes but used slightly different frequencies when they repeated these notes in other bars, that is, pitch drift.

According to the Seashore (Seashore, 1967) study, long notes were sung with an average F0 that coincides with the theoretically correct value. Moreover, many long tones changed their average frequency in various ways during the performance of the tone. Bjørklund (Bjørklund, 1961) found that such deviations were typical for professional singers as opposed to nonprofessional singers. With regard to short tones, the relationship between F0 and the theoretical pitch seems to be considerably more complicated (Sundberg, 2013).

Sundberg & La (Sundberg and La, 2011) analysed the tuning of premier baritone singers and found examples of significant deviations from equal-tempered tuning (ETT), sometimes exceeding 50 cents. In particular, the highest note in phrases with an agitated emotional character was often sharpened. The intonation of such tones was flattened to equal-tempered tuning, and a listening test was run in which musician listeners were asked to rate the expressiveness in a pair-wise comparison between the original version and the version with manipulated tuning. There was a significant preference for the original versions. This result indicates that intonation can be used as an expressive device in singing.

Based on the above explanation, researchers have found that the amount of allowable/imperceptible frequency deviation in each sung note depends on its position in a piece of music. However, because of some limitations in previous studies, the exact acceptable ranges of note's pitch and duration in a piece of music are not fully understood. Therefore, after describing a newly created annotated singing dataset in Chapter 7, the dataset will be examined in Chapter 8 to discover the genuine relationship regarding the acceptability between the perceived F0 and duration of a note against its theoretical frequency and duration. As a result, Chapter 8 provides a novel algorithm to calculate the acceptable range of pitch frequencies and duration of each note based on its position in a piece of music.

Theoretically, each western music note has an exact pitch frequency and duration. However, playing a note without pitch frequency and duration instability is practically impossible. This issue is more challenging when the human voice sings musical notes. Because, unlike musical instruments, it is impossible to tune the notes that the human voice is going to produce before the performance. In other words, although a singer does the tuning, it is down to the singer's skills to identify the correct tuning and then hold a steady tuning over the duration of the note to be sung. In addition, using the same physiological system for breathing and singing simultaneously, and how the humans' brain perceives sounds bring more limitations and complexity for a singer to play notes. In singing, subglottal pressure must be tailored to both pitch and loudness. Since a change in subglottal pressure results in a change in fundamental frequency, singers should accurately reach the target subglottal pressures (Sundberg, 1992). Another issue that adds more complexity to in-tune singing is how the human brain perceives sound. Several psychoacoustic studies, such as (Bjørklund, 1961;

Seashore, 1967; Sundberg, Prame and Iwarsson, 1995; Dalla Bella *et al.*, 2007; Stables, Athwal and Bullock, 2011; Sundberg, 2011, 2013; Sundberg, Lã and Himonides, 2013; Mauch, Frieler and Dixon, 2014), showed that the perception of performed F0 and the duration of a note is impacted by the brain's sound processing mechanism (Faghih and Timoney, 2019b). Therefore, it is understood that singers should sing a note in an acceptable range of F0s and duration.

Although several studies, as listed below, have been conducted to define the perceptually acceptable performed F0s and duration ranges, they still have not precisely identified the ranges according to a note's position in a piece of music. Thus, this study's objective is to define these ranges more accurately by considering the effect of some variables altogether.

According to Seashore (Seashore, 1967), the musical ear is generous and operates in the interpretive mode when it listens to singing. However, there are certainly limits to this generosity. Also, what appears to be generosity may be sensitivity to small, deliberate, and meaningful deviations from what theoretically is "correct". For example, the human hearing system can probably not track swift changes in pitches (Sundberg, 1972). For example, Moore's study (Moore, 2013) indicated that the human ear could not distinguish between two transients less than 10 ms apart.

There is another challenge in defining singing in tune. As discussed in Sundberg's study (Sundberg, 2013), the challenge is that the pitch frequency bands corresponding to tones perceived as being in tune did not always agree with the notes of Equal-Tempered Tuning (ETT). Each octave is divided into equal interval steps in the ETT system. In the 12-ETT system, each octave is divided into 12 steps, and each one is called a semitone.

Moreover, Sundberg (Sundberg, 2013) reported that for some tones, the mean performed F0 accepted as being in tune was shown to vary wildly among expert listeners. These tones seemed to be harmonically (simultaneously) or melodically (sequentially) marked. Most singers seemed to adhere to certain principles in their deviations from the ETT. One was to sing high tones sharp, adding an F0 correction that increased with pitch. The other was to sharpen and flatten, respectively, the tones that were situated on the dominant (right)

and subdominant (left) side of the circle of fifths, where the root of the prevailing chord was the “12 o’clock” reference.

Sundberg et al. (Sundberg, Prame and Iwarsson, 1995) studied what mean performed F0s were accepted as being “in tune” or “out of tune” in 10 commercial recordings of a song that were presented to expert listeners. The results showed considerable variability in the judgments. Analysis of the tones accepted as being in tune by all experts or deemed out of tune by most listeners revealed that most of the tones deemed to be in tune had an average performed F0 that varied within a narrow band of about ± 7 cents. In addition, most tones judged as being out of tune were outside this frequency band. Moreover, they found that singers exhibited the same patterns of changing intonation when performing the same notes.

Nevertheless, they mentioned that singers used slightly different frequencies when they repeated these notes in other bars. They finally concluded that the deviation from the ETT is not the sole correlate of out-of-tune perception. However, because of insufficient samples, they could not definitively conclude under which circumstance of a particular note occurring in a piece of music a singer would sing that note in a lower or higher pitch. Furthermore, their study also exhibited some other limitations: firstly, all of their notes are in only one octave, between D4 and D5. Secondly, the positions of the notes were not considered in their evaluation. With these limitations, their result of ±7 cents cannot be considered as a precise range in all cases. It is quite possible, for example, that the behaviour for higher or lower pitches is different. This could also apply to notes with a shorter or longer duration.

As a counterexample to the ± 7 cents finding by Sundberg et al. (Sundberg, Prame and Iwarsson, 1995), Sundberg and La (Sundberg and La, 2011) analysed the tuning of premier baritone singers. They found examples of quite large deviations from ETT, sometimes exceeding 50 cents. In particular, the highest note in phrases with an agitated emotional character was often sharpened. The intonation of such tones was flattened artificially by using a signal processing algorithm to make it comply with ETT. Then, a listening test was run in which musician listeners were asked to rate the expressiveness in a pair-wise comparison between the original version and the version with manipulated tuning. There was a significant preference for the original versions. This result indicates that intonation can be used as an expressive device in singing.

Another study that showed that the pitch frequency of an individual note may change according to the note position is Sundberg et al. research (Sundberg, Lã and Himonides, 2013). They observed that a professional singer might sharpen tones sung in ETT, sometimes even more than 50 cents. They found that most listeners failed to realise the intonation differences as a pitch effect. They concluded that such sharpening might contribute to expressiveness but that listeners did not appear to perceive pitch as a guide by which they could rate expressiveness. In addition, the expert listeners perceived the original versions as more expressive than those in which the intonation exactly followed equally tempered tuning.

According to the Seashore study (Seashore, 1967), long notes were sung such that the average of the performed F0 coincides with the theoretically correct value. However, the long tones are often slightly flat (approximately 90 cents on average) at the beginning and then gradually corrected during the initial 200 milliseconds of the tone. Moreover, many of the long tones changed their average frequency in various ways during the tone. Bjørklund (Bjørklund, 1961) found that such deviations were typical for professional singers as opposed to nonprofessional singers. As an example to compare professional singers to untrained singers, Sundberg (Sundberg, 1979) examined the maximal speed of voice pitch changes in professional and untrained singers of both genders. He found differences between these four groups (male-professional, female-professional, male-untrained, and female-untrained). It was observed that professional singers change pitch more quickly than untrained subjects on average. The same reflection is made regarding female subjects as compared with male ones. In addition, it has been discovered that untrained singers perform pitch drops significantly faster than pitch elevations.

Sundberg (Sundberg, 2013) found that the relationship between the performed F0 and the theoretical pitch in short tones seems to be considerably more complicated. He observed that each short note takes one vibrato period, and most of the vibrato periods seem to encircle the target frequency approximately.

Bottalico et al. (Bottalico, Graetzer and Hunter, 2017) discussed that pitch inaccuracy in singers is affected by the level of training, the tempo, articulation, semi-phrase direction (ascending or descending), tessitura (low, medium, or high), and the level of the external auditory feedback. According to their study, the mean pitch inaccuracy was between 13 and

58 cents. The worst case was detected for the nonprofessional singers in staccato, fast arpeggio in the high tessitura, and in normal external auditory feedback conditions; and the best case for the professionals was in legato and slow arpeggio in the high tessitura.

Another singing feature that affects pitch frequencies is the singer's physical gestures. For example, in the study by Brunkan and Bowers (Brunkan and Bowers, 2021), most solo singers tended toward more in-tune singing while employing the pointing gesture, whereas most participants became progressively more out of tune with the low in a circular gesture, arms moving outward and upward in front of and to the side of the torso at hip height and above. Similarly, in the study by Manternach (Manternach, 2016), it was discussed that the conductor's movements also affect the amplitude and fundamental pitch frequencies of the songs sung by singers.

Based on the above explanation, researchers have found that the amount of pitch deviation by singers in each sung note depends on their position in a piece of music, the singer's gesture, expressiveness, audience feedback, and the conductor's movement. However, since these studies have some limitations, the exact explanation of singers' behaviours in performing pitch and duration of a note in a piece of music are not identified. Therefore, Chapter 8 investigates a dataset of recorded vocals to discover some particular aspects of the relationship between the performed F0 and duration against its written note and relative duration in a music score. In other words, this chapter introduces two novel models to simulate trained-professional singers' behaviours in singing notes' pitches and duration according to the position of the note in a piece of music and the singing technique applied. Because of the limitations on available annotated singing datasets, some of the note's features that might affect trained-professional singers to change the pitch and duration of a note were investigated in this study. Therefore, the note's features considered for this research are the note's MIDI pitch code and duration in a music score, the pitch intervals to the following and previous notes, the existence of a rest before or after a note, the signing techniques, and whether the note is a repeat.

2.3 Applications

There are some common steps in most of the singing applications, as mentioned in section 1.3. These steps are estimating pitch contour, pre-processing, extracting and calculating notes' pitches and duration, alignment, and analysing the estimated notes. The objectives of this thesis can improve the state-of-the-art algorithms/tools in each of these steps. First, chapters 3 and 4 will evaluate several pitch detection algorithms to find the more accurate ones according to the features of the signing signals. Then, Chapter 5 will introduce a new contour smoother algorithm as a pre-processing. In addition, a novel algorithm for estimating onsets will be introduced in Chapter 6. Finally, two models for estimating the expected ranges of a note's pitch and duration in trained-professional singers will be proposed in Chapter 8. These findings can be applied to different applications.

This section provides a list of possible applications of the results of this PhD study. These applications include alignment, singing assessment, automatic tuning of singing, singing imitation synthesis, and automatic singing transcription.

The following section, 2.3.1, mainly lays out the techniques used in the applications discussed after that.

2.3.1 Aligning sung notes with ground truth

Several applications, such as score following and singing assessments, require alignment. For example, to assess a singing performance, the user's notes should usually be aligned with the ground truth before the assessment. The ground truth can be a musical score, a sound recording, or a mixture of both with/without accompaniment. It should be mentioned that in the alignment, both the duration and pitch of a note should be considered simultaneously.

Some well-known alignment algorithms, such as DTW and HMM, will be explained in the following.

2.3.1.1 DTW (Dynamic Time Warping)

Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions. Then, the sequences are warped nonlinearly to match each other (Müller, 2007).

An (*N*, *M*)-warping path is a sequence $p = (p_1, \dots, p_L)$ with $p_l = (n_l, m_l) \in [1{:}N] * [1{:}M]$ for $L \in [1{:}L]$ satisfying the following three conditions (Müller, 2007).

1- Boundary condition: $p_1 = (1,1)\ and\ p_L = (N, M)$

2- Monotonicity condition: $n_1 \leq n_2 \leq \cdots \leq n_L\ and\ m_1 \leq m_2 \leq \cdots \leq m_L$

3- Step size condition: $p_{L+1} - p_L \in \{(1,0),(0,1),(1,1)\} for\ L \in [1{:}L-1]$

In addition, every index from the first sequence must be matched with one or more indices from the other sequence and vice versa. Figure 2-2 depicts these conditions in DTW. This figure illustrates the paths of index pairs for some sequence X of length N =9 and some sequence Y of length M =7. In Figure 2-2(a), all the above conditions are satisfied, while in panels b, c, and d, the boundary, monotony, and step size conditions are unsatisfied, respectively.

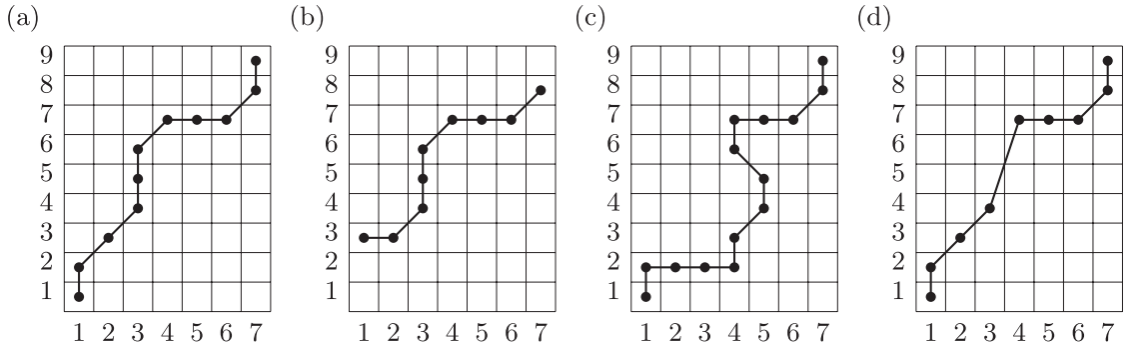


Figure 2-2 Illustration of paths of index pairs for some sequence X of length N =9 and some sequence Y of length M =7. (a) Admissible warping path satisfying the conditions (1), (2), and (3). (b) the boundary condition is violated. (c) the monotony condition is unsatisfied, (d) the step size condition is disregarded (Müller, 2007)

The optimal match is denoted by the one that satisfies all the restrictions and rules. In addition, the optimal match should have a minimal cost, where the cost is computed as the sum cumulative of absolute differences, for each matched pair of indices, between their values (Müller, 2007).

There are two ways that DTW can be employed for singing alignment.

a) Converting the musical notation (such as a MIDI file) to a sequence of frequencies and then comparing the user voice with the frequencies from the MIDI file using DTW. However, an attempt, as an example, to implement this procedure did not lead to a good result based on the experience of Molina et al. (Molina et al., 2013).
b) Converting the user voice to MIDI values and then comparing it with the original MIDI file.

Nevertheless, in real singing situations, the user may sing some extra notes or make articulations that are not in the score. Additionally, they may miss some notes. In these circumstances, it will be impossible for DTW to assess the similarity accurately.

Another problem identified with using DTW in this context is that although it may be possible to create an alignment between the frequency contour of the sung notes with the ground truth, it cannot inherently know the thresholds at which each sung note pitch and duration should be considered as correct. For example, DTW can tell us that the difference frequency between two notes is 50 hertz, but it cannot determine that the user performed the sung pitch in a perceptually correct manner. This is the same situation for the sung note duration. In practice, systems based on DTW (Dong *et al.*, 2010; Molina *et al.*, 2013; Schramm, Nunes and Jung, 2015; Gupta, Li and Wang, 2017; Luo *et al.*, 2018) have been recognised not to be useful for singing assessment purposes, because the feedback of these systems is just a number that indicates the minimum distance between user performance and the target melody. However, as it is discussed in Chapter 8, there is a range of allowable pitch and duration for each note according to its position in a piece of music.

Molina et al. (Molina *et al.*, 2013) used the following cost formula of the DTW as given in equation (2-1) in order to have a better result:

$$M_{ij} = \min\{(f_{0\,T}(i) - f_{0U}(j))^2, \alpha\} \quad (2\text{-}1)$$

where $f_{0\,T}(i)$ is the fundamental frequency (f_0) value of the target melody in the frame i, $f_{0U}(j)$ represents the f_0 value of the user's performance in the frame j, M_{ij} is the cost value, and α is a constant user-defined threshold.

When the squared f_0 of the difference becomes larger than α, it is assumed that a spurious case has been found and its contribution to the cost matrix is limited. In Molina et al. study (Molina *et al.*, 2013), a discussion is made on limiting the DTW path between 10 and 80

degrees for intonation, but unfortunately, the exact value of α is not provided, nor is a formula by which it can be calculated.

In the system by Schramm et al. (Schramm, Nunes and Jung, 2015), after obtaining musical scores from the user performance by employing an automatic melodic transcription algorithm, an alignment algorithm is used to assess the similarity between the ground truth and the user performance. They presented a different alignment algorithm which is very similar to DTW. However, their algorithm does not propagate the cumulative error since it does not need to obey the boundary conditions of the DTW. When they tried to find a method to match segments from the user performance with the ground truth, they discovered that the sequence of the notes performed by the user could be determined to be similar to the sequence of notes in the ground truth, but it will not consider any possibility of missing or extra notes that could exist in the user's sequence. Therefore, a combination of correct and incorrect notes in close proximity in the user sequence could be adjusted by the warping algorithm to a smaller size set/sequence in the ground truth that contains the correct notes only. Thus, the combined duration in the sung version would be shrunk by warping (and very likely with some level of distortion created) to fit the ground truth. Figure 2-3 provides an example of this grouping and alignment process in which six segments in the lower part of the figure, with a grey fill, are mapped into three notes, the upper part of the figure outlined in green (Schramm, Nunes and Jung, 2015). As can be seen in Figure 2-3, this alignment algorithm cannot identify the extra notes correctly because the algorithm simply combines them to make a fit with the one note in the ground truth.

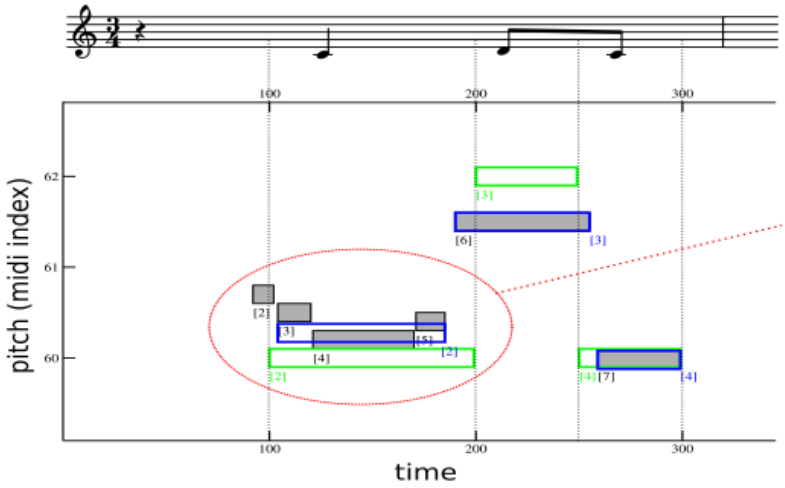


Figure 2-3 Grouping process of several segments (grey) into one music note (blue) (Schramm, Nunes and Jung, 2015)

Luo et al. (Luo *et al.*, 2018) used Canonical Time Warping (CTW), which is very similar to DTW. They tried to find an alignment between the user performance and a professional version of that piece of music. Therefore, the result of this algorithm is the same as DTW, but it is only a number that indicates the distance between two sounds. This kind of output cannot be useful for users who want to understand and correct their singing errors.

In the Dong et al. (Dong *et al.*, 2010) study, they researched a method to align the singing voice with a MIDI file by using DTW. Instead of converting the user performance to MIDI, they convert the MIDI values to a frequency which is not a better choice because it still does not imitate human singing behaviour in changing pitches and duration. They used the following formula, (2-2), to convert the MIDI to frequencies.

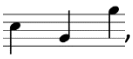
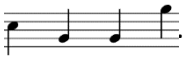
$$f = \left(2^{\frac{p-69}{12}}\right) * 440 \qquad (2\text{-}2)$$

Where f is the frequency in Hertz, and p is the midi-note value.

Although this formula generates a series of frequencies from the notes of the MIDI file, it should be recognised that the MIDI representation of the notes is very different to the intricacies of those notes created by a human voice. For example, human singing not only comprises steady notes but also pitch transitions and modulations that are not captured by digitally generated MIDI notes (Gupta, Li and Wang, 2017). In addition, as they only looked to find a method to align a correct singing performance to its corresponding MIDI file, they did not consider the possibility of mistakes in the actual performance and how to handle them with their algorithm.

Some other situations in a singing assessment are also challenging for DTW. For example, if the original note is a whole note, but the user sings eight notes in succession. From a musical perspective, this would be a terrible performance, but DTW may determine it to be not very bad. Indeed, while the duration and pitches should be considered together, DTW compares them separately. This can be illustrated in the following example:

Consider two series of notes as [440, 493.88, 523.25] (A4, B4, C5) and [440, 493.88, 493.88, 523.25] (A4, B4, B4, C5) that are input to the DTW algorithm. The DTW-determined minimum distance between the two series will be 0, implying that both series are the same.

However, in music notation, they are different. That is, the first series is  , and the second one is  .

Another problem with using a DTW-based alignment algorithm is that the minimum distance is unreliable. For example, if the target series is [261.63, 329.63, 392.00, 523.25] (pitch frequencies of C4, E4, G4, and C5, respectively) and a user omits to sing the second or the third note, the minimum distance is 8.91, but if the missing note is the last note, then the minimum distance will be 18.75. Thus, in the case of this type of mistake, the DTW result is changed significantly. However, this numerical result is not a reason to conclude that the version's performance with the minimum distance of 8.91 is better than 18.75.

Moreover, as will be discussed in Chapter 8, acceptable ranges for the note's pitch and duration need to be considered. Therefore, the sung notes' duration and pitches in DTW should be compared with the acceptable ranges instead of the ground truth values. In this case, the differences for the notes inside the acceptable boundaries should be assumed to be zero.

2.3.1.2 HMM (Hidden Markov Model)

A Hidden Markov Model is a machine learning approach that works as a state machine to model stochastic signal sources. Regarding singing transcription, the training data consists of the frame-level features extracted from recorded singings, and the training is usually performed unsupervised with the Baum-Welch algorithm (Ryynänen, 2006).

A HMM is constructed for each reference signal, in which the transition probabilities describe note insertions and deletions, repeats, and skips, while pitch errors are described by output probabilities. Then, the aligned signal is counted as an output sequence from the HMM, and the most probable sequence of latent states is assessed with the Viterbi algorithm for alignment (Nakamura, Yoshii and Katayose, 2017).

Nevertheless, HMMs cannot fully support the objectives of some applications, such as singing assessment, because of its limitations that are discussed in the following.

Based on the Nakamura et al. (Nakamura, Yoshii and Katayose, 2017) study on polyphonic piano performances, it has been discovered that deviations in performances due to asynchronies between hands/voices require special treatments. Such asynchronies result

in the reordering of notes with different score times, which is the leading cause of alignment errors for HMMs (and also DTWs) that are not specially designed to handle them (Nakamura, Yoshii and Katayose, 2017). To illustrate, consider the case where a ground truth note is A4 with a frequency of 440 Hz, but the user performs it at 425 Hz, which is somewhere between the notes of A4 and A^b4 (410.3 Hz). However, there is no note between A4 and Ab4 in western music notation. Therefore, the algorithm will try to guess the intended note as either A4 with a frequency of 440 Hz or A^b4 with a frequency of 410.3 Hz, and then associate this with the ground truth note. The missing part of these studies is that each note should have an acceptable pitch and duration range, which will be covered in Chapter 8.

2.3.1.3 Viterbi algorithm

One of the applications of the Viterbi algorithms is to make the HMM evaluation faster by reducing the number of paths. In Ryynänen's study (Ryynänen, 2006), the Viterbi algorithm and the HMM technique were integrated. The resulting method was that when the HMM parameters have been learned, the state sequence that maximises the observed data's posterior probability will be estimated using the Viterbi algorithm. An alternative state-sequence estimation scheme is the token-passing algorithm, which is designed to find the most probable path through a network of connected HMMs (Ryynänen, 2006). In another study by Mayor et al. (Mayor, Bonada and Loscos, 2006), a Viterbi Matrix was employed to find the most probable path from all possible paths. Another application of the Viterbi algorithms is finding the minimum path, which indicates the distance between the user performance and the target melody, such as in the work done by Lal (Lal, 2006).

Thus, the findings in Chapter 8 can help Viterbi algorithms to have a better alignment estimation by considering the acceptable range of pitch frequencies and duration.

2.3.2 Score following

One of the applications of this thesis's objectives is in score-following, which is a real-time alignment of the audio signals from musical performances to a given score (Nakamura, Nakamura and Sagayama, 2016). Several studies have been done on the score following, such as (Cont, 2010; Nakamura, Nakamura and Sagayama, 2013, 2016; Dorfer, Arzt and Widmer, 2017). The general approach uses a HMM to align the user performance with the music score.

According to Nakamura et al. (Nakamura, Nakamura and Sagayama, 2016), one of the issues that should be considered during the alignment is the user's errors since the audio signals associated with music performances can vary widely even if the same score is used. Nakamura et al. (Nakamura, Nakamura and Sagayama, 2016) provided four typical sources of variety in a monophonic audio performance which are the following.

(a) Acoustic variations: Spectral features of audio performances depend on musical instruments and are not stationary. In addition, audio performances usually include noise caused by the surrounding environment and musical instruments (e.g. resonance, background noise, breath noise, and other acoustics).

(b) Temporal fluctuations: The tempo of the performance, onset times, and durations of performed notes deviate from those indicated in scores due to the performer's skills, physical limitations of musical instruments, and musical expressions. For example, performances during practice are often rendered at a slow tempo to avoid errors. Moreover, a technique in music termed Rubato, associated with the interpretation of the piece, gives the player rhythmic freedom to change the speed.

(c) Performance errors/variation: Performers may make errors due to a lack of performance skills or misreading the score. Errors are categorized into pitch errors (substitution errors), dropping notes (deletion errors), and adding extra notes (insertion errors). Besides, performers may pause between notes, for example, to turn a page of the score and check the next note. Furthermore, for expression, this is the activity of interpreting a score that every musician must do and distinguishes each musician's approach to a piece of music.

(d) Repeats/skips: Performers may repeat and/or skip phrases during practice. Furthermore, the performers generally add or delete a repeated section.

Most of the studies on score following algorithms have not resolved these errors entirely (Orio, Lemouton and Schwarz, 2003; Schwarz, Orio and Schnell, 2004; Pardo and Birmingham, 2005; Nakamura *et al.*, 2014). However, one of the well-developed systems that considered these errors is the study by Nakamura et al. (Nakamura, Nakamura and Sagayama, 2016). They used two HMMs, one to find the probability of the sequence of notes and the other for figuring out the name of the note of the current signal. Still, there are some limitations to their

approach. As they mentioned, “the accuracy of score following generally depends on the parameters of the emission probabilities”. In addition, “the parameters can be learned from every musical instrument if necessary data is available, and we can form a detailed model for a specific instrument” (Nakamura, Nakamura and Sagayama, 2016). They evaluated their system for clarinet, and the substitution errors were restricted to three types typical in clarinet performances: errors in semitone, whole-tone and perfect 12^{th}. The first two errors are often caused by fingering and misreading the score, and the last error is caused by overblowing on a clarinet. Their proposed system could be evaluated with a human voice, and the errors that often happen during singing should be considered instead. In addition, they compared their algorithm with another tool known as Antescofo (Cont, 2010) and showed that their algorithm could find errors better than Antescofo, but still, their algorithm could not identify all the errors.

The objectives of this thesis can help with better note estimation and also alignment for the score following application. The central requirement of the score following algorithms is estimating the sung notes. As discussed in 2.3.4, the state-of-the-art note estimator algorithms have some errors in estimating notes, especially from singing signals. Thus, after selecting a more accurate algorithm to estimate pitch contour according to Chapters 3 and 4, and then smoothing it based on the proposed pitch contour smoother algorithm in Chapter 5, the notes can be estimated more accurately by the onset detection algorithm discussed in Chapter 6. Finally, the alignment algorithms can benefit from the models proposed in Chapter 8, as discussed already in Section 2.3.1.

2.3.3 Singing Assessment Systems

Enhancing the singing assessment systems is other application of this thesis's goals. Generally, we can divide the current singing assessment systems into two categories: (1) Entertainment and (2) Educational.

2.3.3.1 Entertainment

In the commercial marketplace, there are several applications for judging singing. Their general idea is the same: comparing the user's performance with the target melody and then giving a score to the user based on the similarity between the user's performance and the target melody. For example, Lal (Lal, 2006) presented an entertainment application which

compares user performance with the target melody and finally gives the user a score between 0 to 10 to indicate how much the user performance was similar to the target melody. Another example is Singstar [1], a game app based on the PSx console. In this app, when a piece of music is played, its user/users should sing its lyric. Users can see which notes they sang that had the correct pitch, duration, and onset displayed on the screen. Finally, the user with the highest score is the winner. A screenshot of Singstar can be seen in Figure 2-4. At the bottom of Figure 2-4, the blue part, the performance of one of the users, can be seen and at the top, the red part, the performance of another user. On the right side of Figure 2-4, the score of each user is presented.

Figure 2-4 A screenshot from Singstar application [www.gamestop.ie]

Ultrastar [2] can be considered as another example, which is very similar to Singstar. It should be mentioned that none of these tools compares user performance with musical scores, but they compare it with the original singer's version, which is a sound recording and not transcribed. In addition, their purpose is not to teach singing but to be a fun activity for its users.

[1] Available online at https://www.singstar.com. Accessed on 01/11/2022.
[2] Available online at http://ultrastardx.sourceforge.net/. Accessed on 01/11/2022.

2.3.3.2 Education

Lin et al. (Lin *et al.*, 2014) introduced a system that evaluates user singing performance with a musical score and gives the users three scores to assess pitch, rhythm, and total score. However, the primary drawback of their system is its use of DTW for evaluation, as critiqued in 8.2.1.1.

WinSingad (Howard *et al.*, 2004) is a software to display the pitch contour, spectral ratio, and spectrogram to the user in real-time. Therefore, since the user receives immediate feedback, they can try to fix any problems that arise. In this system, the user's role is to analyse their performance based on the pitch contour shown on the screen and then identify and fix their problems. In other words, WinSingad does not offer any performance insights or clever evaluation algorithm; it simply depicts the F0 of the singing performed by the user. To estimate F0, they used the algorithm introduced by Gruenz and Schott (Gruenz and Schott, 1949), and then with several post-processing stages, they tried to correct any incorrectly estimated pitches. Unfortunately, they did not provide an evaluation of their pitch contour estimator's accuracy, and their codes or detailed algorithm were not outside evaluation. Thus, the generated pitch contour's accuracy with their software is unclear.

Another software is Sing & See [1], which displays the user's pitches in real-time, as seen in Figure 2-5. This software also provides the spectrogram. Unfortunately, this software does not evaluate user performance to determine users' errors, but users must find their problems by examining the pitch contour drawn on the screen. MiruSinger (Nakano, Goto and Hiraga, 2007) is similar to WinSingad and Sing&See, again looking to display the pitches performed by the user without analysing and comparing user performance with the ground truth.

[1] Available online at http://www.singandsee.com. Accessed on 01/11/2022.

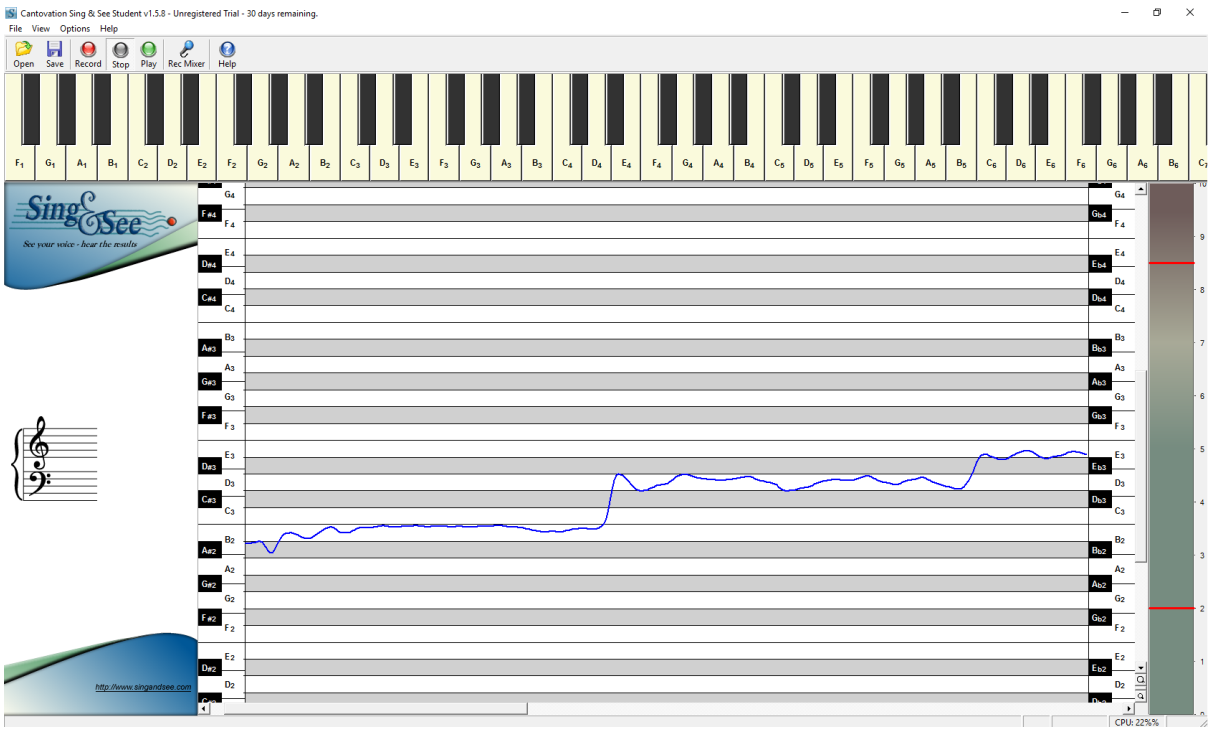


Figure 2-5 A screenshot of Sing&See software

One of the other related works is (Molina *et al.*, 2013). This looks to find a different approach to singing voice assessment. Instead of using the musical score as the target melody, they asked a professional singer to sing their target melody in order to have a set of human ground-truth signals against which they could assess users' performances. They used the mean of the intonations and rhythm (onsets) to discover the similarity between the user performance with the target melody. This appears to be the reason that their system was found to give better results in comparing a user's performance with the target melody. However, there are some weaknesses in their proposed system too. The main weakness is that their system is based on a variant of DTW (by considering F0) and note-level similarity.

Generally, in automatic singing assessment, it is essential to discover each of the user's errors and highlight them individually to the user so that they can figure out the cause of each one. In other words, the notes should be extracted and then compared to the ground truth, and finally, each note's accuracy should be determined according to the note's position in the melody.

Mayor et al. (Mayor, Bonada and Loscos, 2006; Mayor, Oscar., Bonada, Jordi., Loscos, 2009) introduced an algorithm for the tutoring of singing. Their main goal was to accurately segment the user performance based on untrained HMMs with probabilistic models built out

of a set of heuristic rules. In addition, they used Viterbi to find out the best path of the input sequence according to the reference MIDI score. An overview of their system can be seen in Figure 2-6.

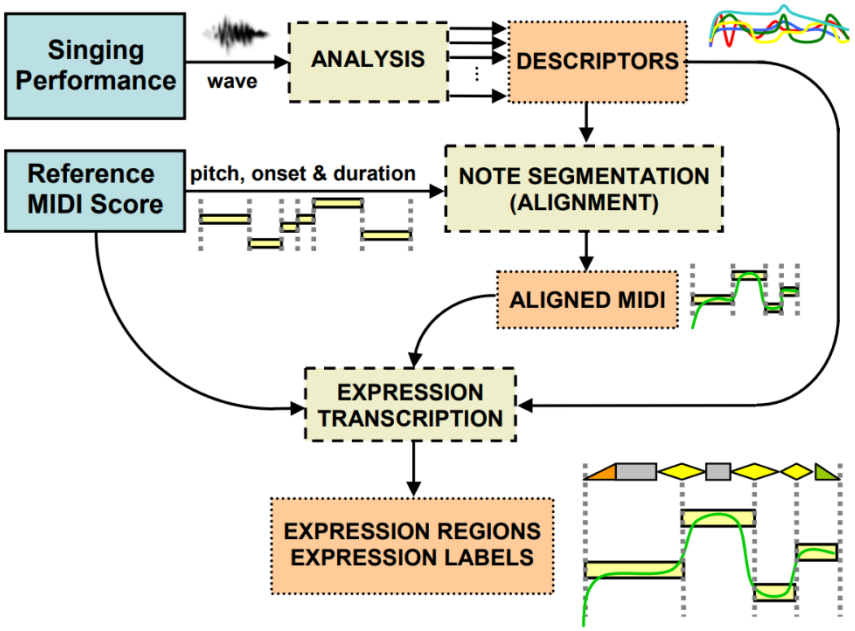


Figure 2-6 Overview of the analysis and note segmentation/expression transcription process proposed by Mayor et al. (Mayor, Bonada and Loscos, 2006)

However, there are some issues with their system:

1- They used a tolerance of 30ms for the segmentation and considered boundaries within this margin correct. However, they did not include the source of the evidence from which they selected this value of 30ms. The boundaries for the notes of different duration should most likely be quite variable, as discussed in section 2.2.5.

2- Their method is first based on the transcription of the user performance, which is then compared with the reference MIDI score. The difficulties with this approach are highlighted in section 2.3.1.

3- Although they mentioned that their segmentation was correct to a value of 95% over 1000 notes, they did not make their test cases available, so it is impossible to find out under which conditions their systems can work to an accuracy of 95%.

4- In addition, regarding timing/tempo, they indicated that the "begin time and duration should be close to the MIDI note onset and duration, respectively. Each note must have a minimum duration. The beginning and the end of the note must have a pitch. The average pitch of the note should match or be close to the reference pitch (after octave

correction), and the same should happen with intervals between two notes" (Mayor, Bonada and Loscos, 2006). However, these are vague sentences because they did not indicate how much numerically they should be close to the MIDI values.

2.3.3.3 Sight singing

Sight singing is one of the fundamental lesson activities for all music students, no matter their musical discipline. In sight singing, pupils learn how to sing a written musical score by interpreting notes' pitches, duration, and loudness in real-time. To acquire the skills to be a good singer frequently involves a teacher and requires much time for practice on one's own. The most important things to learn from the beginning are to stay 'in-tune' and 'in-time', that is, to have accurate intonation, a strong sense of rhythm, and accurate articulation (Henry, 2011). Typically, the teacher will help a student develop these capabilities using exercises and offering them musical pieces at increasing difficulty levels. It is then up to the student to practice these prior to the next lesson. However, they often find mastery of the essential skills to be one of the most difficult challenges (Henry, 2011).

Usually, sight-singing classes are held as one or two sessions per week. Therefore, a student should receive regular feedback. Nevertheless, if a student is practising his/ her lesson incorrectly and does not absorb the feedback, they may get used to singing incorrectly, which may take longer then to fix later on. Furthermore, stumbling over difficulties with no encouragement can demoralise some to the point that they can feel that they will never overcome certain musical obstacles. Therefore, having a tool that could give them immediate and detailed feedback would be extremely useful and assist them in making more rapid and confident progress (Henry, 2015; Schramm, Nunes and Jung, 2015).

In the last few years, several researchers have proposed software systems to assist the learner in their practise that can give them feedback and, thus, accelerate their progress for more serious students of music (Nakano, Goto and Hiraga, 2006, 2007; Hoppe, Sadakata and Desain, 2006; Lal, 2006; Mayor, Bonada and Loscos, 2006; Mayor, Oscar., Bonada, Jordi., Loscos, 2009; Cano, Dittmar and Grollmisch, 2011; Molina, 2012; Abeßer *et al.*, 2013; Lin *et al.*, 2014; Schramm, Nunes and Jung, 2015, 2016; Henry, 2015; Yu *et al.*, 2016; Tardón *et al.*, 2018; Gupta, Li and Wang, 2018; Luo *et al.*, 2018). The basics of these systems are pretty straightforward: The user's singing is digitized by the computer that, first of all, detects the

pitch of the notes they produce, and analyses the pitch contour to remove any algorithmic errors. Following this, an alignment is made with a ground truth, often in the form of a symbolic representation (such as MIDI) of the song melody using known techniques that include Dynamic Time Warping (DTW) (Dixon and Effects, 2005; Dong *et al.*, 2010; Devaney *et al.*, 2011; Abeßer *et al.*, 2013; Molina *et al.*, 2013; Lin *et al.*, 2014; Schramm, Nunes and Jung, 2016; Valero-Mas, Salamon and Gómez, 2015; Schramm, Nunes and Jung, 2015; Yu *et al.*, 2016; Gupta, Li and Wang, 2017, 2018; Tardón *et al.*, 2018; Luo *et al.*, 2018), Hidden Markov Models (HMM) (Mayor, Oscar., Bonada, Jordi., Loscos, 2009; Devaney *et al.*, 2011; Nakamura, Yoshii and Katayose, 2017), and the Viterbi algorithm. This process creates scores that measure the goodness of the match between the sounds produced by the users and ground truth for pitch intonation and rhythm accuracy.

One of the well-developed systems for teaching sight-singing is the study by Schramm et al. (Schramm, Nunes and Jung, 2015, 2016). They analysed user hand gestures by using a camera-based system as a timing measure instead of a computer-generated metronome. In this case, their system can evaluate users' performance based on their tempo. In addition, since, on occasion, in some music pieces, it can be necessary to change tempo and/or time signature, their system can handle these situations easily. Moreover, their system is one of the rare systems that can analyse a user's performance on a note-by-note basis; most of the other systems give an overall score only as feedback to their users. Their evaluations on the pitch, onset, and offset are based on a dataset generated by themselves and labelled by a committee of sight singing experts. They analysed the experts' labels (correct/incorrect pitch, onset, and offset) by employing Gamma probability to determine the accuracy of a user's performance. The reason for using Gamma probability is that since the experts' votes were not precisely the same, they required a way to compute the probability of the correctness of a note based on the aggregate experts' opinions. Figure 2-7 depicts their Gamma probability distributions for correct and incorrect answers based on the experts' annotations where the pitch is shown by Δf in midi scale, onset by Δs in seconds, and offset Δe in seconds. The authors of the paper mentioned their other work (Schramm, Jung and Miranda, 2015), where the Gamma probability could work successfully, and how the characteristics of their data were similar.

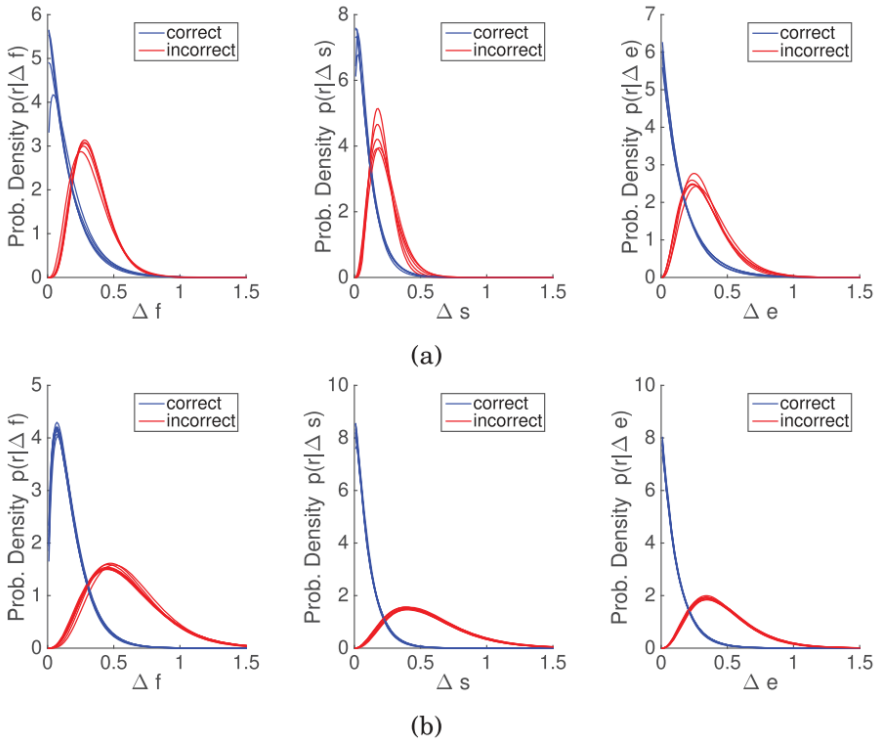


Figure 2-7 Gamma probability density functions estimated from two distinct training datasets. (a) annotated by 30 expert listeners. (b) annotated by five expert listeners. The parameters pitch, onset, and offset are labelled as correct or incorrect based on the annotation (votes) given by the experts (Schramm, Nunes and Jung, 2016).

However, there are some difficulties with their study. In particular, there is an essential question regarding the practicalities of the hardware equipment because it has a computer vision element. There is some doubt about the percentage of pupils who want to learn sight singing who would feel comfortable standing in front of a camera and moving their hand for a long time to practice their sight-singing. Also, while this kind of system may be interesting at the beginning, over time, it is not clear that which percentage of students who started using these systems for learning would like to continue their learning with this system, particularly because mastering sight singing requires several months or years of commitment.

Another issue in their study is that their method for finding the acceptable range of values for pitch, onset, and offset is unreliable, as they mentioned they did not have sufficient samples. In addition, their dataset was based on the five short pieces within which most of the notes were only in one octave, from A3 to A4. Finally, the samples in their dataset came from a small pool of just two music professors and three undergraduate students. Therefore, it could be difficult to have a high level of confidence in the strength of their evaluations. This is because the number of samples was inadequate, and only two of five performers most likely had expertise in sight singing, and we do not know about the expertise of the other three

persons. Therefore, although the reported results appear to be reasonable(Lin *et al.*, 2014; Henry, 2015; Schramm, Nunes and Jung, 2015, 2016), some issues require attention. The most important of these issues is how to align the sung notes with the ground truth, as discussed in section 2.3.1.

This thesis' aims can provide significant progress on automatic sight singing systems by explaining how to capture and edit pitch contours in Chapters 3-5, extract notes in Chapter 6, and distinguish the correctly sung note from the incorrect ones according to the findings in Chapter 8.

2.3.4 Automatic tuning of singing

Tuning singing refers to changing pitch notes to be in tune. The current approaches for auto tunning are based on comparing the sung notes with the nearest pitch frequency in an equal-tempered scale (Salazar *et al.*, 2015; Technologies, 2022), professional singers (Luo *et al.*, 2018), or the instrumental accompaniment track (Wager *et al.*, 2020). However, several concerns exist about the naturality of the sound generated by these software tools.

Therefore, the finding of this study can introduce a new approach that compares the note with the features of the previous and the following notes. Thus, the shifting note can be more natural and does not require professional song versions or instrumental accompaniment tracks.

2.3.5 Singing imitation synthesis

Signing imitation synthesis refers to generating sound by hardware or software that mimics a human performance to that the sounds are similar to human singing.

Goto et al. (Goto *et al.*, 2012) analysed only the characteristics of notes, such as pitch and amplitude, but not the position of the note in a piece of music to imitate singing.

Similarly, in the study by Zhou et al. (Zhou *et al.*, 2020), a Generative Adversarial Network was trained by extracting different Mandarin syllables. Then, their machine generated sound for XML-Music files according to the trained machine. They asked ten people to vote on the naturality of the generated sounds with their system, and used the Mean Opinion Score (MOS) to calculate the average rating of the people's opinion. The MOS was

3.12 (out of 5). Their results can be improved by considering the effect of the position of the notes in a piece of music on their pitches, duration, and loudnesses.

In the other study by Jeerapradit et al. (Jeerapradit, Suchato and Punyabukkana, 2018), they considered several features of notes such as F0, volume, and interval to the previous and following notes as well as the duration of them. However, their system can be improved by considering more features such as singing technique or the existence of a rest before or after notes.

Moreover, singing synthesizers, such as Synthesizer V[1] and Vocaloid[2], generate very accurate pitches according to the equal temperament system, but as discussed in section 2.2.5, singers change the notes' pitches according to the positions of the notes in a piece of music. Thus, these synthesizers can benefit from the findings of Chapter 8 to produce more natural sounds. As an illustration, Figure 2-8 provides a screenshot from the Vocaloid. The inputs of this software are musical scores and lyrics, and then the software will sing the lyrics based on the notes. As shown in Figure 2-8, this tool plays the sounds with absolute pitches, which is unnatural. Although articulations can be introduced, it is a complicated process and difficult as the user has to try out each one until they find something that sounds good; this could be very time-consuming as it needs to be done for every note. Thus, notes' pitches and duration must be changed according to their positions in the piece of music and the applied singing technique, as discussed in section 2.2.5.

[1] https://dreamtonics.com/en/synthesizerv/
[2] https://www.vocaloid.com/

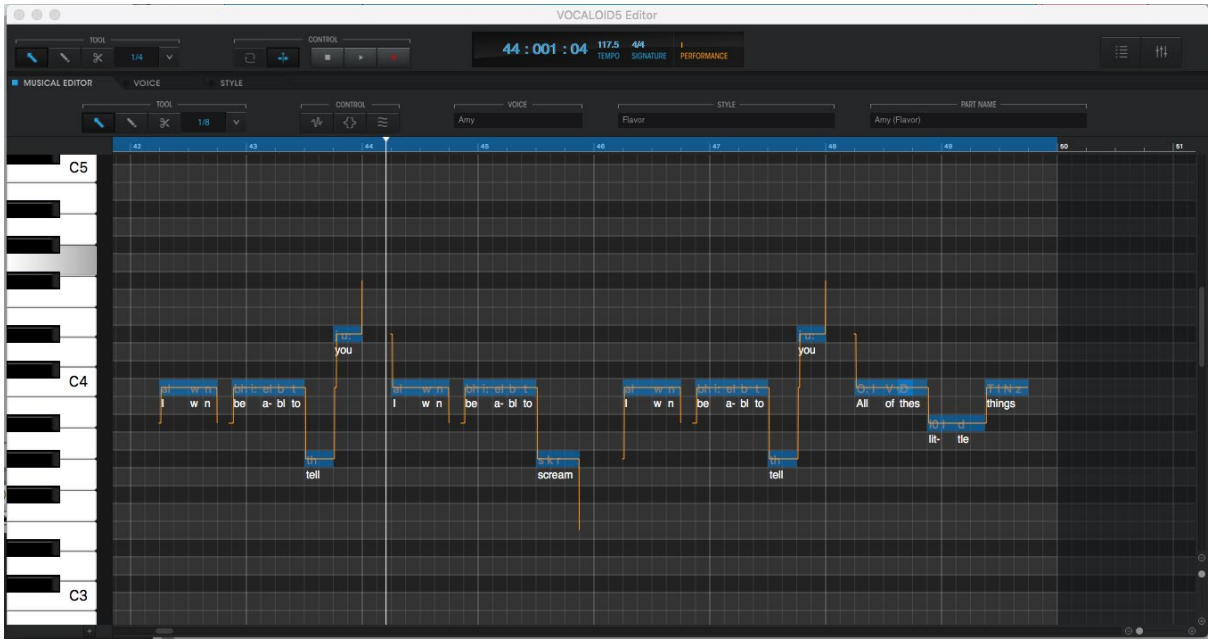


Figure 2-8 A screen shot of Vocaloid software

2.3.6 Automatic singing transcription

Automatic singing transcription refers to a computer tool that automatically converts the sung notes to musical scores. Several commercial tools are available to convert sounds to sheet music, such as Soundslice [1], Klangio [2], Melody Scanner [3], ScourCloud [4], and Sing2Notes [5]. As reported by researchers such as (Nishikimi *et al.*, 2019, 2021), one of the challenges in automatic singing transcription is estimating the corresponding note in music score for each sung note. The findings in Chapter 8 can lead to a better estimation approach for guessing the pitches and the duration of the notes. In addition, the findings in Chapters 3 to 6 can result in better approaches for estimating onsets as well as calculating pitches.

2.4 Conclusion

In conclusion, the state-of-the-art algorithms/tools have some difficulties correctly estimating pitch contour from singing signals in a real-time environment. Thus, Chapters 3 and 4 guide the selection of a proper pitch detection algorithm according to the features of the input signals. Then, they can benefit from the smoother pitch contour algorithm introduced

[1] https://www.soundslice.com/transcribe/
[2] https://klangio.com/
[3] https://melodyscanner.com/
[4] https://scorecloud.com/
[5] https://sing2notes.com/

in Chapter 5. In addition, there are some problems with extracting notes from signing signals such that the onset detection algorithm proposed in Chapter 6 would be suitable to improve this task. Moreover, in general, having more data can lead to more accurate conclusions. Thus, the annotated VocalSet dataset generated in Chapter 7 can help this objective. Finally, to analyse and synthesise singing signals, it is necessary to quantify singers' behaviour with respect to changing notes' pitches and duration according to their positions in a piece of music. The algorithms provided in Chapter 8 can help us to understand better how trained-professional singers perform the pitch and duration of a note according to its position in a piece of music and the variety of singing techniques that they could apply.

Chapter 3
Pitch detection from singing signals

This chapter aims to provide an overview on some offline and real-time state-of-the-art pitch detection algorithms and evaluate their accuracies in singing signals. This chapter provides a guidance of selecting pitch detection algorithm for singing signals. The contents of this chapter are already published in the following paper.

- *Faghih, Behnam & Timoney, Joseph, "An investigation into several pitch detection algorithms for singing phrases analysis", The 30th Irish Signals and Systems Conference (ISSC 2019), Maynooth, Ireland.*

This chapter's principal aim is to evaluate some state-of-the-art pitch detection algorithms to find the most accurate one for singing signals. The following section will introduce the dataset to be analysed and explain the difficulties associated with the pitch detection of these files. It will also detail the software framework for the analysis. Lastly, it will describe and justify the evaluation criteria. Section 3.2 will explain the pitch detection algorithms and their implementations, including the importance of the parameters associated with some of them. Then, section 3.3 will provide the results and explain each algorithm's performance. The final section, 3.4, will conclude this chapter.

3.1 Methodology

3.1.1 Dataset

We used the VocalSet dataset (Wilkins *et al.*, 2018), a singing voice dataset consisting of more than 10 hours of monophonic recorded audio of professional singers demonstrating both standard and extended vocal techniques on all five vowels. VocalSet contains recordings from 20 different singers (11 males and nine females) with a range of voice types. VocalSet has not only the full set of vowels but also a diverse set of voices on many different vocal techniques, singing in contexts of scales, arpeggios, long tones, and excerpts. For this study, the C scale and arpeggios performance of 10 males and nine females in both fast-forte and slow-forte were selected; in other words, the musical material is the same and is of a loud volume (forte), but in one case, it is sung at a quick tempo (fast) which in the second case the tempo is much slower (slow). Therefore, the total number of files used was 76.

Table 3-1 illustrates the distribution of the notes in the selected files with the total number of their reputations. As can be seen in the table, the paled notes are between C3, 130.815 Hz, and D5, 587.33 Hz. In addition, the minimum number of repetitions of the notes belongs to D5, with 18 repetitions, and the maximum is 132 for C4.

Table 3-1 The distribution and total number of repetitions of the notes played by the singers in selected files.

Note	Pitch frequency (Hertz)	Number of repetitions
C3	130.815	80
D3	146.83	40
E3	164.815	80
F3	174.615	40
G3	196	80
A3	220	40
B3	246.94	40
C4	261.625	132
D4	293.66	56
E4	329.63	72
F4	349.23	36
G4	392	72
A4	440	36
B4	493.88	36
C5	523.25	54
D5	587.33	18

The reason for categorising the files according to gender is, as reported by other studies such as (Drugman and Alwan, 2011; Gonzalez and Brookes, 2014; Jouvet and Laprie, 2017; de Obaldía and Zölzer, 2019), the performances of pitch detector algorithms are different in male and female voices. Thus, there is a similar approach by the researchers that categorized their datasets to males' and females' voices to show the performances of the algorithms in each gender. The reason for this difference is not reported, but it is not simply changing the window size for calculating low- and high-pitch frequencies. One possible reason can be that the F0s' harmonies generated by men's and women's voices may differ. However, finding the reason for the difference in pitch estimators' performances in males' and females' voices is beyond this thesis's objective.

3.1.2 Tools

With the natural singing files, there is no accompanying file containing the exact musical pitches that are being sung and the times at which they are sung. To be able to assess the accuracy of the pitch detection algorithms, therefore, such an extension to the dataset must be made. It was considered that one possible way to achieve this is to use another signal decomposition tool that will facilitate the isolation of the fundamental component only from which an accurate pitch track can be obtained. This tool must be operated manually, and the fundamental frequencies must be identified visually. The *Spear* tool (Klingbeil, 2005) was discovered as being suitable for generating this ground truth. The Spear tool performs a frame-by-frame sinusoidal analysis (Serra and Smith, 1990), identifies all the important peaks in each frame, and connects together peaks that exhibit a trajectory. Additionally, the Spear tool provides both a visualization of all the important frequency components in an audio file, and a means by which they can be edited and removed. Thus, all unwanted components except the fundamental can be deleted, and a ground truth can be achieved. In Figure 3-1, two screenshots of the visualization of Spear are given. In the above part of this figure, the highlighted line, the red line, is the base pitch, and any other lines in black and grey are the harmonics. The strength of a component is indicated by the colour, varying from grey to black, illustrating weak to strong. By manually finding the fundamental frequencies, all other components can be deleted, and a frequency-varying sinusoid with respect to the fundamental frequency can be resynthesized. The isolated fundamental is shown in the lower panel in Figure 3-1. On all occasions, selecting the base pitch among the components was found to be straightforward.

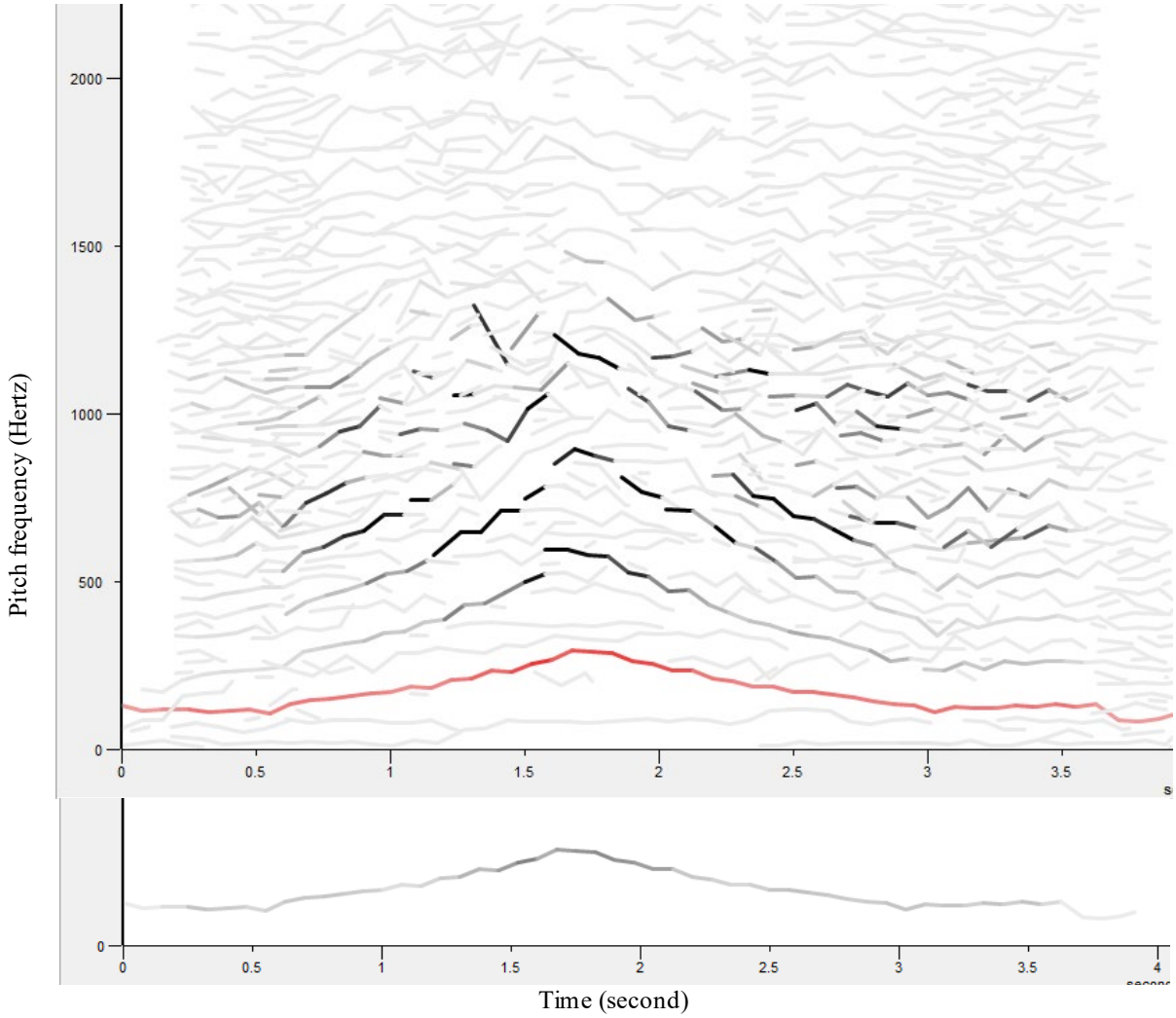


Figure 3-1 Example of the interface for Spear with the fundamental frequencies highlighted in red. In general, strong components are black, and weak ones are grey.

3.2 Pitch detection algorithms

Four well-known pitch detection algorithms (PYIN, Praat, PLL-Based, and Kalman filter) were selected for this study. Different tools were employed to implement these algorithms.

The *Tony* tool (Mauch *et al.*, 2015) was used to analyse as it contains an implementation of the PYIN algorithm (Mauch and Dixon, 2014). One of the valuable features of this tool is that once the pitches have been detected, all the estimated frequencies can be saved into a text file. The PYIN algorithm is based on the YIN algorithm, and its approach is to consider multiple candidates for the pitch based on a probabilistic interpretation of the earlier YIN (de Cheveigné and Kawahara, 2002) pitch detection algorithm. A HMM is used to produce the

final pitch track from the estimates (Mauch and Dixon, 2014). Regarding the Yin algorithm, upper and lower F0 search bounds are essential parameters for most methods. In contrast to other methods, YIN needs no upper limit (it tends, however, to fail for F0s beyond one-quarter of the sampling rate) (de Cheveigné and Kawahara, 2002). A wide range increases the likelihood of ''finding'' an incorrect estimate, so relatively low error rates despite a wide search range indicate robustness (de Cheveigné and Kawahara, 2002).

The *Praat tool* (Boersma and van Heuven, 2001) is employed to analyse the dataset based on the Boersma algorithm (Boersma, 1993). This uses an autocorrelation approach followed by dynamic programming to find the best path among a set of pitch candidates. The author of this paper (Boersma, 1993) was looking to tackle some issues regarding the sampling and windowing approaches to pitch detection. The main issue is accurately determining the position and height of the maximum peak in the autocorrelation function and, thus, the correct pitch (Boersma, 1993). In PRAAT, all the estimated frequencies can be saved into a text file.

Matlab is used to implement the Extended Complex Kalman Filter (ECKF) (Das, Smith and Chafe, 2017) and the PLL-Based pitch detection (Zolzer, Sankarababu and Moller, 2012) algorithms.

The ECKF algorithm operates on a sample-by-sample basis. This algorithm is based on the Kalman filter, a well-known approach to tracking parameters in noise. However, in the case of the ECKF, there is a nonlinear relationship between the changing state (pitch value) and the observation (the output waveform). Furthermore, the iterative nature of the algorithm means that at the beginning of a sound, it has difficulty estimating the pitch, but after a while, the parameter values adapt to give better estimates.

Similar to ECKF, the PLL-based pitch detection provides a sample-by-sample instantaneous pitch estimation. Its basic operation is to lock its internal oscillator to the input signal in such a way to minimize the error in phase between the two (Zolzer, Sankarababu and Moller, 2012). PLLs are a common tool in communication applications, and only recently have found applications for audio pitch detection.

One of the difficulties in implementing the PLL and Kalman algorithms is finding the best values for their parameters. In this study, in order to find the best value for their parameters, after selecting a wave file from the dataset, testing loops with small step sizes were used to create values for the parameters across their possible ranges. Many files were generated to assess the different values for parameters. Then, by plotting the data, the values for the parameters can work well with the selected sound file can be ascertained. Consequently, those parameters were then applied to the whole dataset, and the estimated pitch frequencies were saved.

The author of this dissertation created a Singing Data Analyser tool, written in C#, and was used to manage the testing of these pitch detection algorithms. The inputs of this software tool are the text files generated by the Tony, Praat, or Matlab software. After preparing the format of the input text files, this tool facilitated the plotting of the results.

3.3 Results

Praat and PYIN worked well with all dataset items, as highlighted in Table 3-2 and Table 3-3, without any incorrect pitch estimation. In these tables, *oct* means the octave-doubling problem, and *inc* means any other incorrect estimation other than octave-doubling. On the other hand, the performances of the PLL and Kalman were not promising. Because out of the 76 files, only the pitch contours from 16 and 48 files were found correctly by the PLL and Kalman algorithms, respectively. In addition, it can be observed that the Kalman algorithm worked better for female voices than male voices, with 70 correct estimations for females compared to 54 for male performances. In order to find the reason for this issue, one of the men's voices was selected to find the best algorithmic parameters for that file. After that, these new algorithmic parameters were applied to the files of men's voices only, but then this implementation resulted in more inaccurate pitch values in comparison to previously. Therefore, the sensitivity of the ECKF algorithm was clear.

Moreover, in both PLL and Kalman implementations, more problems were observed than just octave doubling: it was often trebling or more due to an octave overestimation or higher harmonic tracking. Examples of octave doubling are shown in Figure 3-2 (a) and other problems in Figure 3-2 (b). However, the detected contours for the PYIN and Praat algorithms

are overlaid with the ground truth contour and are thus hidden by it because they are the same except in a few instances that can be seen in Figure 3-2 (b) for the PYIN around time 3.75 sec and 5.5 sec that they are not fully aligned with the ground truth.

Table 3-2 the number of incorrect instances of f0 determination in the pitch detection algorithms for the fast-forte data

	Female				**Male**			
	Scale		Arpeggios		Scale		Arpeggios	
	oct[a]	inc[b]	oct	inc	oct	inc	oct	inc
PYIN	0	0	0	0	0	0	0	0
PLL	0	4	0	5	4	6	2	8
Kalman	2	0	2	1	4	3	5	2
Praat	0	0	0	0	0	0	0	0

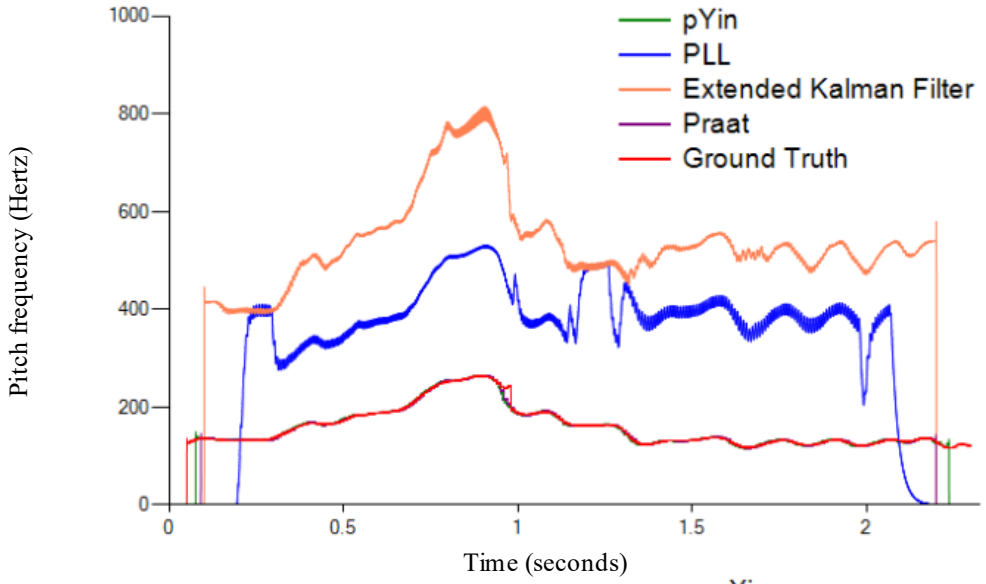
[a] oct = Octave-doubling
[b] inc = incorrect

Table 3-3 The number of incorrect instances of f0 determination in the pitch detection algorithms for the slow-forte data

	Female				**Male**			
	Scale		Arpeggios		Scale		Arpeggios	
	oct[a]	inc[b]	oct	inc	oct	inc	oct	inc
PYIN	0	0	0	0	0	0	0	0
PLL	3	3	2	4	3	6	4	6
Kalman	0	0	0	1	2	1	2	3
Praat	0	0	0	0	0	0	0	0

[a] oct = Octave-doubling
[b] inc = incorrect

(a)



(b)

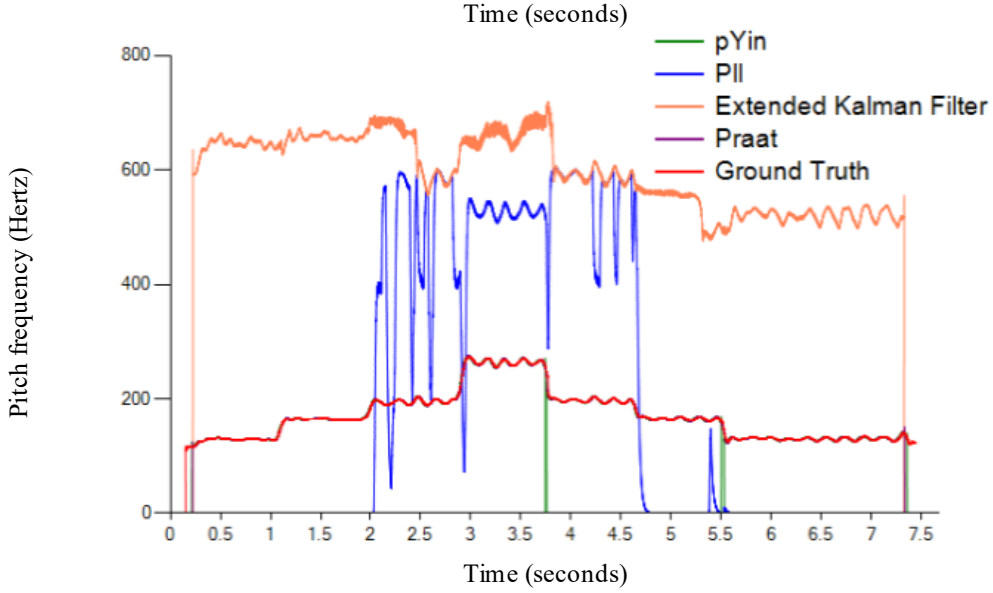


Figure 3-2 Example outputs of the various pitch detection algorithms

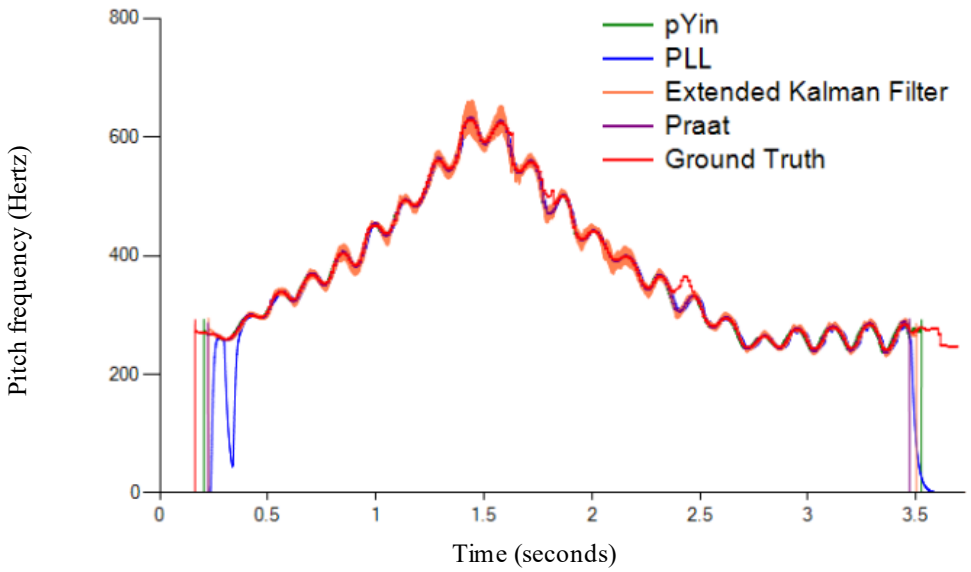


Figure 3-3 This picture depicts the PLL algorithm problem at the beginning of sounds

Another observation was that since the Kalman filter and the PLL are real-time algorithms that detect the pitch on a sample-by-sample basis, they will settle before they start to track correctly, as shown in Figure 3-3 that at the beginning of estimation, they could not calculate the F0s correctly. However, the PYIN and Praat algorithms have some pre-processing to get everything into line, forcing errant values to confirm their detected contour.

Another comparison was conducted to ascertain the standard deviation of the differences between each algorithm and the ground truth. It was conducted by subtracting on a sample-by-sample basis the estimated pitch of each algorithm from its corresponding pitch in the ground truth for all the 76 recorded files. Finally, the average of each algorithm's standard deviations was calculated and presented in Table 3-4. This measurement can provide an overview of the frequencies detected compared to ground truth. If the value is zero, the algorithm estimated the pitches without any variance, and if the result is closer to 0, it is better than when it is far from zero. This table shows that the best performance was recorded for PYIN, followed by Praat and Kalman, and the worst was for the PLL algorithm.

Table 3-4 Diff between the standard deviation of each algorithm with ground truth

	PYIN	Praat	Kalman	PLL
Diff Std with ground truth's Std (in Hertz)	54.13	64.15	83.99	99.89

3.4 Conclusion

This chapter has analysed the performance of four pitch detection algorithms. The PLL and ECKF operate on a sample-by-sample basis, so they seem suitable for real-time implementations. From the experimental results, it was observed that PYIN had the best performance, followed by Praat. However, these two algorithms are offline; thus, they are unsuitable for the real-time singing analysing objectives of this thesis, albeit they will be used to generate ground truth data in Chapters 5, 6 and 7. On the other hand, the performance of the ECKF and PLL algorithms was not good and, in many cases, exhibited inaccuracies that were manifested as the pitch appearing in the incorrect octave and displaying spurious large deviations. One problem recognized with the ECKF and PLL algorithms is that they are very sensitive to the set of algorithm parameters and highly dependent on the input data. This means that for different input data, it is necessary to find the most appropriate values before

applying the algorithm. This is a disincentive to applying these algorithms, which means a significant pre-processing step is required.

The results from a previous study by (Gupta, Li and Wang, 2017) support our result that the PRAAT (Boersma and van Heuven, 2001) pitch estimator gives the best voicing boundaries even without post-processing, while the source-filter model-based STRAIGHT (Kawahara, Estill and Fujimura, 2001) pitch estimator is the most robust algorithm in noisy conditions. Finally, the modified autocorrelation-based estimator YIN (de Cheveigné and Kawahara, 2002) achieves the best pitch detection accuracy, but it requires many post-processing steps depending on the properties of the music type being analysed, as described in (Dixon and Effects, 2005).

It is found that the PLL and ECKF algorithms should be modified to find the best implementation arrangement to work well with human voices. In addition, an effective method should be found that could adjust the algorithm parameters dynamically in scenarios where input signals have properties that vary rapidly.

The generated ground truth dataset and the found reliable offline pitch detection algorithms, PYIN and Praat, can be used as ground truth to compare the real-time algorithms' accuracy. Therefore, the next chapter will evaluate several real-time pitch detection algorithms.

Chapter 4

An investigation into several real-time pitch detection algorithms in singing signals

The two main objectives of this chapter are: firstly, comparing several real-time pitch detection algorithms with different parameters to find the best ones for detecting the pitch from live monophonic singing. Secondly, finding a method to distinguish the set of correctly detected pitches. The contents of this chapter are already published in the following preprint paper.

- *Faghih, Behnam & Timoney, Joseph, "Real-Time monophonic singing pitch detection".* http://dx.doi.org/10.13140/RG.2.2.22054.19526

This chapter evaluates the efficiency and correctness of several real-time pitch detection algorithms applied to the singing voice based on different. Moreover, several approaches are investigated to assess the estimated pitches' accuracy. Therefore, after explaining the research methodology in section 4.1, the results and discussion are presented in section 4.2. A section with a conclusion will follow this.

4.1 Materials and Methods

4.1.1 Pitch detection algorithms

As it is mentioned in Chapter 3, the newer real-time pitch detection algorithms could not find the F0s from signing signals appropriately. Thus, this chapter selected some well-known/ common in used real-time algorithms, but older algorithms, to evaluate their performance in signing signals.

Therefore, the seven real-time pitch detection algorithms investigated are Yin (de Cheveigné and Kawahara, 2002), spectral YIN or YIN Fast Fourier transform (YinFFT), YinFast, Fast comb spectral model (FComb), Multi comb spectral filtering (MComb), Schmitt trigger, and spectral auto-correlation function (Specacf). The Aubio[1] library (Aubio, no date), one of the well-known libraries in music information retrieval in Python, was used for implementing these algorithms. A short description of each algorithm is presented in the following, and the full details of their workings are given by Brossier (Brossier, 2006).

Yin: as mentioned in Chapter 3, Yin is a time-domain function called the cumulative mean normalised difference function (CMDF). It is related to an earlier pitch detection technique called the Average Magnitude Difference Function (AMDF), proposed as a computationally efficient alternative to the autocorrelation function. The estimated pitch comes from finding the peak in this CMDF. It normally employs some form of interpolation, parabolic interpolation, for example, to improve the accuracy of the estimated pitch (Faghih and Timoney, 2019a).

YinFFT: it is a spectral domain implementation of Yin by employing the Fast Fourier Transform (FFT). In this implementation of the Yin algorithm, the dependency on the threshold

[1] https://aubio.org/manual/latest/cli.html#aubiopitch

parameter is removed by selecting the best period candidate by finding the minimum in the difference function (Brossier, 2006).

YinFast: it is an optimised implementation of the YIN algorithm to improve its time complexity from $O(n^2)$ to $O(n\log(n))$. It uses two Fourier transforms in the different functions of the Yin algorithm to reduce the cost of calculating the whole spectrum (Brossier, 2006).

FComb: it is a spectral pitch detection approach that finds the N peaks with the most significant energy within each spectral frame and stores them in an array. Then, it will compare the peaks to each other by starting with the most energetic one. If the other F0s is a subharmonic of the highest selected peak, and its energy is higher than half of the selected peak, it is designated as the F0 (Lang, 2003; Brossier, 2006).

MComb: it is a spectral frame approach that includes some pre-processing on the input signal to enhance mid-range frequencies and reduce the high and low parts of the spectrum. Then, after detecting the spectral peaks, they are passed to a harmonic comb. The assumption for monophonic signals is that one of these most substantial peaks corresponds to the pitch of the present note (Lepain, 1999; Bello Correa, 2003; Brossier, 2006).

Schmitt trigger: it is one of the time-domain approaches based on using thresholds applied to the waveform of the selected frame to find its periodicity and then estimate the pitch (Simpson, 1987; Lang, 2003; Brossier, 2006).

Specacf: the autocorrelation function compares the time-domain data of the frame with delayed versions of itself on a sample-by-sample basis. The autocorrelation function will show peaks in proportion to the periodicity of the frame. Detecting these peaks and then weighting the spectral components based on their spectral location leads to estimates for F0 (Klapuri, 2000; Brossier, 2006).

4.1.2 Generating a Dataset

The database of this study is the same as the dataset of the previous chapter, discussed in section 3.1.1. Therefore, the ground truth generated in the previous chapter was used.

In order to compare the efficiency of the pitch detection algorithms, several parameters and processes were employed on the selected files from VocalSet, and thus, an extended

dataset was generated. To achieve this goal, the selected files were played through an ordinary speaker and recorded by an ordinary microphone. Based on the experience of Jouvet and Laprie (Jouvet and Laprie, 2017) that a close-talk microphone provides clean data, the microphone was located very close to the speakers, at a distance of less than 10cm.

Moreover, 44100 samples per second were considered for the recording files' sampling rate. Then, each algorithm was applied twice on each file with different window sizes, 1024 and 2048 samples, and the hop size was half the window size. The reason for selecting these window sizes is to have enough samples for the range of the human singing pitch frequencies used in this study. As reported by Paliwal and Wojcicki (Paliwal and Wojcicki, 2008), windows with a duration of around 15-35 ms are a good choice for pitch detection in human voices. Therefore, the window sizes 1024 and 2048 were selected to have the duration of around 12 ms and 23 ms, respectively. Similarly, Gawlik & Wszołek (Gawlik and Wszołek, 2018) used a window with a duration of 22 ms for singing pitch detection.

Therefore, each file was played 14 times, and the seven different algorithms (as explained in Section 4.1.1) were applied twice, but with different window sizes. The detected pitches were saved in different text files. Each algorithm incorrectly identified some pitch values. This occurred when the incorrect spectral peak was identified as the F0 and produced a result that gave an obvious disturbance to the smoothness of the pitch contour. To fix these incorrect pitches, a Smart-Median post-processing algorithm (Faghih and Timoney, 2022b), which will be shortly explained in section 4.1.3.1 and fully in Chapter 5, was applied.

As a result, a dataset containing 2128 files was created. These files were categorised by the algorithm (seven algorithms), Gender (male and female), interval (scale and arpeggios), with and without post-processing, window sizes (1024 and 2048), and speed of performance (slow and fast). These categories and the total number of files in each category are shown in Table 4-1. Thus, there are 304 files in the generated dataset for each algorithm. This dataset is available online in a GitHub repository[1].

[1] https://github.com/BehnamFaghihMusicTech/Singing-Pitch-Detection, accessed on 11 July 2022

Table 4-1. Categories and number of files in the generated dataset for each pitch detection algorithm. The columns titled "with" mean post-processing, and the columns titled "without" mean without post-processing.

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Slow	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10
Fast	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10

4.1.3 Post-processing

After recording the estimated pitches by each algorithm, two post-processes, Smart-Median (Faghih and Timoney, 2022b) and shifting, were applied to the recorded files. The steps of these post-processes are depicted in Figure 4-1.

4.1.3.1 Smart-Median

Smart-Median is a novel pitch contour smoother algorithm defined especially for pitch contours created by real-time pitch detectors on singing signals. This algorithm adjusts the standard median and is tuned to the features of singing pitch contours. The Smart-Median will be discussed in detail in Chapter 5.

Section 4.2.3 explains how the Smart-Median algorithm improves the accuracy of the pitch contours by a factor of 5.

4.1.3.2 Shifting

The detected pitches should be compared with the ground truth to evaluate each algorithm's performance. However, before that, finding the best alignment between the detected pitches and the ground truth is necessary. This has been determined by shifting the recorded file backwards or forwards to ascertain the minimum distance between it and its corresponding ground truth (Jouvet and Laprie, 2017). Figure 4-1(d1) and Figure 4-1(d2), as compared to Figure 4-1(c1) and Figure 4-1(c2), illustrate how shifting makes the best alignment between the estimated and ground-truth pitch contours.

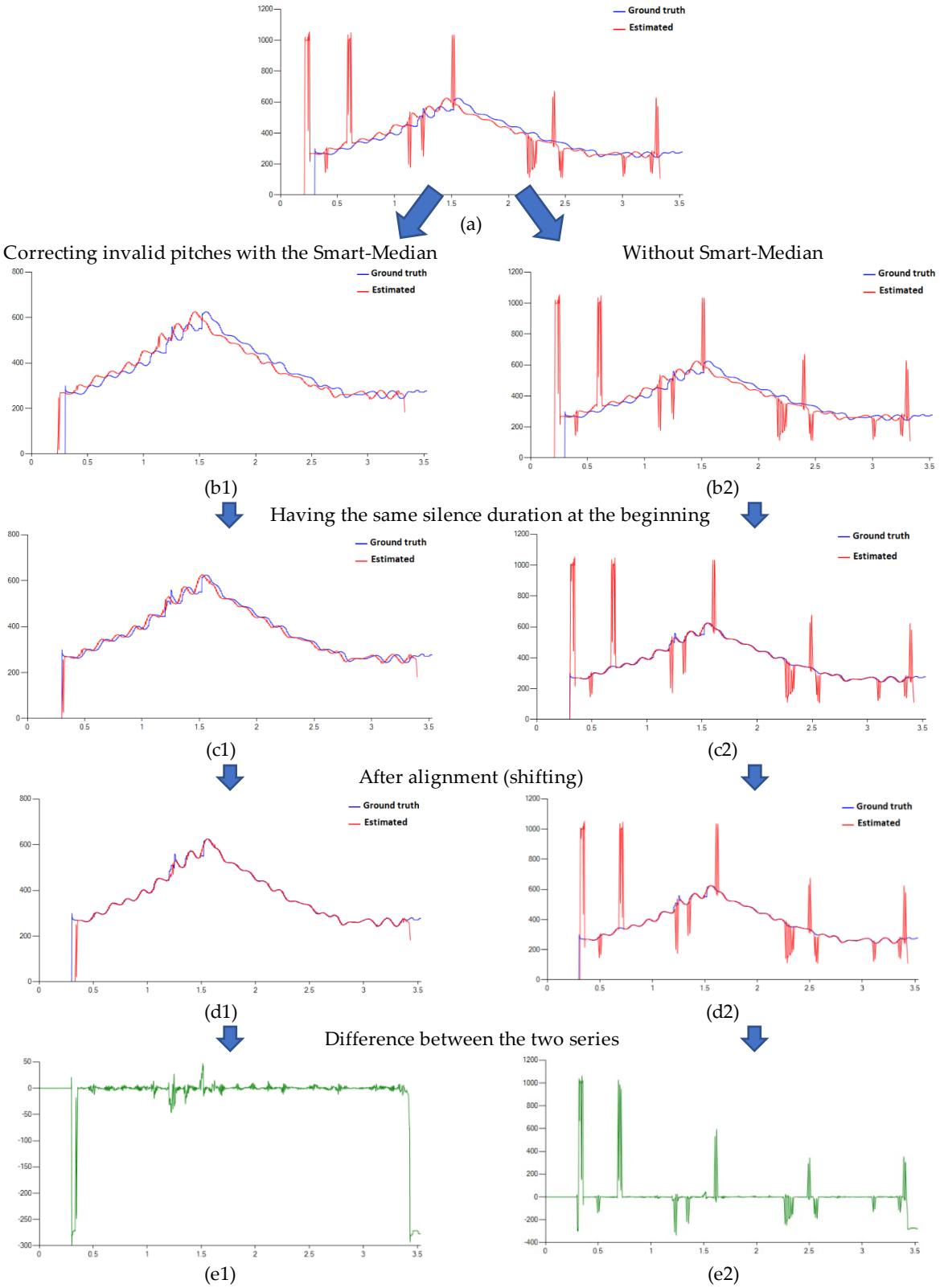


Figure 4-1. Illustrating post processes on detected pitches. (a) is the detected frequencies, (b1) is the detected pitches after replacing invalid frequencies with Smart-Median, (b2) is exactly frequencies in (a) without correcting the invalid pitches. (c1) and (c2) are setting the silence duration at the beginning of the estimated frequencies as long as the ground truth. Frequencies are shifted in (d1) and (d2) to find the best alignment with the ground truth. (e1) and (e2) show the difference between ground truth and estimated frequencies.

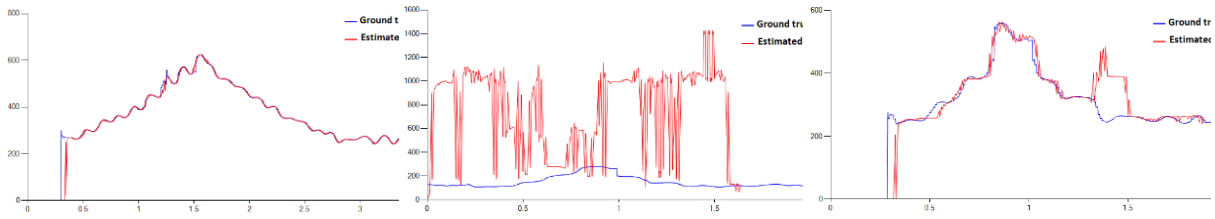
It should be mentioned that having different silence durations at the beginning of each recorded file was unavoidable due to the time of pressing the start recording button. Thus, before shifting, it is set to be precisely the exact duration of silence by adding/removing some zero values at the beginning of each recorded file. Consequently, there is a match at the beginning of the recorded files and corresponding ground truth files. Figure 4-1(c1) and Figure 4-1(c2) illustrate that when the estimated pitches in Figure 4-1(b1) and Figure 4-1(b2) have the same silence at the beginning, we have a better alignment with the ground truth. In addition, it is possible to figure out the delay of each algorithm to estimate the pitches correctly by finding the duration took that the estimated pitch contour aligned well with the ground truth.

4.1.4 The difference between estimated pitch contour and ground truth

To better understand the performance of each pitch detection algorithm, the difference between the estimated pitches and the ground truth is prepared. This is shown in Figure 4-1(e1) and Figure 4-1(e2). Therefore, after shifting the estimated pitches to find the best alignment, the differences between all the values were calculated by subtracting each estimated F0 value from its corresponding ground truth's F0 value, and the result was saved in a text file. Finally, these files are combined with the dataset.

4.1.5 Labelling estimated pitch contours

After generating the dataset as explained above, all the files were plotted (similar to Figure 4-1(d1)) to label whether the estimated pitch contours were correct. All the files were categorised into three groups: **1**- correct (when the plotted contours of the estimated pitches are perfectly aligned on the corresponding ground truth contours, Figure 4-2(a)), **2**- incorrect (such that the estimated pitch contour is not correct and that no matter what extra post-processing is applied they will not become correct, as illustrated in Figure 4-2 (b)), **3**- almost correct means that this needs more post-processing to improve the result (i.e. on comparing the pitch contour of the estimated pitches with the pitch contour of the ground truth, it is recognised that a small number of pitches are misestimated, and it is expected that with some judicious post-processing, they could be corrected, as exemplified in Figure 4-2(c)).



(a) correct (b) incorrect (c) needs more process

Figure 4-2. The three categories for estimated pitch contours based on their correctness. (a) almost all the pitches are correct. (b) most of the pitches are incorrect, and (c) a few of the estimated pitches are incorrect, but it is expected that the incorrect pitches can be fixed with more post-processing.

As illustrated in Table 4-2, none of the algorithms can estimate the pitch contour properly without the Smart-Median post-processing. The best performance without post-processing was recorded for MComb, followed by YinFast, YinFFT, and Yin. While Schmitt, FComb, and Specacf could not correctly determine pitch contour without post-processing. A possible reason that Schmitt and Specacf algorithms could not estimate F0s correctly is that they are working on the time domain and sample-by-sample approach, which is challenging according to singing signals that are very fluctuation compared to signals from most instruments.

Table 4-2. Number of correct and incorrect estimated pitch contours for each algorithm without post-processing

	Correct	Need more process	Incorrect
Yin	4	133	15
YinFast	19	124	9
YinFFT	19	100	33
FComb	0	3	149
MComb	32	42	78
Schmitt	0	0	152
Specacf	0	0	152

Nevertheless, according to Table 4-3, after the post-processing, Yin, YinFFT, and YinFast could correctly produce more than 58% of pitch contours. Although YinFFT recorded the highest number of correct pitch contours, it seems possible to improve Yin and YinFast performances by designing some new post-processing algorithms.

As can be observed from Table 4-2 and Table 4-3, the Smart-Median post-processing improved the performance of the algorithms by more than five times overall; that is, there were 74 correct pitch contours without the post-processing as compared to 382 correct pitch contours after the post-processing.

Table 4-3. Number of correct and incorrect estimated pitch contours for each algorithm after post-processing

	Correct	Need more process	Incorrect
Yin	89	60	3
YinFast	101	50	1
YinFFT	107	26	19
FComb	20	44	88
MComb	65	21	66
Schmitt	0	0	152
Specacf	0	1	151

4.2 Results and Discussions

There are some common evaluation strategies used when comparing pitch detection algorithms. Researchers such as (Cheng *et al.*, 1976; Rabiner *et al.*, 1976; Drugman and Alwan, 2011; Jouvet and Laprie, 2017; Drugman *et al.*, 2018; de Obaldía and Zölzer, 2019) usually compare pitch detection algorithms based on four methods. First, the Voicing Decision Error (VDE) (Nakatani *et al.*, 2008) is the proportion of frames for which an error in the voicing decision is made. Second, the Gross Pitch Error (GPE) (Nakatani *et al.*, 2008) is the proportion of frames that are considered to be voiced by both the ground truth and the pitch estimator where the relative error of F0 is higher than a certain threshold. Third, Fine Pitch Error (FPE) (Wei Chu and Alwan, 2009) is defined as the standard deviation of the distribution of the relative error of F0 for which this error is below a threshold (usually 20%). Fourth, the F0 Frame Error (FFE) is the proportion of frames where an error is made. FFE alone can provide an overall performance of a pitch detection algorithm (Wei Chu and Alwan, 2009; Drugman and Alwan, 2011).

In this study, besides FFE, several other comparisons have been made on the data obtained from the pitch contours estimated by various algorithms to contrast the algorithms' performance, thus ascertaining their efficiencies in different situations. The bases for the comparisons for this study were as follows:

- *Correctness*: how many pitch contours estimated by each algorithm are entirely correct?

- *Delay*: what is the time delay from the commencement of the analysis to when the estimate of F0 is accurate?
- *Post-processing effect*: what is the percentage of the number of correctly detected pitches by each algorithm before and after post-processing?
- *Overall*: how accurate is the estimated F0?

4.2.1 Correctness

In this section, the performances of each algorithm based on their correctness are evaluated. These evaluations are categorised into three groups of data: 1- both fast and slow, 2- fast, and 3- slow performances.

4.2.1.1 The correctness of both fast and slow performance

Table 4-3 shows, without considering any particular category, the order of performance of all algorithms from the best to the worst after post-processing. Out of 152 pitch contours in total for each algorithm, the percentage of contours for each that were identified as correct were, in order of best first, YinFFT = 70.39%, YinFast = 66.45%, Yin = 58.55%, MComb = 42.76%, FComb = 11.84%, Schmitt = 0%, and Specacf = 0%.

From Brossier (Brossier, 2006), their experience of analysing recorded Opera performances of two men and two women, the ordering of the algorithms in terms of their accuracy was YinFFT 56%, MComb 52%, FComb 49%, Yin 37% and Schmitt 23%, which supports our result. The main difference between the Brossier result and ours is the position of the Yin algorithm in the ranking. It should be mentioned that before post-processing, we had the same experience that the performance of the MComb was better than Yin, but after the post-processing, the order switched.

In addition to this, in our experience, the accuracy of the algorithms is different for males and females. This is illustrated in Table 4-4. For females, the best algorithm to the worst one after the post-processing are YinFFT = 100%, YinFast = 93.06%, Yin = 72.22%, MComb = 65.28%, FComb = 25%, Schmitt = 0%, and Specacf = 0%. On the other hand, the order for males after post-processing is Yin = 46.25%, YinFFT = 43.75%, YinFast = 42.5%, MComb = 22.5%, FComb = 2.5%, Schmitt = 0%, and Specacf = 0%. Therefore, the best approach for females is YinFFT, but for males, it is Yin. Generally, it is observed that pitch detector algorithms could

work better on females' voices than males' voices. The possible reasons for this different performance of the algorithms can come from the difference between men's voices and women's voices spectrogram. That is, the harmonies layers and their amplitude in spectrograms are different (Jitendra and Radhika, 2021).

Table 4-4. Total correctness of each algorithm categorised by gender, music type, post-processing, and window size in both fast and slow performance

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin	1	0	17	10	1	1	16	9	0	0	12	9	1	0	9	7
YinFast	7	1	16	18	5	2	16	17	2	0	11	8	2	0	11	4
YinFFT	8	4	18	18	4	3	18	18	0	0	4	13	0	0	4	14
schmitt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FComb	0	0	2	6	0	0	3	7	0	0	0	1	0	0	0	1
MComb	2	9	10	10	2	9	14	13	0	5	3	5	1	4	3	7
Specacf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It can be concluded from these results that the algorithms perform better on Female voices than Male ones, with 25.4% as opposed to 11.25% correct contours, respectively. Although we used two different window sizes, considering our results alongside those of the de Obaldía and Zölzer study (de Obaldía and Zölzer, 2019), it seems that for detecting the low pitch frequencies exhibited by men's voices, a window size of 4096 samples in 44100Hz sample rate should be a better choice. However, although de Obaldía and Zölzer (de Obaldía and Zölzer, 2019) used window sizes of 2048 and 4096, they also experienced better results on female voices over male voices. However, a long window size may not be suitable for real-time applications since it reduces the resolution of the estimated pitches. For example, the window size of 4096 in 44100Hz sampling rate, making an extra delay of 46 ms compared to a window size of 2048. In addition, it can cause more inaccuracy on detecting onset and offset of notes, which is discussed in Chapter 6, because onset of a note will be considered one of the estimated pitches.

Regarding window size, a significant performance difference was not found between the two window sizes for our analysis in that there are 35.15% and 36.65% correct pitch contours

for 1024 and 2048 window sizes, respectively, after post-processing. Similarly, there was no performance difference between the Scale and Arpeggios test signals because 35.9% correct pitch contours were observed for both.

To recap, a significant different in the window sizes was not found, but the algorithm with the maximum number of correct pitch contours for females was YinFFT, while for males, it was YinFast. Furthermore, for both genders, post-processing is necessary. Finally, post-processing has improved the accuracy of algorithms by more than a factor of 5.

4.2.1.2 The correctness of fast performance

For the fast performance, as can be observed from Table 4-5, the best performance after post-processing was recorded by YinFast with 76.32% accurate pitch contour estimations, followed by YinFFT = 68.42%, Yin = 63.16%, and MComb = 61.84%. On the other hand, the worst algorithms were FComb, Schmitt, and Specacf, with 11.84%, 0%, and 0%, respectively.

Table 4-5. Total correctness of each algorithm categorised by gender, music type, post-processing, and window size in only fast performance

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin	0	0	9	3	1	1	8	1	0	0	6	9	0	0	5	7
YinFast	7	0	9	9	5	1	8	9	2	0	9	4	2	0	8	2
YinFFT	6	4	9	9	2	1	9	9	0	0	2	6	0	0	2	6
schmitt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FComb	0	0	0	4	0	0	0	3	0	0	0	1	0	0	0	1
MComb	2	9	9	9	2	9	9	8	0	5	2	4	0	4	2	4
Specacf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In addition, the algorithms do not have the same accuracy for both male and female voices. The order from the best to the worst algorithm for females' voices is YinFFT = 50%, YinFast = 48.61%, MComb = 48.61%, Yin = 29.17%, FComb = 9.72%, Schmitt = 0%, and Specacf = 0%. On the other hand, the order for males is Yin = 33.75%, YinFast = 28.75%, YinFFT = 20%, MComb = 15%, FComb = 2.5%, Schmitt = 0%, and Specacf = 0%. Thus, the best ones after post-processing for female voices were YinFFT, with YinFast, and MComb coming very close, but for male voices, it was Yin.

Post-processing generally improved algorithms' accuracy by a factor of more than 3. Again, the algorithms work much better on female than male voices, i.e., 53.17% correct compared to 28.57%, respectively. A significant difference between the two window sizes of 1024 and 2048 is not observed. Similarly, a significant difference is not observed between the results for Scale and Arpeggios test data.

To recap, the best accuracy for female voices was obtained by the YinFFT algorithm with a window size of 1024, but for male voices, it was simply Yin with a window size of 2048. Moreover, for both genders, post-processing was necessary.

4.2.1.3 The correctness of slow performance

Regarding the slower performances, the best accuracy after post-processing was shown by YinFFT with 72.37% accurate pitch contours, followed by YinFast = 56.58% and Yin = 53.95%. On the other hand, the worst algorithms are MComb, FComb, Schmitt, and Specacf, with the number of correct pitch contours being 23.68%, 15.79%, 0%, and 0%, as shown in Table 4-6.

Table 4-6. Total correctness of each algorithm categorised by gender, music type, post-processing, and window size in only slow performance

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin	1	0	8	7	0	0	8	8	0	0	6	0	1	0	4	0
YinFast	0	0	7	9	0	1	8	8	0	0	2	4	0	0	3	2
YinFFT	2	0	9	9	2	2	9	9	0	0	2	7	0	0	2	8
schmitt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FComb	0	0	2	2	0	0	3	4	0	0	0	0	0	0	1	0
MComb	0	0	1	1	0	0	5	5	0	0	1	1	1	0	1	3
Specacf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The accuracy of the algorithms is the same for both male and female voices. For both genders, the best one is YinFFT, and the best algorithm to the worst one is YinFFT, YinFast, Yin, MComb, FComb, Schmitt, and Specacf. Post-processing improved the accuracy of the algorithms by approximately 17 times.

As previously observed, the algorithms work better on female than male voices, with 48.41% correct pitch contours compared to 16.79% correct ones after post-processing, respectively. A significant difference between the two window sizes, 1024 and 2048, is not observed, albeit the bigger window size worked a bit better on male's voices. Similarly, a significant difference is not observed between the performances of Scales and Arpeggios.

In summary, for both genders, the best accuracy is obtained with the YinFFT, but for female voices, a window size of 1024 worked better, while for male voices, it was a window size of 2048. In addition to this, post-processing is necessary to improve the performance of both genders' voices.

4.2.2 Delay

It will take a short time for most algorithms to start estimating the pitches correctly. Since the algorithms use different methods for estimating F0, this time is quite variable. This subsection evaluates the time each algorithm took to estimate the pitch contour correctly. We describe this as a delay.

In order to find the delay, only pitch contours deemed to be correct after the post-processing were selected, and the exact amount of silence was set at the beginning of each file to have the exact duration of the silence at the beginning of the corresponding ground truth file as can be observed looking from Figure 4-1(b1, b2) to Figure 4-1(c1, c2). Three categories were considered for evaluating the delay: 1- both slow and fast performances, 2- fast performances, and 3- slow performances.

4.2.2.1 Delay in both fast and slow performances

Table 4-7 shows the average duration of the delays recorded by each algorithm. A shorter delay is believed to be better than a longer one since it can estimate accurate pitches more instantaneously. It can be observed from Table 4-7 that the best performance recorded by MComb is with a 25 ms delay, followed by FComb with a delay of 34ms, YinFFT exhibits a delay of 45ms, Yin has one of 58ms, and YinFast was the slowest with a delay of 60ms.

Alongside this, the average duration of the delays for the algorithms differs for the male and female voices. The best performance was given by FComb for female voices, but for Male voices, it was by MComb.

Post-processing, on average, increases the offset due to a delay from 94 ms to 245 ms. The reason is that the post-processes are ameliorating the incorrectly detected pitches at the beginning of the sounds.

Overall, the algorithms worked better with female voices than male voices, leading to a 96 ms delay compared to 243 ms, respectively.

Table 4-7. The average delay in each algorithm categorised by gender, music type, post-processing, and window size in both fast and slow performance (in milliseconds)

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin	65	4	38	21	27	0	35	37			174	80	95		107	78
YinFast	22		39	19	34	8	48	20	89		147	93	126		101	38
YinFFT	32	15	24	14	43	14	40	16		16	129	120		34	61	67
Schmitt																
FComb			13	14				0				73			43	59
MComb	17	7	25	11	30	4	25	9		32	40	40		42	38	26
Specacf																

According to Table 4-7, the delay recorded for the 2048 window size is almost half of that for the 1024 window size, on average; i.e., it was given by the values of 126 ms and 213 ms, respectively.

However, a significant difference is not observed between the Scale and Arpeggios test signals' delay values, 177 ms and 163 ms on average, respectively.

To recap, for both male and female voices, the best accuracy concerning delay was obtained for MComb with a window size of 2048. Moreover, as can be seen in Table 4-8 and Table 4-9, the same conclusion can be reached regarding the fast and slow performances. Thus, the speed of performance does not significantly influence the delay.

Table 4-8. The average delay in each algorithm categorised by gender, music type, post-processing, and window size in only fast performance (in milliseconds)

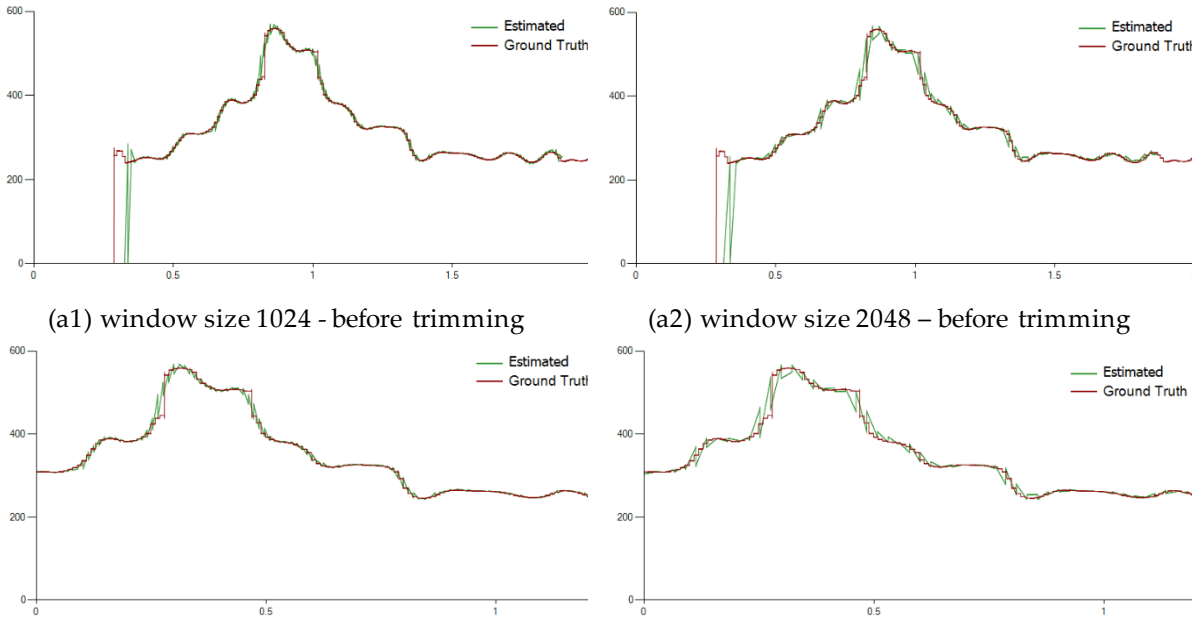
	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin			52	16	27	0	33	23			145	104			114	90
YinFast	22	6	32	6	34	2	27	-4	89		116	52	143		100	17
YinFFT	28	5	31	13	33	1	32	12			81	83			133	68
Schmitt																
FComb				14				3				73				59
MComb	17	7	25	10	30	4	22	6		32	61	41		42	41	40
Specacf																

Table 4-9. The average delay in each algorithm categorised by gender, music type, post-processing, and window size in only slow performance (in milliseconds)

	Female								**Male**							
	Scale				**Arpeggios**				**Scale**				**Arpeggios**			
	Without		*With*		*Without*		*With*		*Without*		*With*		*Without*		*With*	
Window size	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**	**1024**	**2048**
Yin	65		40	27			36	61			192	61	95		102	
YinFast			45	22		14	66	23			178	114	91		109	51
YinFFT	34	15	17	15	44	10	48	21		16	130	144		34	62	58
Schmitt																
FComb			13					1							43	
MComb				15			34	12			52	39			35	6
Specacf																

4.2.3 Evaluating the accuracy of the estimated F0

Another analysis conducted was to create a reliable technique that determines whether or not an estimated F0 is correct. As can be observed from Figure 4-1(d1, e1), pitch detector algorithms do not estimate pitches precisely with the same value but within an acceptable range. For example, the estimated pitch contour is not perfectly aligned with the ground truth in Figure 4-3.



(a1) window size 1024 - before trimming

(a2) window size 2048 – before trimming

(b1) window size 1024 – after trimming

(b2) window size 2048 – after trimming

Figure 4-3. Trimming estimated pitch contour. (a1) and (a2) are post-processed estimated pitch contours with window sizes 1024 and 2048, respectively. (b1) and (b2) represent (a1) and (a2) by removing 15% of duration from the beginning and also 15% from the end of the pitch contours.

Only the pitch contours estimated correctly, with the FFE less than 1%, were considered to find the acceptable range. However, it should be mentioned that almost all correctly estimated contours can still have some problems at the beginning and end of the sounds. Therefore, the sound waveforms were truncated by 15% at either end to remove the incorrectly estimated pitches, as shown in the lower panels of Figure 4-3.

The range of frequencies after this trimming was approximately between 100 and 600 Hz. This range covers approximately from G2 to D5, from a baritone to a mezzo-soprano, which is almost the most common range of human pitch frequency. For example, children aged between 1 and 15 years old have a frequency range between the notes G3, 196 Hz, and E5, 659.25 Hz (Welch, 1979), and a normative adult male voice profile is approximately between 85 Hz and 580 Hz, and for a female voice is from 150 Hz to 750 Hz (Heylen *et al.*, 2002).

As one of the observations of this study, after finding the best alignment, none of the pitch detection algorithms used in this study estimated any pitches with the exact frequency as the ground truth. In other words, there is not any point in Figure 4-3 where the estimated pitch and the ground truth have precisely the same value. Therefore, an acceptable range of

frequency values should be determined to assess the correctness of an estimated pitch. From other studies (Drugman and Alwan, 2011; Jouvet and Laprie, 2017; de Obaldía and Zölzer, 2019), three different methods can be employed to discover an acceptable range of frequencies based on ground truth: 1- fixed distance around ground truth F0, 2- the standard deviation of differences, and 3- percentage.

4.2.3.1 Fixed distance around ground truth F0

One of the methods used for finding an acceptable range of frequencies is to consider an upper and lower bound that is effectively both greater and less than the corresponding F0 in the ground truth, as shown in (4-1).

$$S_i = \begin{cases} true, & GTF0_i - interval \leq EF0_i \leq GTF0_i + interval \\ false, & otherwise \end{cases} \quad (4\text{-}1)$$

where $interval$ denotes a range around the Ground Truth F0 (*GTF0*), and $EF0_i$ is the estimated F0 in the index i. The distance metric that satisfies 95% of the estimated pitches is considered to be an acceptable distance. That is, the requirement is that 95% of the estimated pitches should fall within a certain interval distance from the ground truth pitch frequency. Table 4-10 shows the acceptable distance for each algorithm. It is observed that YinFFT and YinFast have the minimum distances, FComb has the maximum, and the other algorithms are close to each other.

Table 4-10. The acceptable fixed distance from F0 in the ground truth

Algorithm	Distance (Hertz)
Yin	35
YinFast	28
YinFFT	21
Schmitt	NA
FComb	53
MComb	36
Specacf	NA

However, Table 4-10 only shows the distance where 95% of the estimated pitches are in that range, but it does not present the distribution of data, and because of this, the second method, in the following section, is included.

4.2.3.2 The standard deviation of differences

This method calculates the differences between points in the ground truth and the estimated pitches, then computes the standard deviation of the differences, as in (4-2) and (4-3).

$$difference_i = EF0_i - GTF0_i \tag{4-2}$$

$$STD = StandardDeviation\ (difference) \tag{4-3}$$

where $difference$ is a vector to record all the differences, $EF0$ is a vector that contains all the estimated pitches, $GTF0$ is a vector that includes all ground truth pitches, and i is the index. $StandardDeviation$ is a function to calculate the standard deviation based on its input.

After finding the standard deviation for each estimated pitch contour, the pitch contours labelled as correct, where the estimated pitches were in the vicinity of $GTF0$ and were within C times the standard deviation on either side of it, are determined using the expression given in (4-4).

$$S_i = \begin{cases} true, & GTF0_i - (C * STD) \leq EF0_i \leq GTF0_i + (C * STD) \\ false, & otherwise \end{cases} \tag{4-4}$$

where S is a vector that stores whether or not the estimated pitches are correct, and C is a multiplier coefficient.

Several coefficients, from 0.1 to 2.5 with the step of 0.1 (that is, 0.1, 0.2, 0.3, …, 2.4, 2.5), were employed to find the best one for each algorithm. The minimum coefficient value that satisfies 95 per cent of estimated pitches was considered the best one, see Table 4-11. In Table 4-11, the acceptable interval from the estimated F0 to the ground truth's F0 was calculated by multiplying the standard deviation by the coefficient value. Then, the estimated F0 was considered as correct if it was in the range of the corresponding pitch frequency of ground truth plus and minus the acceptable range.

Table 4-11. The average standard deviation of differences between estimated pitches and ground truth with the coefficient of the acceptable distance

Algorithm	Average of STD (Hertz)	Coefficient	Acceptable range
Yin	12.88	2.1	±27.05
YinFast	13.3	2.1	±27.93
YinFFT	10.27	2.1	±21.57
Schmitt	NA	NA	NA
FComb	21.17	2.5	±52.93
MComb	13.88	2.3	±31.92
Specacf	NA	NA	NA

It was observed that YinFFT had the narrowest acceptable distance, which means that it has less variance than other algorithms, as shown in Table 4-11. Since Schmitt or Specacf did not estimate any entirely correct pitch contours, the standard deviation for these algorithms could not be calculated.

4.2.3.3 Percentage

The other method applied to find an acceptable frequency range is by calculating the range based on a percentage of the pitch frequency of the ground truth, as in (4-5).

$$\begin{cases} true, & GTF0_i - (r * GTF0_i) \leq EF0_i \leq GTF0_i + (r * GTF0_i) \\ false, & otherwise \end{cases} \quad (4\text{-}5)$$

where r is a coefficient between 0 and 1 to calculate the acceptable percentage value.

In this method, we were looking for a percentage around the ground truth's pitches that satisfies at least 95% of the estimated pitches, Table 4-12.

Table 4-12. Per cent of ground truth F0 for finding the acceptable estimated pitch

Algorithm	Percentage
Yin	8%
YinFast	7%
YinFFT	6%
Schmitt	NA
FComb	13%
MComb	9%
Specacf	NA

As shown in Table 4-12, pitches estimated by YinFFT are at the lowest distance from the ground truth based on the percentage measure, while FComb has the furthest distance.

4.2.3.4 A discussion of the three methods

The three methods for finding the acceptable range will be evaluated in this part. To compare the calculated ranges by each of the three methods, if the difference of the ranges is equal to or bigger than a semitone, as the smallest pitch difference in Western music, is considered a perceptible difference; otherwise, the difference is not significant.

As can be seen in Table 4-10 and Table 4-11, overall, there is no significant difference between the fixed distance around ground truth F0 and Standard Deviation methods. The most noticeable difference is observed with the Yin algorithm, with a range of $\pm 35\ Hertz$ as compared to one of $\pm 27.05\ Hertz$ for the distance and standard deviation techniques, respectively. Since most musical pitches differ by much more than 8Hz, this cannot be a perceptible difference. Therefore, these two approaches can be considered to be similar.

Figure 4-4 to Figure 4-8 show the acceptable range of each algorithm for the three methods using an example. It is assumed that the method that exhibits a narrower range around the ground truth would be deemed as being the best approach.

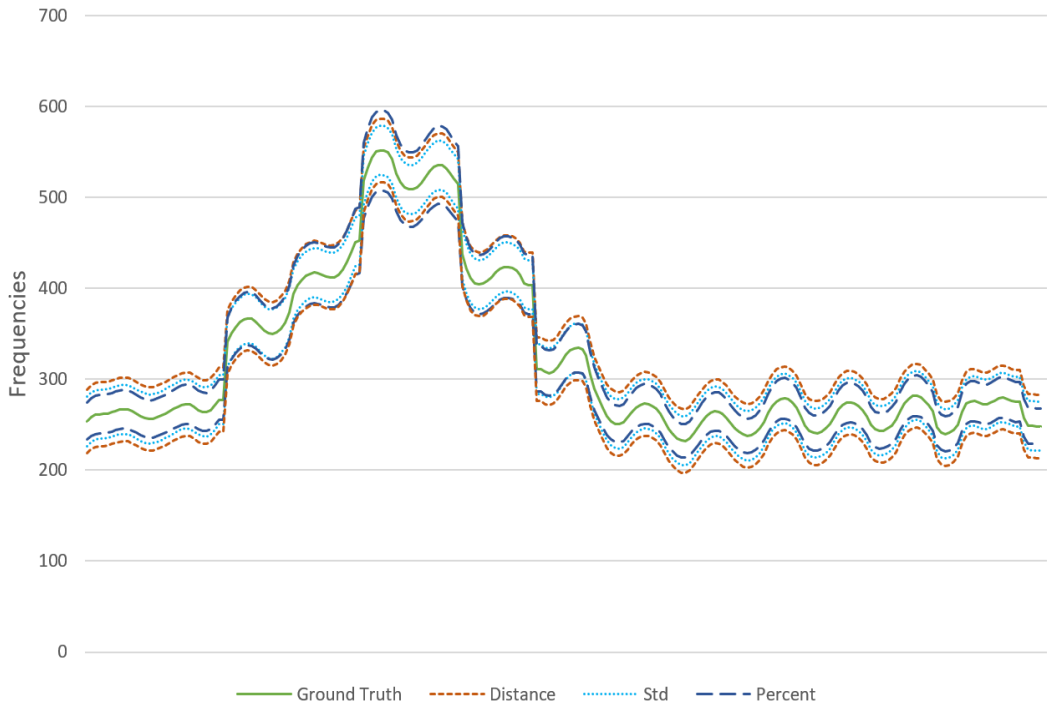


Figure 4-4. The acceptable range for the Yin algorithm in the three methods: Distance, Standard Deviation, and Percentage. Each colour shows the acceptable range by each algorithm.

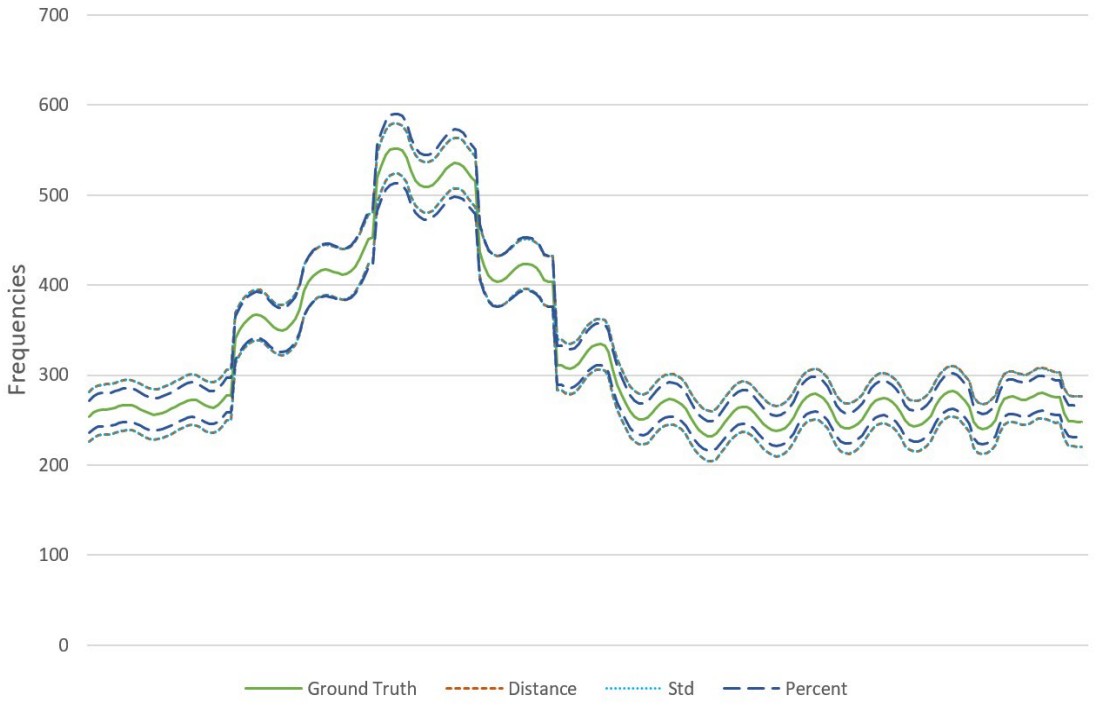


Figure 4-5. The acceptable range for the YinFast algorithm in the three methods: Distance, Standard Deviation, and Percentage. Each colour shows the acceptable range by each algorithm.

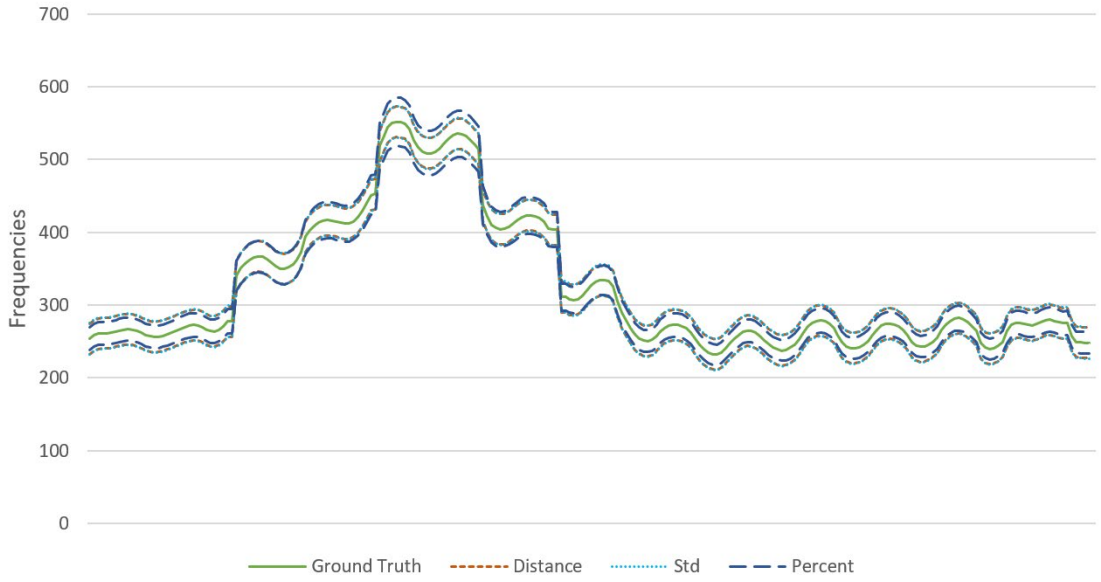


Figure 4-6. The acceptable range for the YinFFT algorithm in the three methods: Distance, Standard Deviation, and Percentage. Each colour shows the acceptable range by each algorithm.

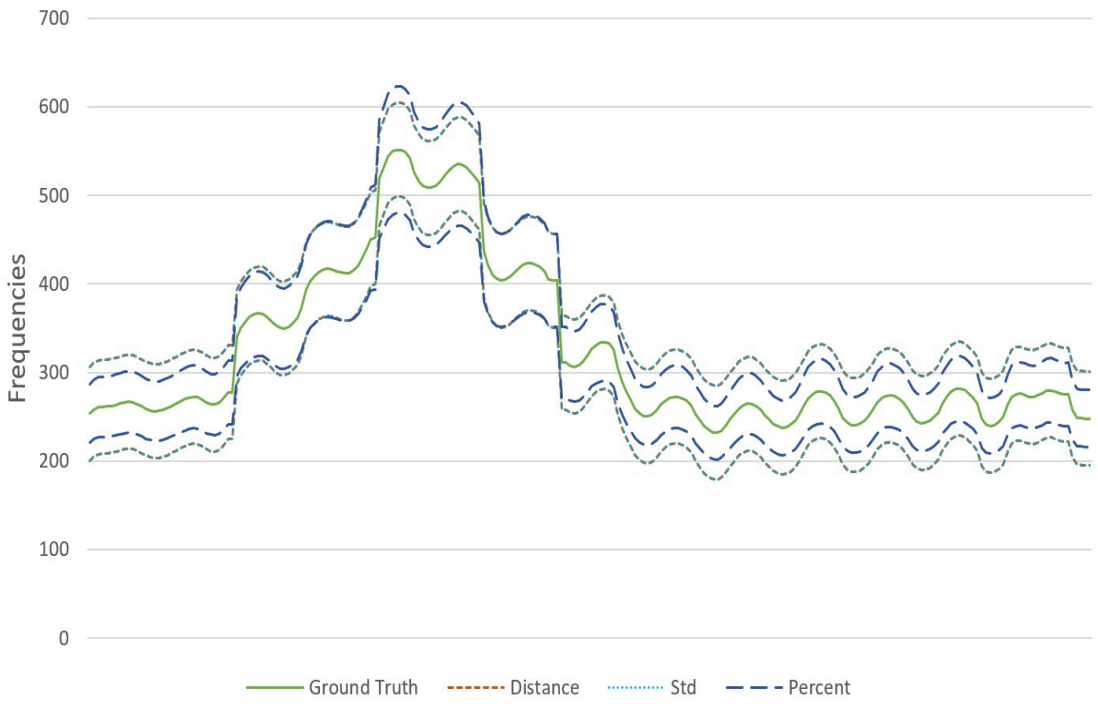


Figure 4-7. The acceptable range for the FComb algorithm in the three methods: Distance, Standard Deviation, and Percentage. Each colour shows the acceptable range by each algorithm.

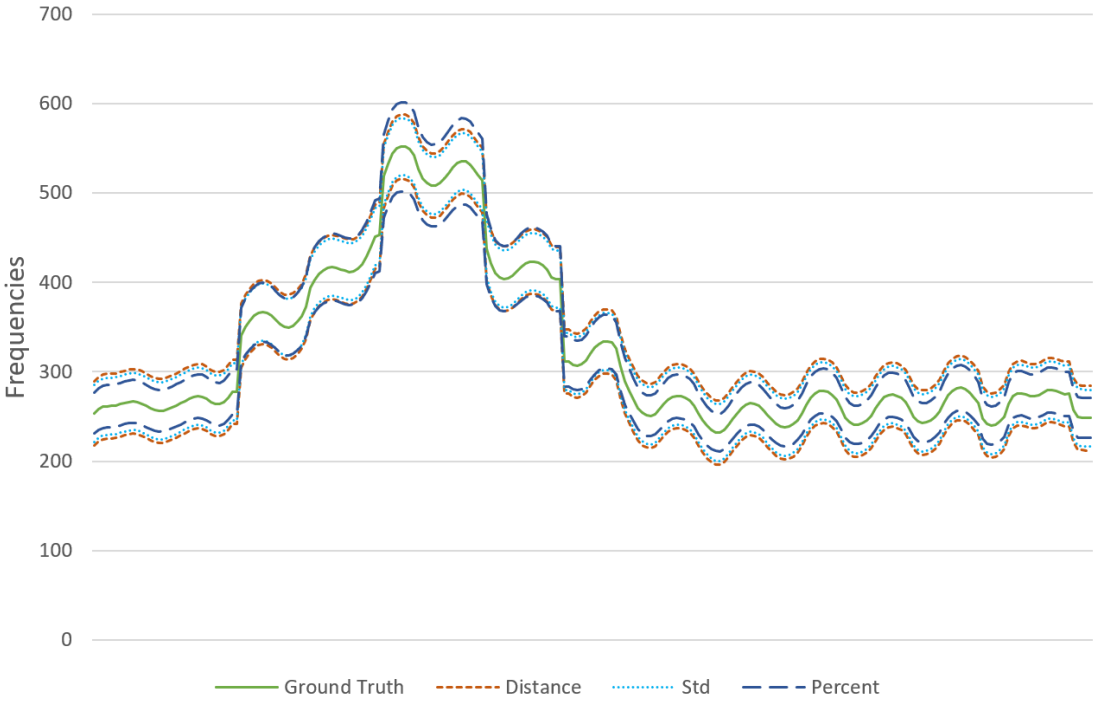


Figure 4-8. The acceptable range for the MComb algorithm in the three methods: Distance, Standard Deviation, and Percentage. Each colour shows the acceptable range by each algorithm.

Since the Distance and Standard Deviation methods are not based on intonation, they cover the same range of frequencies for higher and lower pitches. However, the range of the Percentage method will become narrow or wide when the frequencies are low or high.

Therefore, as can be observed from Figure 4-4 to Figure 4-8, it seems that it is better to use the Percentage method for frequencies that are less than a pitch frequency of 350 Hz, and the Distance or Standard Deviation method for frequencies that are greater than 350 Hz.

It is worth noting that although if the pitch frequencies become higher, the acceptable range with the Percent method will become very wide, the humans' voice pitch range is narrow and only in some exceptional cases a very wide range will be calculated by the Percent method.

Besides, from Table 4-10 to Table 4-12 and Figure 4-4 to Figure 4-8, it can be concluded that the acceptable range will not become wider or narrower with the Distance and Standard Deviation methods by changing the pitch frequency.

De Obaldía and Zölzer (de Obaldía and Zölzer, 2019) selected error margins of 8%, 20%, and 10 Hz with respect to the ground truth, and they achieved their best results for the 20% error, which may support our results that the fixed distance should be more than 10 Hz. Similarly, some other studies (Drugman and Alwan, 2011; Jouvet and Laprie, 2017) considered that any more than a 20% difference from the ground truth could be deemed to be an incorrectly estimated pitch.

4.3 Conclusions

In this chapter, after preparing a dataset of estimated pitches from seven real-time pitch detection algorithms, the functionality of each algorithm was evaluated based on 1- the number of pitches estimated correctly by categorising them based on gender, window size, the speed of the music, and post-processing, 2- the delay of each algorithm to estimate pitches correctly, and 3- the approaches to evaluate the accuracy of the estimated F0.

Finally, three methods for finding an acceptable range were evaluated. Generally, this chapter provides guidance for selecting a real-time pitch detection algorithm for singing signals according to the features of the sung. Based on all the evaluations, the following conclusions could be made:

The overall best real-time algorithm from the seven tested algorithms for female voices was YinFFT, with a window size of 1024. In addition, the speed of performance is not an issue. Moreover, the delay before starting to determine the correct pitches is 25 ms.

The best real-time algorithm from the seven tested algorithms for male voices was Yin when the notes are playing fast, and in slow performance, the best one was YinFFT. The algorithms produced a more accurate pitch contour with a window size of 2048 compared to a window size of 1024. The delay before finding the correct pitches for the Yin algorithm is 107 ms, and for YinFFT is 71 ms.

The length of the intervals between notes does not impact the pitch accuracy of the delay.

The best method from the three presented methods to find the acceptable range for all the algorithms is the percentage, although, for FComb and MComb, significant differences between the three methods were not observed.

As discussed above, the pitch detector algorithms cannot estimate the pitches without errors, and the estimated pitch contours require some alternations. Thus, the next chapter will discuss the Smart-Median, mentioned already in this chapter, in detail.

Chapter 5

Pitch contour smoother

This chapter introduces a novel algorithm for smoothing estimated pitch contour from singing signals. In addition, the algorithm will be compared with 15 different contour smoother algorithms. This chapter entirely come from one of our journal publications listed in the following.

- Faghih, Behnam & Timoney Joseph, Smart-Median: A New Real-Time Algorithm for Smoothing Singing Pitch Contours. Applied Sciences. 2022; 12(14):7026. https://doi.org/10.3390/app12147026

As seen in the previous chapters, chapters 3 and 4, the pitch detector could not estimate the pitch contours of singing signals in real-time without any error. Thus, the estimated pitch contours need to be smooth to alter the incorrectly estimated F0s. Therefore, this chapter introduces a new contour-smoother algorithm based on the features and applications of pitch contours derived only from singing. For this purpose, after explaining several typical contour-smoother algorithms, the methodology applied will be described. Then, the proposed algorithm is explained in Section 5.3, followed by the results and discussion. Finally, a conclusion is provided in Section 5.6.

5.1 Current Contour Smoother Algorithms

Several contour-smoother algorithms are commonly used to smooth pitch contours. This section provides a list of these algorithms. To refer to the smoother algorithms within this chapter, a code has been assigned to each algorithm listed in Table 5-1. In addition, this table indicates which of the considerations listed in section 2.2.2.1 are deemed by the algorithms. As the table shows, only the Smart-Median algorithm, described in section 5.3, counts all the considerations.

Table 5-1. Code of each of the contour smoother algorithms with indicating the code(s) of their considerations according to the list in section 2.2.2.1

Code	Algorithm	The Considerations
00	Smart-Median	All
01	Gaussian (sigma = 1)	2
02	Savitzky–Golay filter	2
03	Exponential	2
04	Window-based (window_type = ‘rectangular)	2
05	Window-based (window_type = ‘hanning’)	2
06	Window-based (window_type = ‘hamming’)	2
07	Window-based (window_type = ‘bartlett’)	2
08	Window-based (window_type = ‘blackman’)	2
09	Direct Spectral	2
10	Polynomial	2
11	Spline (type = ‘linear_spline’)	2
12	Spline (type = ‘cubic_spline’)	2
13	Spline (type = ‘natural_cubic_spline’)	2
14	Gaussian (sigma = 0.2, n_knots = 10)	2

15	Binner	2
16	LOWESS	2
17	Decompose (type = ‘Window-based’, method = ‘additive’)	2
18	Decompose (type = ‘lowess’, method = ‘additive’)	2
19	Decompose (type = ‘natural_cubic_spline’, method = ‘additive’)	2
20	Decompose (type = ‘natural_cubic_spline’, method = ‘multiplicative’)	2
21	Decompose (type = ‘lowess’, method = ‘multiplicative’)	2
22	Decompose (type = ‘natural_cubic_spline’, method = ‘multiplicative’)	2
23	Kalman (component = ‘level’)	2
24	Kalman (component = ‘level_trend’)	2
25	Kalman (component = ‘level_season’)	2
26	Kalman (component = ‘level_trend_season’)	2
27	Kalman (component = ‘level_longseason’)	2
28	Kalman (component = ‘level_trend_longseason’)	2
29	Kalman (component = ‘level_season_longseason’)	2
30	Kalman (component = ‘level_trend_season_longseason’)	2
31	Moving Average (simple = True)	2
32	Moving Average (simple = False)	2
33	Median Filter	2
34	Okada Filter	1, 2
35	Jlassi Filter	1, 2, 7

Figure 5-1 illustrates the effect of the smoother algorithms on a single estimated pitch contour. A female singer sang an arpeggio in the C major scale, and the FComb algorithm estimated the pitches. The smoothed contours are plotted in eight different panels. Each panel includes ground truth (GT), the original estimated (ST) contours, and the smoothed contours generated by some of the smoother algorithms.

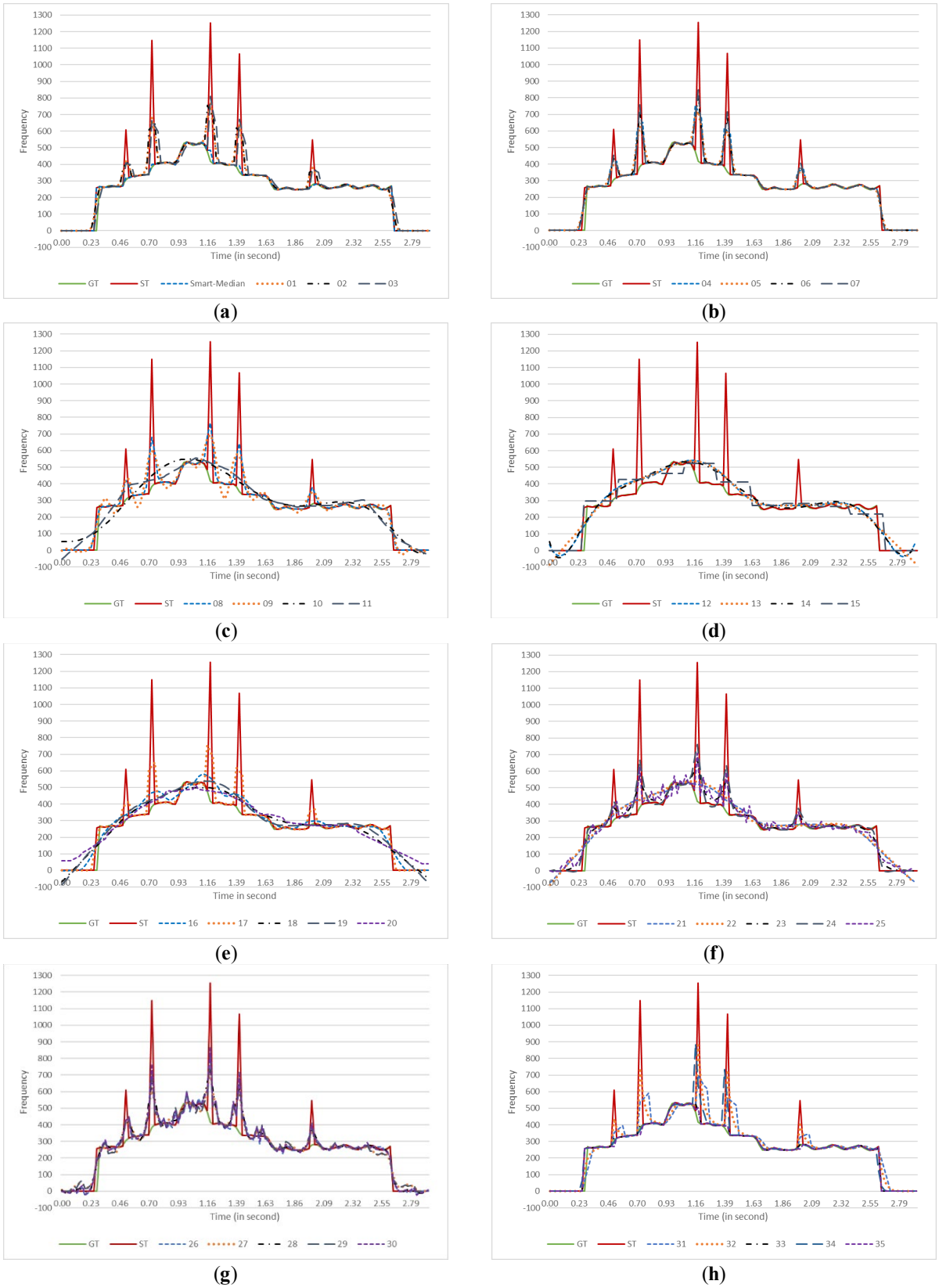


*Figure 5-1. The effect of each contour-smoother algorithm on a pitch contour from a female singer producing arpeggios in the C major scale. The pitch estimator algorithm was FComb. GT = Ground Truth (PYIN), ST = Estimated pitch contour. The smoothed contours are plotted in parts (a–h) for more straightforward observation. Each panel (**a**–**f**) plots three smoothed contours, while panels (**g**,**h**) have four contours each. Descriptions of the algorithms' codes are provided in Table 5-1.*

In addition, the Python libraries employed to implement these smoothers are listed in Table 5-2.

Table 5-2. Python libraries used for smoothing pitch contours.

Python Library	Smoother Algorithm
TSmoothie[1]	Exponential, Window-based (Convolution), Direct Spectral, Polynomial, Spline, Gaussian (code 14), Lowess, Decompose, Kalman
Scipy (Virtanen *et al*., 2020)	Savitzky–Golay filter, Gaussian (code 01), Median
Pandas (Reback *et al*., 2020)	Moving average

Each of the algorithms is described below.

5.1.1 Gaussian Filter

Generally, in signal processing, filtering removes or modifies unwanted error and noise signals from a series of data. Therefore, Gaussian filters smooth out fluctuations in data by convolution with a Gaussian function (Deng and Cahill, 1994). The one-dimensional Gaussian filter is expressed as (5-1):

$$Sm_i = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(Es_i)^2}{2\sigma^2}\right) \quad (5\text{-}1)$$

where Es_i is the original signal at position i, and Sm_i is the smoothed signal at position i. In addition, σ^2 indicates the variance of the Gaussian filter. The smoothing degree depends on the variance value size (Deng and Cahill, 1994). Although the Gaussian filter smooths out the noise, as shown in Figure 5-1(a), some correctly estimated F0 may also change, i.e., become distorted (Deng and Cahill, 1994).

5.1.2 Savitzky–Golay Filter

This particular type of low-pass filter was introduced into analytical chemistry but soon found many applications in other fields (Savitzky and Golay, 1964). It can be considered a weighted moving average (Dai *et al.*, 2017), and is defined as follows (5-2):

[1] https://pypi.org/project/tsmoothie/, accessed on 1 February 2022

$$Sm_i = \sum_{k=-M}^{M} h_k \, Es_{i-k} \tag{5-2}$$

where Es_i is the original signal at position i, and Sm_i is the smoothed signal at position i. M is window length and h_k are the filter coefficients that indicate the boundaries of the data. Specifically, this filter defines Sm_0 (i.e., the output at time $i = 0$) to be the value of the coefficient of a polynomial of order K that best fits the time series data Sm_i over the interval $|i| \leq M$ (Dai *et al.*, 2017). The drawback of the Savitzky–Golay (SG) filter, according to Schmid et al. (Schmid, Rath and Diebold, 2022), is that the data near the edges is prone to artefacts. Figure 5-1(a) illustrates its effect on a contour that this filter reduces the sharpness of the errors, but they still exist.

5.1.3 Exponential Filter

This approach is based on weighting the current values by the previously observed data, assuming that the most recent observations are more important than the older ones. The smoothed series starts with the second point in the contour. It is calculated by (Rej, 2003), (5-3):

$$Sm_i = \alpha Es_{i-1} + (1 - \alpha) \, Sm_{i-1} \qquad 0 < \alpha \leq 1 \qquad \text{i} \geq 3. \tag{5-3}$$

where α is called the smoothing constant. This filter demonstrated a similar alternation as Savitzky–Golay filter on the pitch contour shown in Figure 5-1(a).

5.1.4 Window-Based Finite Impulse Response Filter

In this approach, a window works as a mask to filter the data series. Different window shapes can be considered for filtering data. Each window point is usually between 0 and 1. Therefore, this method uses weighted windows. If Es_i is considered a signal at index i, and a window at index i as w_i, the smoothed signal Sm_i is calculated as follows (5-4):

$$Sm_i = w_i \, Es_i \tag{5-4}$$

The window types used in this study are described below.

5.1.4.1 Rectangular Window

This means that the window's values are all equal to one; Figure 5-1(b).

5.1.4.2 Hanning Window

The Hanning window is defined as follows according to Branu (Braun, 2001)(5-5):

$$W_H(i) = \begin{cases} 0.5\left[1 - \cos\left(2\pi\frac{i}{N}\right)\right] & 0 \le i = N-1 \\ 0 & otherwise \end{cases} \quad (5\text{-}5)$$

where N is the length of the window; Figure 5-1(b).

5.1.4.3 Hamming Window

The Hamming window is defined as follows according to Branu (Braun, 2001) (5-6):

$$w_{HM}(i) = \begin{cases} 0.54 + 0.46\cos\left(2\pi\frac{i}{N}\right) & 0 \le n \le N-1 \\ 0 & otherwise \end{cases} \quad (5\text{-}6)$$

where N is the length of the window; Figure 5-1(b).

5.1.4.4 Bartlett Window

The Bartlett window is defined (Braun, 2001) using (5-7), Figure 5-1(b):

$$w_b(i) = \begin{cases} 1 & 0 \le i \le N-1 \\ 0 & otherwise \end{cases} \quad (5\text{-}7)$$

5.1.4.5 Blackman Window

The Blackman window is defined (Podder *et al.*, 2014) by (5-8):

$$w_{black}(i) = a_0 + a_1 + a_2\cos\frac{4\pi i}{N-1} \qquad for\ -\frac{N-1}{2} \le i \le \frac{N-1}{2} \quad (5\text{-}8)$$

where N is the window length, and a_0, a_1 and a_2 are constants (5-9):

$$a_0 = \frac{1-\alpha}{2}, a_1 = \frac{1}{2}, a_2 = \alpha/2 \quad (5\text{-}9)$$

The α is static and equals 0.16; Figure 5-1(c).

5.1.5 Direct Spectral Filter

In this approach, a time series is smoothed by employing a Fourier Transformation. The essential frequencies remain, and others are removed. It operates similarly to multiplying the frequency domain by a rectangular window. In other words, it is a circular convolution generated by transforming the window in the time domain; Figure 5-1(c).

5.1.6 Polynomial

This approach uses weighted linear regression on an ad-hoc expansion basis to smooth the time series. It is a generalization of the Finite Impulse Response (FIR) filter that can better preserve the desired signal's higher frequency content without removing as much noise as the moving average (Orfanidis, 2018). The first derivative of the polynomial evaluated at the midpoint of the *N*-interval is generated by multiplying the position data Es_i by coefficients and adding these multiplications, as shown in (5-10) (Luers and Wenning, 1971):

$$Sm_{(N+1)/2} = \sum_{i=1}^{N} W_i Es_i \quad (N = \text{number of data points(odd)}) \tag{5-10}$$

where W_i are the weights (coefficients) of the polynomial fit of degree p. The weights depend on the degree p, and the number of points, *N*, used in the fit; Figure 5-1(c) is an example. As can be seen from the plot, this approach shows the data trend.

5.1.7 Spline

This approach employs Spline functions to eliminate the noise from the data. It works by estimating the optimum amount of smoothing required for the data. Three types of spline smoothing were used in this study: 'linear' (Figure 5-1(c)), 'cubic' (Figure 5-1(d)), and 'natural cubic' (Figure 5-1(d)). The details of this approach are provided in (Craven and Wahba, 1978; Hutchinson and de Hoog, 1985).

5.1.8 Binner

This approach applies linear regression on an ad-hoc expansion basis within a time series. The features created by this method are obtained by binning the input space into intervals. An indicator feature is designed for each bin, indicating into which bin a given observation falls. The input space consists of a single continuous increasing sequence in the time series domain (Jones, 1995); an illustration is shown in Figure 5-1(d). As can be seen in the plot, the altered pitch contour is not aligned well with the ground truth.

5.1.9 Locally Weighted Scatterplot Smoothing (LOWESS) Smoother

This is a non-parametric regression method. LOWESS attempts to fit a linear model to each data point based on local data points; Figure 5-1(e). This makes the procedure more versatile than simply including a high-order polynomial (Cleveland, 1979, 1981).

5.1.10 Seasonal Decomposition

One of the considerations in analysing time series data is dealing with seasonality. A seasonal decomposition deconstructs a time series into several components: a trend, a repeating seasonal time series, and the remainder. One of the benefits of seasonal decomposition is its capacity to locate anomalies and errors in data (Wen *et al.*, 2020). Seasonal decomposition can estimate the notes and transitory in a pitch contour, but the vibrations sung in each note are removed. Therefore, it can show the movements between changes and notes in a pitch contour, as shown in Figure 5-1(e, f).

Two component assessments that would be interpreted as seasonal by the algorithm are: 'additive' and 'multiplicative'. In the additive method, the variables are assumed to be mutually independent and calculated by summation of the variables. The multiplicative approach considers that components are dependent on each other and is calculated by the multiplication of the variables (Dagum, 2010).

Seasonal decomposition can be employed using different smoothing techniques. The smoothing techniques used in this study are Window-based, 'LOWESS', and 'natural_cubic_spline'.

5.1.11 Kalman Filter

The Kalman filter is a set of mathematical equations that provides an efficient recursive means to estimate the state of a process in a way that minimises the norm of the squared error. The Kalman filter uses a form of feedback control, assessing the process state and then obtaining feedback in the form of (noisy) measurements. The equations for the Kalman filter have two parts: time update equations and measurement update equations. The time update equations operate as predictor equations, while the measurement update equations are corrector equations. Thus, the overall estimation algorithm is close to a predictor-corrector

algorithm, i.e., correcting to improve the predicted value. In the standard Kalman filter, it is assumed that the noise is Gaussian, which may or may not reflect the reality of the system that is being modelled (Welch, 2021). Thus, the more accurate the model used in the Kalman algorithm, the better the performance.

The Kalman smoother can be represented in the state space form. Therefore, a matrix representation of all the components is required. Four structure presentations in the contours are considered: 'level', 'trend', 'seasonality' and 'long seasonality', and a combination of these structures can be considered. Examples of the effects of different variations of the Kalman filter are shown in Figure 5-1(f,g). Generally, as seen from the plots, although the Kalman filter reduced the sharpness of the existing errors, it also created some new small errors.

5.1.12 Moving Average

This simple filter aims to reduce random noise in a data series (Smith, 1999) by following the formula (5-11):

$$Sm_i = \frac{1}{n}\sum_{j=0}^{n-1} Es_{i+j} \tag{5-11}$$

where Es is the original pitch contour, Sm is the smoothed pitch contour, and n is the number of points analysed at any given time and is referred to as the window length of the filter. The larger the value of *n*, the greater the level of smoothing. An example can be seen in Figure 5-1(h) that the moving average not only could smooth the contour but also distorted some of the correctly estimated pitches.

5.1.13 Median Filter

The Median filter approach is similar to the moving average. Still, instead of calculating the average of a window of length *n*, the Median of the window is considered (5-12). Unlike the moving average filter, which is a linear system, this filter is nonlinear, rendering a more complicated analysis:

$$Sm_i = Median\,(Es_i, Es_{i+1}, Es_{i+2}, \dots, Es_{i+n-2}, Es_{i+n-1}) \tag{5-12}$$

where Es is the original pitch contour, Sm is the smoothed pitch contour, and n is the number of points to calculate the Median at each instant. Figure 5-1(h) illustrates the effect of this method on a pitch contour that could smooth it very well.

5.1.14 Okada Filter

This filter is a combination of moving average and Median filters. This filter aims to remove the outliers from a contour while closely retaining its shape, and not incurring any softening of the contour definition at transitions typically observed with smoothing. Each of the estimated points Es_i in a contour is compared with its immediate previous and successive points, Es_{i-1} and Es_{i+1}, respectively. If Es_i is the median of Es_{i-1}, Es_i, and Es_{i+1}, then it does not need to be changed; otherwise Es_i will be replaced by the average of ES_{i-1} and Es_{i+1}, as shown in (5-13). In this case, the first and the last point will not be changed (Okada, Ishikawa and Ikegaya, 2016).

$$Sm_i = Es_i + \frac{Es_{i-1} + Es_{i+1} - 2Es_i}{2(1 + e^{-\alpha(Es_i - Es_{i-1})(Es_i - Es_{i+1})})} \tag{5-13}$$

When α, weight, is sufficiently large, it can perform two operations: (1) if $(Es_i - Es_{i-1})(Es_i - Es_{i+1}) \leq 0$, Es_i is assigned to Sm_i; and (2) if $(Es_i - Es_{i-1})(Es_i - Es_{i+1}) > 0$, Sm_i is assigned by $(Es_{i-1} + Es_{i+1})/2$.

Figure 5-1(h) exemplifies the impact of this algorithm and can be regarded as an example of effective smoothing based on the contour.

5.1.15 Jlassi Filter

This technique was presented by Jlassi et al. (Jlassi, Bouzid and Ellouze, 2016). This approach has two main steps; first, finding the incorrect points in the pitch contour by considering those that exhibit a difference of more than a set threshold from both their previous and successive points. Second, replacing the incorrect point with the average of the last two points (5-14):

$$Sm_i = \begin{cases} \frac{Es_{i-2} + Es_{i-1}}{2} & |Es_i - Es_{i-1}| > Threshold \quad and \quad |Es_i - Es_{i+1}| > Threshold \\ Es_i & otherwise \end{cases} \tag{5-14}$$

The value for $Threshold$ is assumed to be 30, as mentioned in the original paper. Figure 5-1(h) illustrates the effect of the algorithm.

5.2 Materials and Methods

5.2.1 Dataset

The VocalSet dataset (Wilkins *et al.*, 2018), as described in section 3.1.1, was used to evaluate the algorithms' accuracy. This dataset includes more than ten hours of recordings of 20 (11 males and nine females) professional singers. VocalSet includes a complete set of vowels and a diverse set of voices that exhibit many different vocal techniques, singing in contexts of scales, arpeggios, long tones, and melodic excerpts. For this study, a portion of VocalSet was selected; the scales and arpeggios were sung across the vowels in loud slow and fast performances. The total number of files used from VocalSet was 511.

5.2.2 Ground Truth

In order to evaluate the accuracy of each of the smoother algorithms, ground truth pitch contours were required to compare the smoothed pitch contours. In other words, in this study, the best smoothing algorithm was considered the one that produced contours most similar to the ground truth. According to studies by Faghih and Timoney (Faghih and Timoney, 2019a, 2022a) and discussed in Chapter 3 and Chapter 4, a reliable offline pitch detector algorithm called PYIN (Mauch and Dixon, 2014) was used. The implementation of PYIN by a Python library called Librosa (McFee *et al.*, 2022) was used for this study. The pitch contours estimated by PYIN were saved in several CSV files with two columns, time in seconds and F0. These were all plotted to ensure the accuracy of the pitch contours estimated by PYIN. Those that included irrational jumps were considered incorrect and deleted. Therefore, after removing those contours, the number of ground truth files remaining was 447. It should be noted that removing these files did not make an unreal/easier dataset for the smoother algorithms because, as discussed in the following section, several pitch detection algorithms were used to generate different errors. Thus, this removal only secures the study with reliable ground truth files.

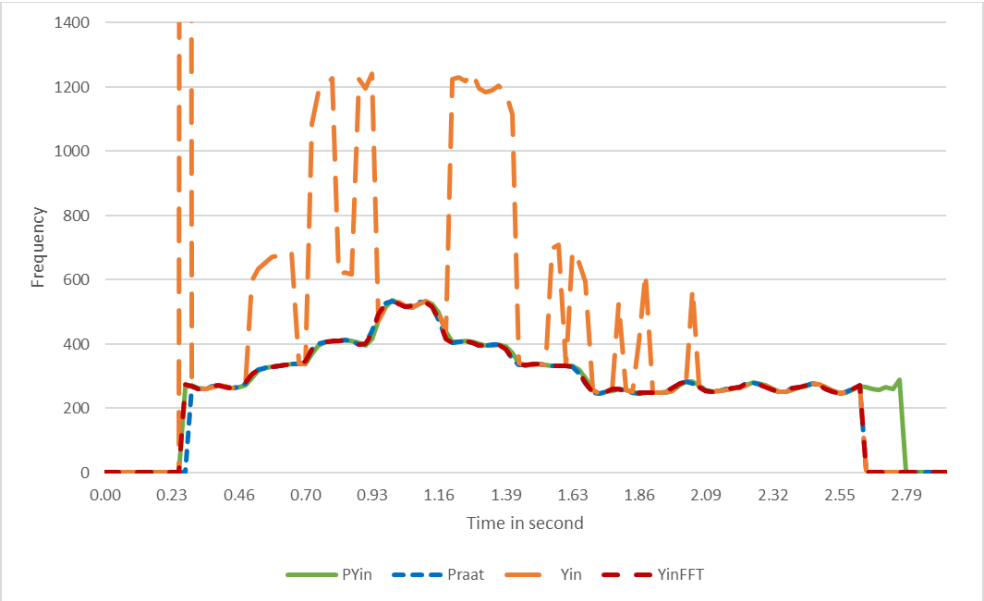
5.2.3 Pitch Detection Algorithms to Generate Pitch Contours

To evaluate the proposed smoother algorithm, we used a similar approach as (Ferro and Tamburini, 2019), employing several pitch contours with different random error (unsmoothed) points. As Faghih and Timoney's (Faghih and Timoney, 2022a) study discussed, also in Chapter 4, six real-time pitch detection algorithms with different estimated contours were employed to obtain the required contours. The pitch detector algorithms were Yin (de Cheveigné and Kawahara, 2002), spectral YIN or YIN Fast Fourier transform (YinFFT), Fast comb spectral model (FComb), Multi-comb spectral filtering (Mcomb), Schmitt trigger, and the spectral auto-correlation function (Specacf). The implementation for these algorithms came from a Python library, Aubio (https://aubio.org/manual/latest/cli.html#aubiopitch, accessed on 10 June 2021) (Aubio, no date), a well-known library for music information retrieval. Since this chapter focuses on smoothing pitch contours, descriptions of these algorithms are not provided in this chapter but in Chapter 4. The reason for selecting these real-time pitch estimators was that, based on Chapter 4, none of them can estimate F0s without error in singing signals. In addition, the accuracy of these algorithms varies, which helped us evaluate the contour-smoother algorithms in different situations.

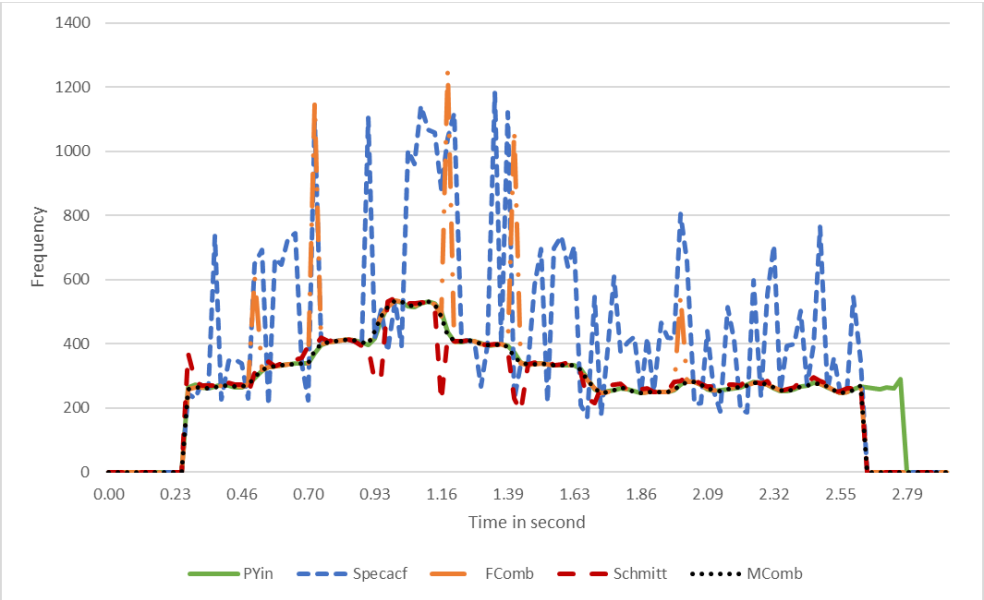
In addition, to compare the accuracy of the algorithms in conditions where the pitch contours included no or only a few errors, an offline pitch-detector algorithm provided in the Praat tool (Boersma and van Heuven, 2001) based on the Boersma algorithm [29] was used. According to Chapter 3, the Praat and PYIN accuracies tend to be similar.

The settings used for pitch detection for women's voices were 44,100 for sample rate, 1024 for window size, and 512 for hop size. The related settings for men's voices were 44,100 Hz, 2048, and 1024 for sample rate, window size, and hop size, respectively. Therefore, the distance between two consecutive points in a pitch contour for women's voices was 11.61 milliseconds, and for men's voices was 23.22 milliseconds.

As shown in Figure 5-2, the contours generated by the different pitch detectors exhibited various errors. Therefore, the total number of contours used to evaluate the smoother algorithms was 2682 (corresponding to the six pitch detectors run on each of the 447 wav files).



(a)



(b)

*Figure 5-2. Pitch contours for a female singer of arpeggios in the C scale. **(a)** pitch contour estimated by PYIN (ground truth), Praat, Yin, and YinFFT algorithms. **(b)** pitch contour estimated by PYIN (ground truth), Fcomb, Schmitt, Mcomb, and Specacf.*

All the provided files, such as the dataset and codes, are available in a GitHub repository[1].

5.2.4 Evaluation Method

Several evaluation metrics were used to compare the accuracy of the smoothing algorithms. The metrics used for the evaluations were R-squared (R^2), Root-Mean-Square

[1] https://github.com/BehnamFaghihMusicTech/Smart-Median, accessed on 11 July 2022

Error (RMSE), Mean-Absolute-Error (MAE), and F0 Frame Error (FFE). A well-known Python library called Sklearn (Buitinck *et al.*, 2013) was used for the metrics, except for the FFE metric that this thesis's author created. These metrics are explained in the following subsections.

5.2.4.1 R-Squared (R^2)

The formula for this metric is as follows (5-15) (Colin Cameron and Windmeijer, 1997):

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(GT_i - SM_i)^2}{\sum_{i=1}^{N}\left(GT_i - mean(GT)\right)^2} = 1 - \frac{\text{Regression Sum of Squares } (RSS)}{\text{Total Sum of Squares } (TSS)} \quad (5\text{-}15)$$

where N is the total number of frames, GT is the ground truth contour, SM is the smoothed contour, and the $mean(GT) = \frac{1}{N}\sum_{i=1}^{N} GT_i$.

In the best case, when all the points in the ground truth contour and the estimated contour are similar, R^2 is equal to 1; otherwise, R^2 is less than 1. A value closer to 1 means more similarity between the two contours.

5.2.4.2 Root-Mean-Square Error (RMSE)

This metric is calculated according to the following formula (5-16):

$$RMSE = \sqrt[2]{\frac{\sum_{i=1}^{N}(GT_i - SM_i)^2}{N}} \quad (5\text{-}16)$$

In the best case, when the two contours have precisely the same values, the RMSE is 0; otherwise, it is more significant than 0. Closer values to 0 mean more similarity between two contours.

5.2.4.3 Mean-Absolute-Error (MAE)

Equation (5-17) shows how to calculate this metric:

$$MAE = \frac{\sum_{i=1}^{N}|GT_i - SM_i|}{N} \quad (5\text{-}17)$$

MAE is similar to RMSE, but because of the squared difference, RMSE considers a more significant penalty for points at a greater distance from corresponding points in the ground truth contour.

5.2.4.4 F0 Frame Error (FFE)

FFE is the proportion of frames within which an error is made. Therefore, FFE alone can provide an overall performance measure of the accuracy of the pitch detection algorithm (Drugman and Alwan, 2011). This metric calculates the percentage of points in the estimated pitch contour that are within a $Threshold$ distance of corresponding points in the ground truth pitch contour (5-18):

$$FFE = \frac{\sum_{i=0}^{N} \begin{cases} 1 & \left|\frac{SM_i}{Gt_i}\right| \leq Threshold \\ 0 & otherwise \end{cases}}{N} \times 100 \quad (5\text{-}18)$$

where N is the total number of frames/points.

For the $Threshold$, in studies such as (Ferro and Tamburini, 2019), a constant value, e.g., 16 Hz, was used as an acceptable variation from the ground truth. However, as discussed in Chapter 4, a fixed distance from the ground truth may not be a good approach because the perceptual effect of 16 Hz is different when the estimated pitch is 100 Hz compared to 1000 Hz. However, it is also common to use a percentage, usually 20%, as the threshold (Jlassi, Bouzid and Ellouze, 2016), and the same approach is used in this study.

Higher values of this metric indicate a higher similarity between the smoothed and ground truth pitch contours.

The Sklearn library (Barupal and Fiehn, 2011) in Python has implemented the above evaluation metrics, except the FFE metric. This library was used for calculating the metrics.

It should be mentioned that there are other algorithms for finding the similarities between pitch contours, such as those of Sampaio (Sampaio, 2018), Wu (Wu, 2013), and Lin et al. (Lin, Wu and Kao, 2008). However, these aim to determine a perceptual similarity between two pitch contours. In other words, those researchers sought to determine one melody's similarity to another. However, the metrics could be used in this study that comparing two contours point by point instead of looking for the similarity between the trends of the two contours. Therefore, those algorithms were not suitable for this study.

5.3 Smart-Median: A Real-Time Pitch Contour Smoother Algorithm

The approach applied in this study to adjust the incorrectly determined pitch values was based on the Median method and has been named Smart-Median. The Smart-Median method is based on the belief that each contour should be smoothed based on its data features and intended applications. In other words, a general contour smoother may not be suitable for all applications. The considerations for designing the Smart-Median are given in section 2.2.2.1.

5.3.1 Smart-Median Algorithm

The flowchart shown in Figure 5-3 illustrates how incorrectly estimated pitches can be distinguished. In addition, it indicates which estimated pitches should be selected to calculate the median for the wrongly detected pitches.

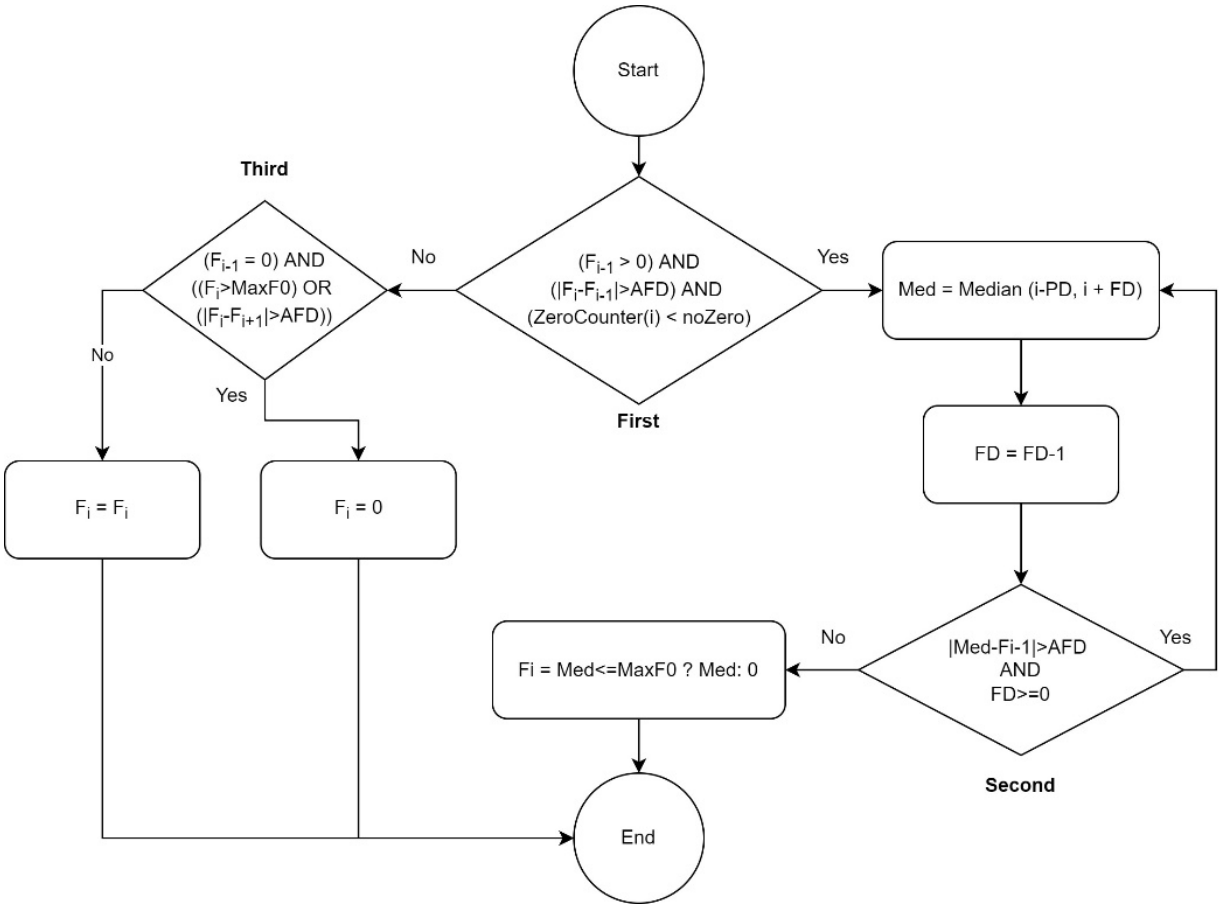


Figure 5-3. The central part of the Smart-Median algorithm for smoothing a pitch contour.

There are several variables and functions in the flowchart, explained as follows:

1) F_i refers to the frequency at index i.

2) AFD (Acceptable Frequency Difference) indicates the maximum pitch frequency interval acceptable for jumping between two consecutive detected pitches. In two studies on speech contour-smoother algorithms (Zhao, O'Shaughnessy and Nguyen, 2007; Jlassi, Bouzid and Ellouze, 2016), 30 Hz was selected as the AFD according to the researchers' experiences. Because the frequency range that humans use for singing is wider than for speaking, a larger AFD is needed for singing. According to the dataset used, the largest interval between two consequently notes sung by men was from C4 to F4, at frequencies of approximately 261 Hz and 349 Hz, respectively, so the maximum interval was 88 Hz for men. The largest interval between notes sung by women was C5 to F5, at frequencies of approximately 523 Hz and 698 Hz, respectively. Therefore, the biggest interval for women was 175 Hz. According to our observations of pitch contours, the human voice cannot physically produce such a big jump within a 30 ms timestep; i.e., for moving from C4 to F4 or from C5 to F5, more than 30 ms is needed. Therefore, it was found that an AFD with a value of 75 Hz was an acceptable choice for pitch contours comprised mostly of frequencies less than 300 Hz (male voices). For those with frequencies that are mostly greater than 300 Hz (female singers), 110 Hz was a good choice for AFD. To obtain these selections for AFD, the researcher manually annotated ten files and then calculated their AFD. Finally, the maximum AFD for the lower frequencies, men's voice, and higher frequencies, women's voice, were considered to be used to detect the events of the files.
3) noZero: this is the minimum number of consecutive zero pitch frequencies that should be considered as a correctly estimated silence or rest. In this study, 50 milliseconds was regarded as the minimum duration for silence to be accepted as correct (Kroher and Gomez, 2016); otherwise, the silence requires adjustment to the local median value.
4) The $ZeroCounter(i)$ method calculates how many frequencies (pitches) of zero value exist after index i. The reason for checking the number of zero values (silence) is to ascertain whether or not the pitch detector algorithm has estimated a region of silence correctly or in error.

5) $Median(i, \mathrm{j})$: calculates the median based on pitch frequencies from index i to index j.
6) PD (Prior Distance): this indicates how many estimated pitches before the current pitch frequency should be considered for the median. In this study, the PD was calculated to cover three estimated pitch frequencies, approximately over a duration of 35 and 70 milliseconds for men's and women's voices, respectively. Nevertheless, the algorithm does not need to wait until this duration becomes available, e.g., at a time of 20 milliseconds, covering 20 milliseconds with PD is sufficient.
7) FD (Following Distance): indicates how many estimated F0s after the current F0 should be considered for the median. In this study, the number three was assigned to FD, meaning that calculating the median of the current wrongly estimated pitch required 35 milliseconds for women's voices and 70 milliseconds for men's voices. Therefore, a buffer delay is required in real-time environments until three extra estimated F0s are available.
8) MaxF0: indicates the maximum acceptable frequency. In this study, for male voices, a value of 600 Hz (near to tenor) and for female voices, a maximum of 1050 Hz (soprano) were considered for MaxF0. Rarely, male and female voices may exceed these boundaries. However, if the singer's voice range is higher than these boundaries, a higher value can be considered for MaxF0.

The first condition in Figure 5-3 aims to calculate whether the frequency at index i is valid. There are three conditions for considering invalid estimates of pitch frequency. First, the previously estimated pitch should not be zero because, after a silence, there should naturally be a significant difference between the current pitch frequency and the rest. Second, the absolute difference between the currently estimated pitch and the previous one should be greater than the AFD. Finally, the number of consecutive zeros from the current index should be less than noZero. This condition checks whether the estimated F0 in the current index is zero, but the pitch detector error could not estimate the F0 correctly.

According to the above conditions, if the current estimated F0 is marked as incorrect estimation, it branches to the right to "Yes". The algorithm then continues by reducing the value of FD until the second condition is no longer true. In other words, the window for

calculating the median shrinks until the difference between the calculated median and the previous point is less than the AFD. Finally, the correct median is held in the Med variable. This should be less than the MaxF0 if it is considered a valid replacement value; otherwise, a zero will be substituted instead.

Since several incorrect estimated pitches have been observed after silences, the third condition in Figure 5-3 checks whether the estimated F0 immediately follows a silence. In this case, the difference between the current estimated F0 and the next estimated F0 is considered. If neither the first nor the third conditions are correct, the estimated F0 is assumed to be accurate and does not need to be changed.

The algorithm's source code is available from the GitHub repository mentioned above for more detail.

5.4 Results

This section provides the results of the comparisons between the Smart-Median algorithm and the other 35 contour smoothers mentioned in Section 5.1. Three groups of data were obtained for evaluation. These groups were 1—the ground truth pitch contour (GT), 2—the original estimated pitches (ES), and 3—the smoothed contour (SM). The metrics explained in Section 5.2.4 were employed to compare these data groups. The data series were compared two by two, i.e., GT with ES, GT with SM, and ES with SM.

Table A-1 to Table A-4 in the Appendix show the accuracy of each of the pitch detector algorithms, and the accuracy of the contour-smoother algorithms applied to the estimated pitch contours to bring them closer to the ground truth pitch contour. The GT–ES columns show the initial difference between the ground truth and the original estimated pitch contour. Next, the differences between the ground truth and the smoothed contours are shown in the GT–MS columns. Finally, the ES- SM columns compare the initially estimated pitch contour and the smoothed pitch contour. The metrics comparing GT and SM are more important than those comparing GT–ES and ES–SM, because the values of GT–SM illustrate the resulting improvement supplied by each algorithm. For example, in the Specacf column in Table A-4, the first row (smoother algorithm with code 00) shows that according to the FFE metric GT–ES = 40, GT–SM = 48, and ES–SM = 61. That is, 40 per cent of the pitches estimated by the

Specacf algorithm were correct. Then, the smoother algorithm improved this to 48 per cent of the acceptable data. Finally, 61 per cent of the values in the estimated pitch and smoothed contour remained in the same range; i.e., the smoother algorithm significantly changed just 39 per cent of the values.

According to Table A-1 to Table A-4 in the Appendix, the Smart-Median was the best algorithm for all pitch contours estimated by Specacf, FComb, Mcomb, Yin, or YinFFT. However, the best accuracy for the pitch contours calculated by Praat was recorded by the contour smoother code 33 (standard median). However, there was no agreement between the metrics employed to select the best smoother pitch contours generated by Schmitt or PYIN.

Table 5-3 aggregates all the data in Table A-1 to Table A-4 in the Appendix. It can be observed in Table 5-3 that all the metrics agree that the Smart-Median worked better than the other smoother algorithms.

Only the GT–SM column was considered to have found significant differences between the accuracy of the algorithms. All the algorithms in the range of the column average plus/minus standard deviation were considered to exhibit a similar accuracy. The algorithms with values outside this range were considered to be in the best or worst category, as shown in Table 5-4. Certain agreements and disagreements existed between the metrics employed to find the best and worst algorithms. For example, the smoother code 07 was in the worst category based on the metrics MAE and RMSE but in the best category based on the FFE metric. These agreements and disagreements are discussed in Section 5.4.

An ANOVA test was used to check the accuracy of the smoother algorithms. For all the metrics, the *p*-value calculated for each smoother algorithm was 0. That means that the accuracy of all the smoother algorithms depended on errors that occurred in the pitch contours, i.e., the smoother algorithms did not work with the same accuracy when each pitch contour was affected by different sources of error.

Table 5-3. Comparing the mean of pitch estimators and contour-smoother algorithms by ground truth based on the four metrics. GT = Ground Truth, ES = Estimated pitch contour, SM = Smoothed contour.

Algorithm	MAE			R^2			RMSE			FFE		
	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM
00	165	**59**	136	−175	**−1**	0.4	451	**91**	426	71	**75**	84
01	165	160	76	−175	−73	0.9	451	313	240	71	64	81
02	165	161	82	−175	−81	0.8	451	327	278	71	66	85
03	165	160	81	−175	−82	0.8	451	327	264	71	67	85
04	165	161	82	−175	−81	0.8	451	327	278	71	66	85
05	165	160	85	−175	−68	0.8	451	304	265	71	65	82
06	165	161	69	−175	−79	0.9	451	324	224	71	66	84
07	165	161	62	−175	−87	0.9	451	338	209	71	66	87
08	165	160	76	−175	−74	0.9	451	315	242	71	65	83
09	165	191	127	−175	−77	0.8	451	321	296	71	51	64
10	165	181	179	−175	−32	0.5	451	228	397	71	45	51
11	165	172	153	−175	−39	0.7	451	240	367	71	50	59
12	165	175	153	−175	−43	0.7	451	248	361	71	50	59
13	165	172	158	−175	−34	0.6	451	228	377	71	48	57
14	165	178	162	−175	−40	0.6	451	239	368	71	49	57
15	165	168	152	−175	−39	0.6	451	241	379	71	55	65
16	165	161	130	−175	−42	0.7	451	243	345	71	56	67
17	165	161	82	−175	−81	0.8	451	327	278	71	66	85
18	165	163	160	−175	−26	0.6	451	210	384	71	48	56
19	165	172	158	−175	−34	0.6	451	228	377	71	48	57
20	165	172	158	−175	−34	0.6	451	228	377	71	48	57
21	168	164	147	−184	−32	0.6	448	220	366	70	50	60
22	165	172	158	−175	−34	0.6	451	228	377	71	48	57
23	165	159	101	−175	−47	0.8	451	262	282	71	53	67
24	165	164	82	−175	−72	0.9	451	312	246	71	55	71
25	165	176	120	−175	−55	0.8	451	283	262	71	51	63
26	165	183	88	−175	−91	0.9	451	344	192	71	53	70
27	165	168	104	−175	−56	0.8	451	285	268	71	53	68
28	165	170	92	−175	−71	0.9	451	311	252	71	54	69
29	165	175	95	−175	−73	0.9	451	316	214	71	53	68
30	165	182	89	−175	−88	0.9	451	340	197	71	53	69
31	168	163	111	−199	−76	0.7	456	303	329	70	65	80
32	165	159	70	−175	−78	0.9	451	321	212	71	61	78
33	165	132	52	−175	−46	0.8	451	238	307	71	72	94
34	165	146	76	−175	−62	0.8	451	284	311	71	65	81
35	165	131	59	−175	−61	0.7	451	228	342	71	72	95

Table 5-4. Dividing the contour smoother algorithms into three categories (best, normal, and worst) based on the standard deviation.

	Best Code (Value)	Normal		Worst Code (Value)
MAE	00 (58.71) 33 (131.85) 35 (131.3)	Avg = 162.56 Min = 141.31 All the other algorithms*	Std = 21.25 Max = 183.81	09 (190.95)
R^2	00 (−0.72) 10 (−31.59) 13 (−34.07) 18 (−26.9) 19 (−34.7) 20 (−34.07) 21 (−32.46) 22 (−34.07)	Avg = −58.01 Min = −79.99 All the other algorithms*	Std = 21.98 Max = −36.03	02 (−80.79) 03 (−82.22) 04 (−80.81) 07 (−87.46) 17 (−80.81) 26 (−90.75) 30 (−87.66)
RMSE	00 (90.67) 18 (209.56) 21 (220.12)	Avg = 275.62 Min = 222.52 All the other algorithms*	Std = 53.1 Max = 328.72	07 (338.41) 26 (343.63) 30 (339.93)
FFE	00 (74.73) 02 (66.21) 03 (66.87) 04 (66.22) 07 (66.48) 17 (66.22) 33 (71.87) 35 (71.99)	Avg = 57.59 Min = 49.24 All the other algorithms*	Std = 8.35 Max = 65.94	10 (44.83) 13 (48.24) 14 (48.6) 18 (48.47) 19 (48.24) 20 (48.24) 22 (48.24)

*it means that all the other algorithms not mentioned in the Best or Worst columns were in the average range.

5.5 Discussion

This section discusses several aspects of the results obtained in Section 5.4. Because this study focuses on the Smart-Median method, the only considerations provided here are those relating to comparisons of Smart-Median accuracy with that of other smoother algorithms.

5.5.1 Comparing the Results of Each Metric

A higher R-squared (R^2) value does not always mean a better fitting (Lewis-Beck and Skalaban, 1990). For example, Table 5-5 shows the R-squared scores of three series of predicted data. These predictions are the estimated pitch frequencies for five sung notes: G2, G2, A2, G2, and G2. According to the R-squared scores in Table 5-5, the order of the best prediction to the worst was 4, 3, 2, and then 1. However, Predict 3 estimated two wrong notes, such that each was one tone above the corresponding ground truth notes (A2 instead of G2), while Predicts 1 and 2 each had only one incorrect note (B2 instead of A2). Therefore,

musically, the third was the worst, but based on R-squared, it was the second-best. In addition, musically, Predict 1 and Predict 2 were similar, and the 0.2 Hz pitch frequency difference could easily have resulted from a different method of F0 tracking, but their R-squared scores were different. In conclusion, we cannot compare two series of smoothed pitches based only on R-squared.

Table 5-5. Comparison of metrics in different series of predicted data.

	1st	2nd	3rd	4th	5th	R^2 Score	RMSE	MAE	FFE
Ground Truth	98 (G2)	98 (G2)	110 (A2)	98 (G2)	98 (G2)	NA	NA	NA	NA
Predict 1	98.2 (G2)	98.2 (G2)	123.2 (B2)	98.2 (G2)	98.2 (G2)	−0.61	5.91	2.8	0.8
Predict 2	98 (G2)	98 (G2)	123 (B2)	98 (G2)	98 (G2)	−0.56	5.81	2.6	0.8
Predict 3	98 (G2)	110 (A2)	110 (A2)	110 (A2)	98 (G2)	−0.33	7.59	4.8	0.6
Predict 4	98.2 (G2)	98.2 (G2)	110.2 (A2)	98.2 (G2)	98.2 (G2)	0.999	0.2	0.2	1

According to the RMSE and MAE columns in Table 5-5, the best to worst series were 4, 2, 1, and then 3. This order is better than that based on R-squared. However, musically, we need to consider the similarity of Predict 1 and Predict 2; based on the FFE column in Table 5-5, Predicts 1 and 2 both had the same value. As shown in Table 5-5, Predict 4 was the best according to all the metrics, and musically, it was also the best. Moreover, although Predicts 1 and 2 were musically similar (FFE metric), Predict 2 was more accurate than Predict 1 (R^2, RMSE, and MAE metrics).

To conclude, a single metric alone cannot provide a clear and accurate evaluation to compare pitch contours, but a firm conclusion can be reached by using all of them.

5.5.2 Comparing Moving Average, Median, Okada, Jlassi, and Smart-Median

The main weakness of the Median, Okada (Okada, Ishikawa and Ikegaya, 2016), and Jlassi (Jlassi, Bouzid and Ellouze, 2016) filters is that they only adjust noises with a duration of one point in the contour. In other words, if more than one consecutive wrongly estimated pitch values occurs within a contour, these algorithms cannot smooth the errors. The following example illustrates the operation of the moving average, Median, Okada, Jlassi, and Smart-Median approaches on a data series.

Table 5-6 shows five estimated F0s as the input, the first row, and how each of the Moving average, Median, Okada, Jlassi, and Smart-Median will modify the data. It was

expected that all these five numbers be close together. Thus, there are two incorrectly estimated F0s with the value 2000 Hz. As shown in Table 5-6, the moving average and Median methods changed some of the correctly estimated values, i.e., the 102 value, which was the second piece of input data. On the other hand, Okada's and Jlassi's approaches did not change any of the values because they looked for significant differences with immediately preceding and following points. However, the Smart-Median is mainly concerned with finding an acceptable jump by comparing the current and previous points. Because of this different approach to the identification of errors, when the pitch contour was already almost smooth (contours estimated by Praat and PYIN) there was no significant difference between the accuracy of these approaches (as seen by comparing rows 00, 33, 34, and 35 in Praat and PYIN columns in Table A-1 to Table A-4 in Appendix). However, while the pitch contours estimated by the other pitch detection algorithms exhibited several errors, Smart-Median showed meaningfully better results than all other methods (observable in Specacf, Schmitt, Fcomb, Mcomb, Yin, and YinFFT columns in Table A-1 to Table A-4 in Appendix).

Table 5-6. An example to illustrate the weakness of the moving average, Median, Okada, and Jlassi algorithms as compared to the Smart-Median.

Input	100 Hz	102 Hz	2000 Hz	2000 Hz	100 Hz
Moving average (window size = 3)	734 Hz	1367 Hz	1367 Hz	1050 Hz	100 Hz
Median (window size = 3)	102 Hz	2000 Hz	2000 Hz	1050 Hz	100 Hz
Okada	100 Hz	102 Hz	2000 Hz	2000 Hz	100 Hz
Jlassi	100 Hz	102 Hz	2000 Hz	2000 Hz	100 Hz
Smart-Median	100 Hz	102 Hz	102 Hz	102 Hz	100 Hz

Generally, according to Table 5-3, the accuracy of the Smart-Median based on the four metrics was much better than all the other algorithms.

5.5.3 Accuracy of the Contour Smoother Algorithms

All the contour smoother algorithms provided strong results according to the R2 and RMSE metrics (by comparing the GT–ES columns with GT–SM columns in Table 5-3). However, only the Smart-Median (00), Median (33), and Jlassi (35) approaches could change the pitch contour significantly to ensure that more of the estimated F0 values were constrained to line within the range of 20% of the ground truth pitch contour (Table A-1 to Table A-4 in Appendix

and Table 5-3 to Table 5-4). Therefore, although all the algorithms smoothed contour errors, many also altered the value of the corrected estimated pitches.

5.6 Conclusions

This chapter has introduced a new pitch-contour-smoother targeted towards the singing voice in real-time environments. The proposed algorithm is based on the median filter and considers the features of fundamental frequencies in singing. The algorithm's accuracy was compared with 35 other smoother techniques, and four metrics evaluated their results: R-Squared, Root-Mean-Square Error, Mean Absolute Error, and F0 Frame Error. The proposed Smart-Median algorithm achieved better results across all the metrics in comparison to the other smoother algorithms. According to this study, a buffer delay of 35 to 70 milliseconds is required for the algorithm to smooth the contour appropriately. For the low frequencies, men's voice, a longer buffer delay is needed than for higher frequencies, women's voice.

Most of the general smoother algorithms did not show an acceptable accuracy. A general observation is that in the ideal case, a smoother algorithm should be defined based on the essential features of the data in the contour and how that data is to be used after smoothing.

The parameters' values were selected according to the singing features and can also be used in other smoother algorithms, like the moving average, to improve their accuracy. The gender of the singer cannot directly affect the Smart-Median accuracy, albeit, as discussed in Chapter 3 and Chapter 4, it affects the pitch detectors' accuracy. After smoothing estimated pitch contours, the sung notes can be extracted with the novel algorithm discussed in the following chapter.

Chapter 6

Onset and Offset detection

This chapter introduces a new method for detecting onsets, offsets, and transitions of the notes in real-time solo singing performances. The text of this chapter come from the following journal paper.

- Faghih B, Chakraborty S, Yaseen A, Timoney J. A New Method for Detecting Onset and Offset for Singing in Real-Time and Offline Environments. *Applied Sciences*. 2022; 12(15):7391. https://doi.org/10.3390/app12157391

This chapter aims to introduce a new onset detection algorithm incorporating more knowledge about the singing features for a more accurate onset estimation.

To achieve the goal, the following section explains the methodology. After that, in Section 6.2, the new algorithm will be discussed in detail. Then, the evaluation results for the proposed algorithm will be presented and discussed in Section 6.3. Finally, the last section concludes the chapter.

6.1 Materials and Methods

This section explains the details of the approach taken to develop our algorithm. It first describes the datasets used, then explains the algorithm thoroughly, followed by the structure of the evaluation procedure.

6.1.1 Datasets

Two onset-annotated vocal datasets, Erkomaishvili (Rosenzweig, Scherbaum, *et al.*, 2020) and SVNote1 (Hoon Heo, Dooyong Sung and Kyogu Lee, 2013; Chang and Lee, 2014), are used for this study. The following paragraphs provide a summary description of these different musical datasets.

6.1.1.1 Erkomaishvili Dataset

This dataset includes 100 monophonic audio files of traditional Georgian vocal music performed by a professional singer, Artem Erkomaishvili. Each audio file contains the fundamental frequencies, segment annotation, onset annotations, and sheet music in XML. Moreover, it contains more than seven hours of music with 40,135 onset annotations. The annotations were estimated manually except for the fundamental frequencies, whose calculation was semi-automated. Moreover, in this dataset, the points for onset and offset in successive notes were deemed to coincide, i.e., the offset of the previous note is the onset of the new note. Since the files were recorded in 1966, the audio files have poor quality. In addition, the recordings are of natural melodic singing rather than only some scales or arpeggios. Therefore, it is a challenging dataset for automatic annotation algorithms.

6.1.1.2 Note-Level Singing Voice Dataset (SVNote1)

This dataset included 30 audio files sung by seven men and three women. Each of the singers recorded three popular pieces of music (1-“soft kitty, warm kitty, little ball of fur”, 2-“school bell”, and 3-“Twinkle, twinkle little star”). These are, in total, around 16 min of music with 1440 onset annotations. In addition, three people annotated each audio file's onsets separately, meaning three sets of annotations were provided for each audio file. The three annotators’ average is considered the ground truth for this study.

These datasets were selected to have a variety of singing techniques, intervals between notes, duration of the notes, and transition from one to another. For example, the Erkomaishvili dataset included many soft onsets, long notes, and vibrato as compared to the SVNote1 dataset. Thus, both datasets can provide various spectrograms of singing to evaluate onset detection algorithms in determining the onset of notes.

6.1.2 State of-the-Art Onset Detection Algorithms

To evaluate our proposed algorithm’s efficiency and accuracy, eight different onset detection algorithms were selected against which to compare the accuracy of the proposed algorithm. The implemented versions of the algorithms in Python were used. The algorithms were taken from implementations across four different Python libraries, namely Librosa (McFee *et al.*, 2015, 2020), Madmom (Böck *et al.*, 2016), Aubio (Brossier *et al.*, 2019), and Essentia (Bogdanov *et al.*, 2013a). The explanations of the algorithms are provided in the following by categorizing them based on the Python libraries.

6.1.2.1 Librosa

Librosa is a well-known library for sound analysis and feature extraction (McFee *et al.*, 2015). It has three different methods to estimate onsets. The first method (referred to as “LibRaw” in this chapter) locates the note onsets based on peaks in the onset strength envelope. The onset strength envelope is calculated by finding the spectral flux, which is the difference in power spectrum between two consecutive frames, applying a threshold, and returning a one-dimensional array representing the change in spectral energy for each frame. Then, based on the onset strength, it peaks where the energy is a minimum based on the heuristic described by Boeck et al. (Sebastian, Krebs and Schedl, 2012). The other two methods

rely on backtracking from the nearest preceding minimum energy point (Jehan, 2005). The second method (called "LibBt") works by backtracking using the onset strength profile, while the third method (named "LibBtRMS") depends on backtracking with the Root Mean Square (RMS) or amplitude value. All these three methods are offline; they have not been designed to work in real-time.

6.1.2.2 Madmom

This Python library provides two real-time onset detection methods (Böck *et al.*, 2012, 2016). The first approach (called "MadmomCNN" in this article) uses a Convolutional Neural Network model that is a real-time version of the model proposed by Schlüter & Böck (Schlüter and Böck, 2013) trained on 26,000 annotated onsets. The model was trained to predict percussive and harmonic onsets with a frame rate of 100 per second. Next, the spectral onset processor method detects the onsets from a logarithmically scaled audio signal representation based on the spectral magnitude and phase, which is referred to using the name "MadmomSF" in this chapter.

6.1.2.3 Aubio

This real-time library uses a window size of 2048 frequency samples to detect onsets (Brossier, 2005). In addition, Aubio sets a threshold value to mark quiet regions. Finally, it constructs a function based on successive spectral frames with a window size of 2048 and a hop size of 1024, meaning that the frame duration was approximately 23 ms for a 44,100 Hz sample rate. The dynamic thresholding and peak selections return the onset frames.

6.1.2.4 Essentia

This offline onset tracking method was used with its default values for the window and hop sizes, 1024 and 512, respectively, for a Hann window (Bogdanov *et al.*, 2013a). Therefore, the duration of each frame was roughly 11.61 ms. There are two approaches to this library. The first method, Essentia Onset HFC (EssHFC), uses a high-frequency content detection function (Masri and Bateman, 1996). The high frequency is calculated by multiplying the magnitude of each frame position (frequency) with the summation of the magnitudes of the spectral frame. The discrete spectrum of N unique points is formulated in Equation (6-1).

$$HFC = \sum_{i=0}^{N-1} i|X(i)|, \quad (6\text{-}1)$$

The second method, Essentia Onset Complex (EssCplx), uses a complex domain spectral difference function to identify significant changes in magnitude and phase (Bello *et al.*, 2004). This algorithm tries to identify significant energy changes on note onsets or the deviation of phase values within the phase spectrum caused due to pitch changes.

Finally, it should be mentioned that all these algorithms/libraries calculate only onsets and do not compute offsets or identify transitions.

6.1.3 The Methods for Evaluation

The accuracy of the proposed algorithm is evaluated by running the algorithms presented in Section 6.1.2 and the proposed algorithm on the datasets mentioned in Section 6.1.1. Then, the F-measure scores were calculated by the mir_eval Python library (Raffel *et al.*, 2014), and the results were ordered so that they could be compared with each other. As mentioned above, the onset points are not exact times but a range of acceptable times. Therefore, to calculate F-measure scores, each of the estimated onsets' points should be compared with a range of points around the ground truth points. Thus, six different window sizes (10, 50, 100, 150, 200, and 250 ms) were considered to calculate the F-measure scores. Furthermore, the F-measure scores' average, variance, and ANOVA were calculated to understand the results better.

6.2 The Proposed Algorithm

This algorithm is based on our observations following investigations that involved many singing pitch contours. From many of the plotted pitch contours, it was noticed that there is a noticeable trajectory change in the fundamental frequency when moving from one note to another. Therefore, the proposed algorithm is focused on evaluating the changes in a pitch contour to identify those meaningful changes that will signify onsets, offsets, and transitions.

The pitch contour is selected because it is a robust indicator of onset compared to other features. For example, Rabiner and Sambur (Rabiner and Sambur, 1975) looked to find significant changes in the sound energy contour to find the start and the end of an isolated utterance. Their approach is based on short-time energy and zero-crossing rate. However, although in the case of a silence existing between notes, as considered by Rabiner and Sambur

(Rabiner and Sambur, 1975), a noticeable change in amplitude contour is easy to see, it is difficult to rely on the amplitude contour as a feature when analyzing legato singing, as unpredictable variations can occur in the movement from one note to the next. In contrast, the fundamental frequency track is either erratic before the onset and then quickly becomes stable or moves smoothly from one value to the next in the case of legato singing, even when the consecutive notes are at the same pitch frequency. Thus, the proposed algorithm can be explained as seven main steps to find the onsets, offsets, and transitions, as shown in Figure 6-1. The steps are explained in the subsequent paragraphs.

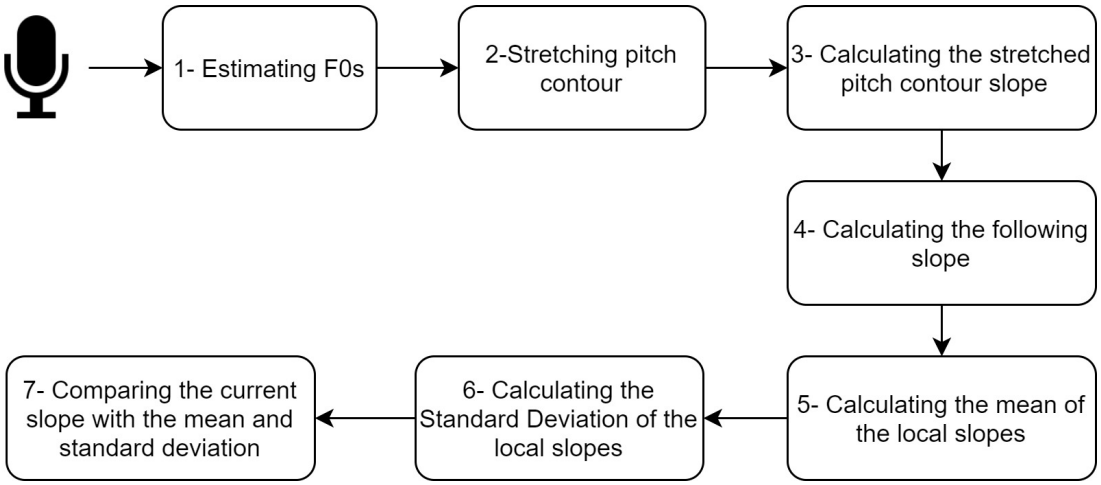


Figure 6-1 The main steps to find onsets in the proposed algorithm.

6.2.1 Estimating F0s

Since the algorithm is based on the fundamental frequencies, the F0s must be estimated correctly. However, as mentioned in the previous Chapters, the current real-time pitch detection algorithms are unreliable when applied to singing phrases. Therefore, according to Chapter 3, a more reliable offline algorithm, PYIN (Mauch and Dixon, 2014), was employed to avoid a compounding effect in this analysis if any real-time pitch detector algorithm would be used. Thus, it was possible to evaluate the accuracy of the onset algorithm without any adverse effects caused by the pitch detection algorithms. A Python library, Librosa (McFee *et al.*, 2020), was used for PYIN.

The main difference between the real-time and offline algorithms is the amount of data they need for the calculation. Therefore, real-time algorithms are only based on the previous data points and/or a few later data points, meaning that only a short buffer delay is required. On the other hand, offline algorithms require a long buffer delay to have sufficient data when

performing their calculations. Using the PYIN algorithm does not mean the proposed algorithm needs a long buffer delay to obtain a large amount of data, but the algorithm can work with a very short buffer delay, as explained below.

6.2.2 Stretching Pitch Contour

Since humans' vocal pitch range is wide, generally from 77 to 900 Hz (Heylen *et al.*, 2002), calculating significant changes occurring on pitch contour has some difficulty. For example, the slope of the line when moving from note $E2 \approx 82$ Hz to the note $F2 \approx 87$ Hz is much less than when it moves from note $E5 \approx 659$ Hz to note $F5 \approx 698$ Hz. Therefore, to counteract any adverse effect of this wide pitch frequency range on the slopes, the F0s are *stretched* to be on the almost same pitch frequency range.

Figure 6-2 plots two estimated pitch contours (panels a and b) and the stretched version of them (panels c and d, respectively). As depicted in Figure 6-2, although (a) and (b) are in different pitch frequency ranges, after stretching, the slopes between notes in both (c) and (d) are almost similar.

The following formulas, Equations (6-2) and (6-3), are used to implement the stretch.

$$max = \begin{cases} F0_i, & F0_i > max \\ max, & otherwise \end{cases}, \tag{6-2}$$

$$F0_i = \frac{F0_i * Threshold}{max}, \tag{6-3}$$

where the variable max holds the maximum F0 estimated until index $i-1$, and the constant value $Threshold$ holds the maximum possible F0. Since the maximum pitch frequencies of the singers in both datasets mentioned above are less than 1000 Hz, for this study, 1000 Hz is considered as the $Threshold$. In Equation (6-2), if the current F0, $F0_i$, is more than the max variable, Equation (6-3) should be run for all the F0s from index 0 to index $i-1$.

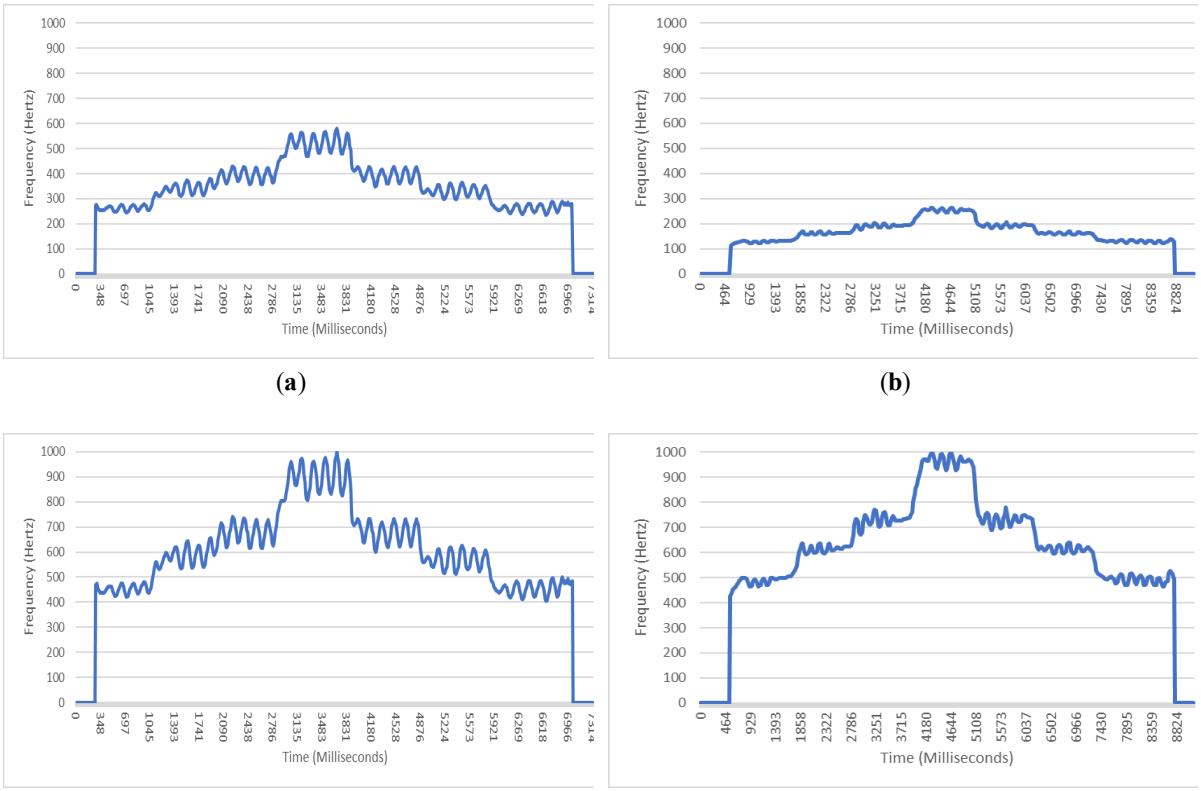
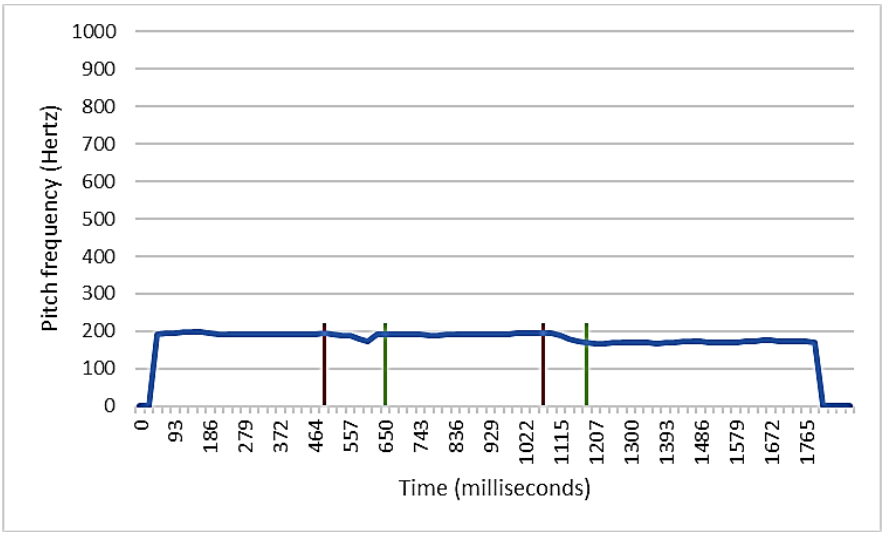


*Figure 6-2 The effect of stretching on pitch contour's slopes. (**c**, **d**) are the stretched pitch contours of (**a**, **b**), respectively.*

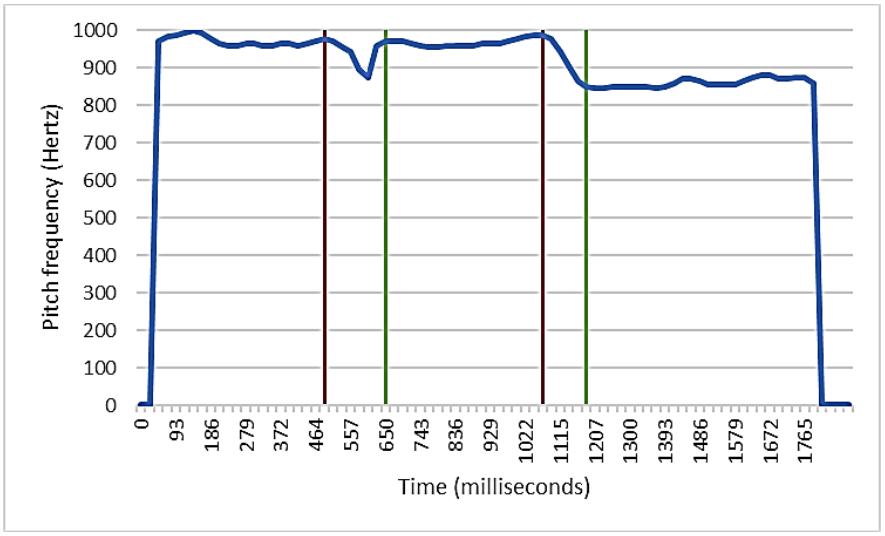
6.2.3 Calculating the Stretched Pitch Contour Slopes

To find the significant changes in F0s, the slopes between points in the pitch contour are needed. Figure 6-3 illustrates the process of calculating the slopes: in the top panel, (a), the estimated pitch contour is plotted; the graph in the middle panel, (b), shows a stretched pitch contour of the contour in panel (a) as discussed in Section 6.2.2, while that the bottom panel, (c), depicts the slopes between the F0s of the stretched pitch contour. It is computed by differentiating the contour. The vertical red lines in Figure 6-3 show the possible offset points, and the vertical green lines are the possible onset points.

(a)



(b)



(c)

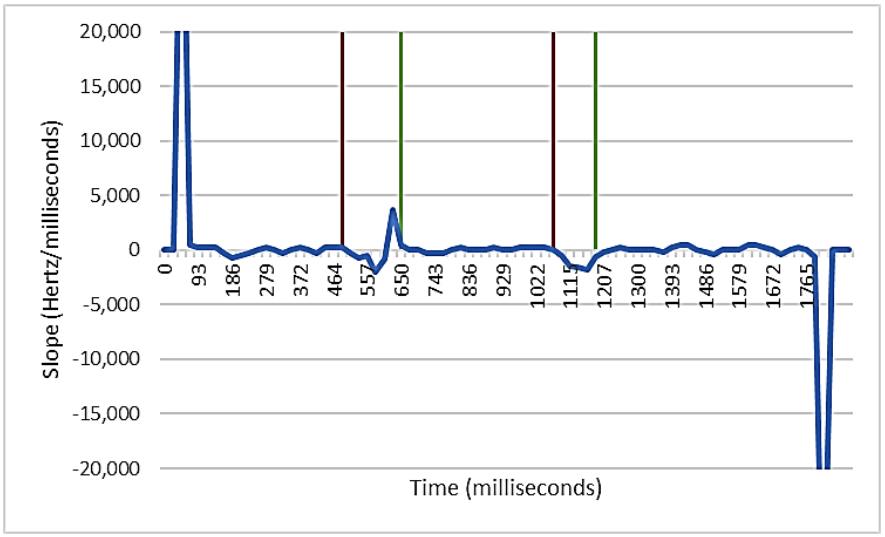


*Figure 6-3 Analyzing the pitch contour. (**a**) The original pitch contour of three notes, the first two notes are the same, and the third one is lower than the previous notes, (**b**) the stretched estimated values for the fundamental frequencies in (a), and (**c**) the slope of the pitch contour computed using differentiation. The red lines show the possible points for offsets, and the green lines are possible onsets.*

6.2.4 Calculating the Summation of Slopes in the following Line

Transitions can be observed in singing as the singer moves from one note to another. An example of this is outlined between the two pairs of orange-coloured lines in Figure 6-4.

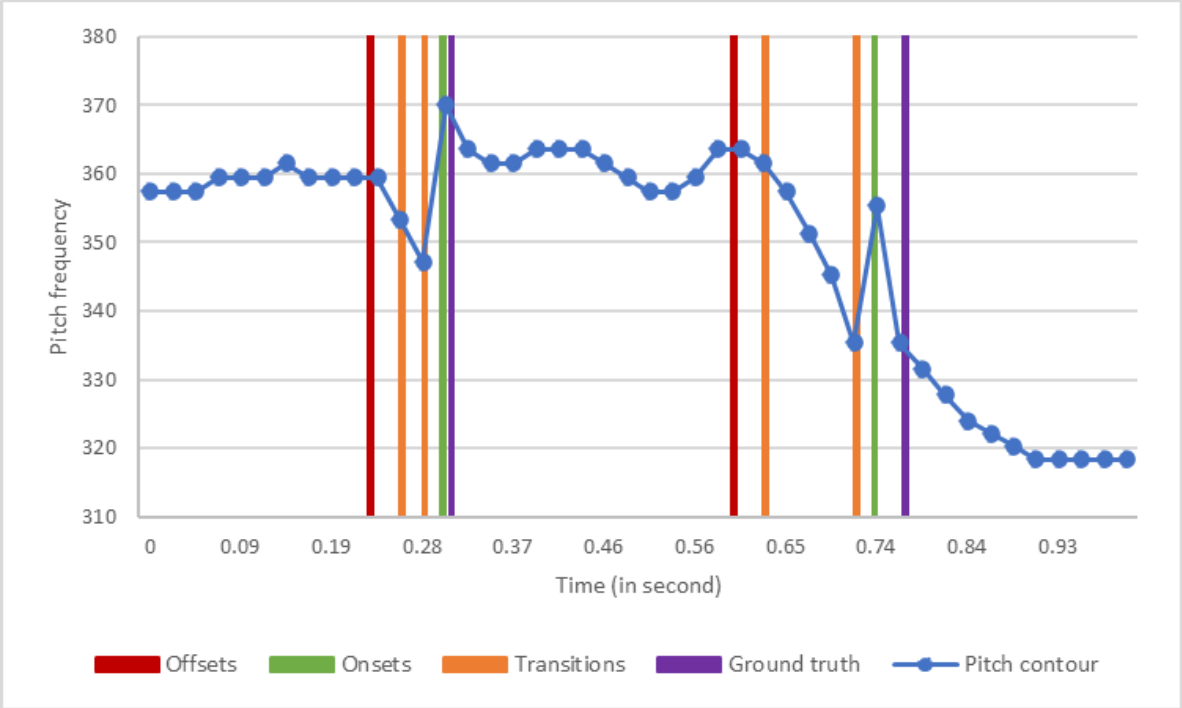


Figure 6-4 Points' statuses on a pitch contour. There are three notes: F4, F4, and E4, in order, sung by a professional female singer. The average pitch frequencies of the notes are 359, 362, and 323 Hertz, respectively.

In this step, the summation of the following points' slopes is calculated to find the transitions at each point. In other words, as far as the direction of the line (upward, downward, or straight) in the stretched pitch contour remains the same, the slopes between every two consecutive points would be added to each other. The algorithm is depicted in Figure 6-5, where i is the current point in this figure.

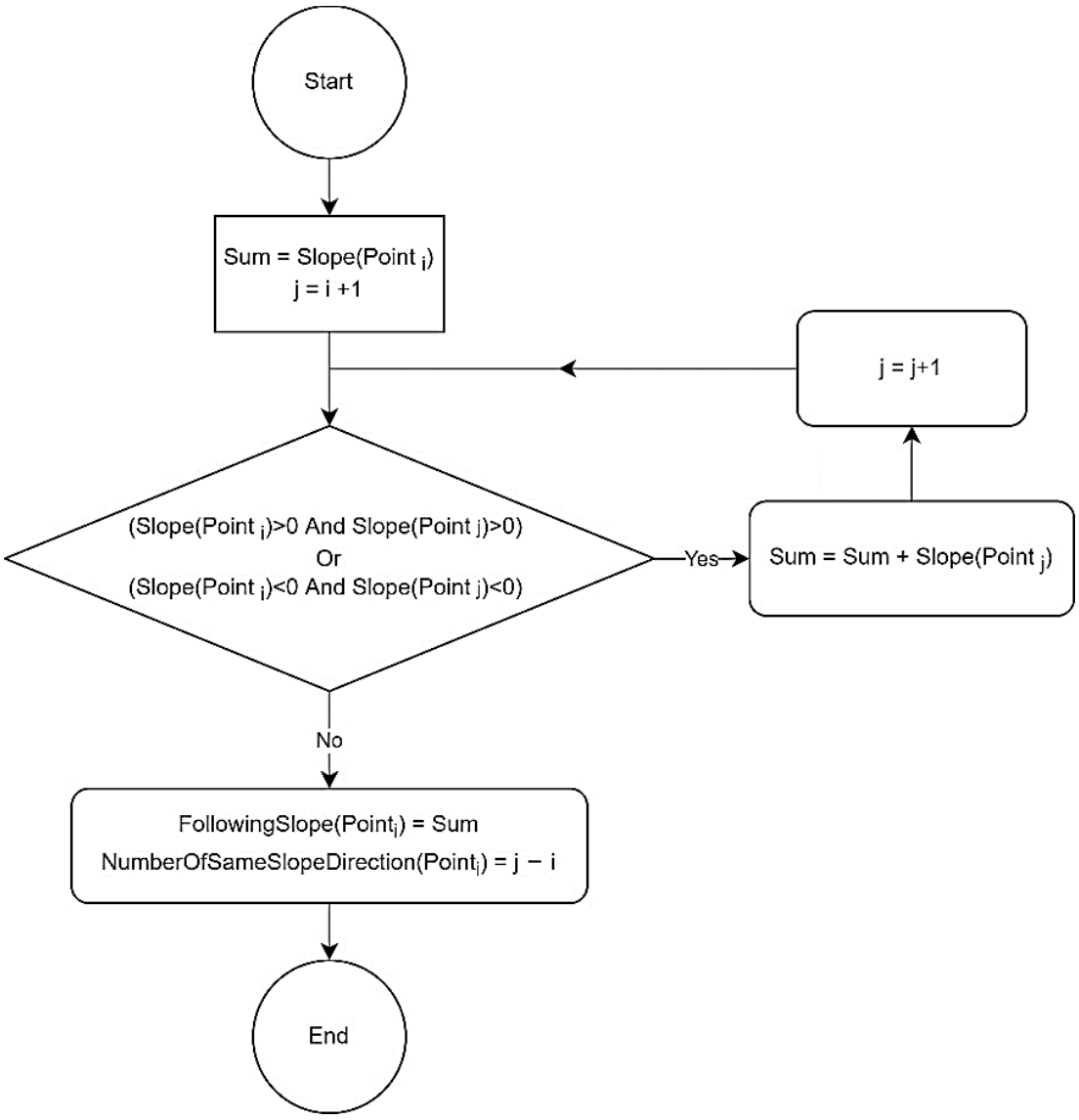


Figure 6-5 Calculating the summation of the following slopes of the differentiated contour.

The algorithm commences by computing the cumulative sum of the consecutive points in the slope representation. In other words, their amplitudes, the values on the *y*-axis in Figure 6-3(c), are summed. According to the evaluation of several manually annotated onsets, offsets, and transitions, it is observed that there is a sharp upward or downward movement between two consecutive notes in a pitch contour. Therefore, a heuristic function implemented using decision logic is applied to assess how much change happens after each new point. In addition, it is found how many consecutive points have the same sign as the current point's slope: that is, how many of the successive values are heading in the same direction. The function that denotes this in Figure 6-5 is named *NumberOfSameSlopeDirection(Point$_i$)*.

Therefore, the algorithm, at this point, detects when the slope changes sign.

6.2.5 Calculating the Mean of the Local Slopes

In this step, the mean of the local slopes needs to be calculated. This mean is always accounted for by considering some of the previous points until the current point, as shown in Equation (6-4).

$$Mean(Point_i) = \frac{\sum_{x=i-n}^{x=i} Slope\,(Point_x)}{n}, \tag{6-4}$$

where n is the size of the window. The value of n is important to produce a mean that can show the mean of the fluctuations in a note. If n is too big, it may include some old-time fluctuations that will result in an incorrect local mean. In contrast, if n is too small, there would not be enough fluctuations to calculate the correct local mean. The n should be selected based on the singing technique, notes' duration, and intervals. In this study, the selected values of n were chosen to be 230 ms for the Erkomaishvili dataset and 46 ms for the SVNote1 dataset. These selections for n were made according to a trial-and-error method of adjusting the n value to have the best result for one of the files of each dataset.

As shown in Figure 6-6, although the median duration of the notes in both datasets is almost similar, roughly 0.42 s, the duration of most of the notes in the Erkomaishvili dataset is longer than the median. In contrast, the duration of the notes in the SVNote1 dataset is distributed approximately uniformly below and above the average. Therefore, the variance of notes' duration in the Erkomaishvili dataset is greater than in the SVNote1 dataset. In addition, the variance of the intervals between notes in the Erkomaishvili dataset is smaller than in the SVNote1 dataset. Thus, two different n values for each dataset were selected.

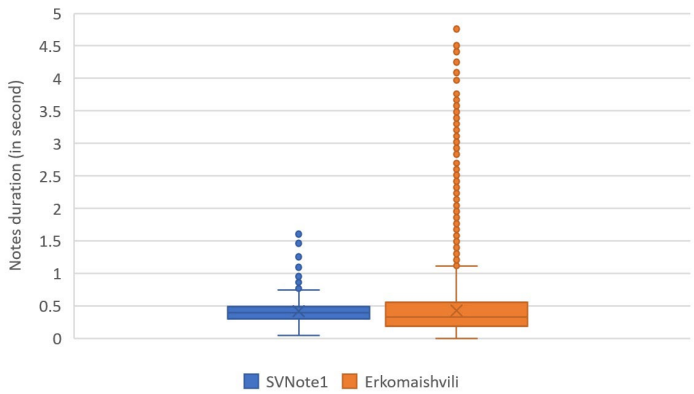


Figure 6-6 Box and whisker of the estimated notes' duration in the SVNote1 and the Erkomaishvili datasets.

6.2.6 Calculating the Standard Deviation of the Local Slopes

To define a significant trajectory change in the fundamental frequencies, the sample standard deviation of the local slopes is calculated as shown in Equation (6-5).

$$STD(Point_i) = \sqrt{\frac{\sum_{x=i-n}^{x=i}(Slope(Point_x) - Mean(Point_i))^2}{n-1}}, \quad (6\text{-}5)$$

The same window size (n value) as for calculating the mean was used for estimating the standard deviation.

6.2.7 Comparing the Current Slope with the Mean and Standard Deviation

In this step, all the required information is prepared to determine if a significant change has occurred in the fundamental frequency trajectory.

Each of the points in the pitch contour can have only one of the following statuses:

a) Onset: this means the point is an onset.
b) Offset: this means the point is an offset.
c) StartTransition: this means a transition will follow, and this point is the start of the transition.
d) EndTransition: this means it is the end of the transition.
e) None: this means this point is neither an event's start nor the end.

These statuses are illustrated in the diagram in Figure 6-4. The red and green lines show offset and onset events, respectively, while the orange lines denote a transition from a note to the following note, i.e., the points between an offset and its subsequent onset.

Figure 6-7 illustrates the algorithm for finding each point's status. This algorithm works based on the values calculated by the algorithm illustrated in Figure 6-5. This algorithm is run iteratively on each of the estimated pitch values.

First, a $Threshold$ for the local pitch contour's slope must be calculated. This is completed by adding the mean of the local slopes at $Point_i$ to the product of the standard deviation of the local slopes at P$oint_i$ and t coefficients. The t is a user-specified value that indicates which range of frequencies, based on their variation from the mean, should be considered as belonging to the same note. The value t does not define a fixed variation from the mean but is derived based on the singer's techniques. For instance, when the singer uses

vibrato, the variation is higher than singing in an unmodulated tone. Thus, since the Erkomaishvili dataset has more variations in its tones than the SVNote1 dataset, thresholds of 5 and 2 were selected, respectively, by employing a trial-and-error method.

Second, if the slope at $Point_i$ is bigger than the $Threshold$, it means that a trajectory change has happened. This significant change should be an $Onset$, $Offset$, or $StartTransition$. If it is the first trajectory change after a silence (see Branch B in Figure 6-7), it is a movement to reach an $Onset$; otherwise (see Branch A in Figure 6-7), the current point is an $Offset$. Based on each of these situations, Onset, Offset, StartTransition, and EndTransition statuses will be marked. The start and end of transitions are consecutively after and before an Offset and an Onset, respectively. In other words, the start and end of transitions are one point apart from the Offset and Onset points.

When the algorithm finds a significant trajectory change at $point_i$, all the events between $Point_i$ and $Point_{i+j}$ will be labelled; thus, the following point that needs to be checked is $i + j + 1$. Therefore, there is a jump with a size of $j + 1$ at the end of the algorithm to set the i value for the next iteration.

In the beginning, the $FirstTime$ variable is set to $True$, and also when a rest is reached (when $F0_i$ equals zero), a $True$ value will be assigned to this variable.

A full implementation of the algorithm has been released to provide all the details in a GitHub repository[1].

[1] https://github.com/BehnamFaghihMusicTech/Onset-Detection, accessed on 15 July 2022

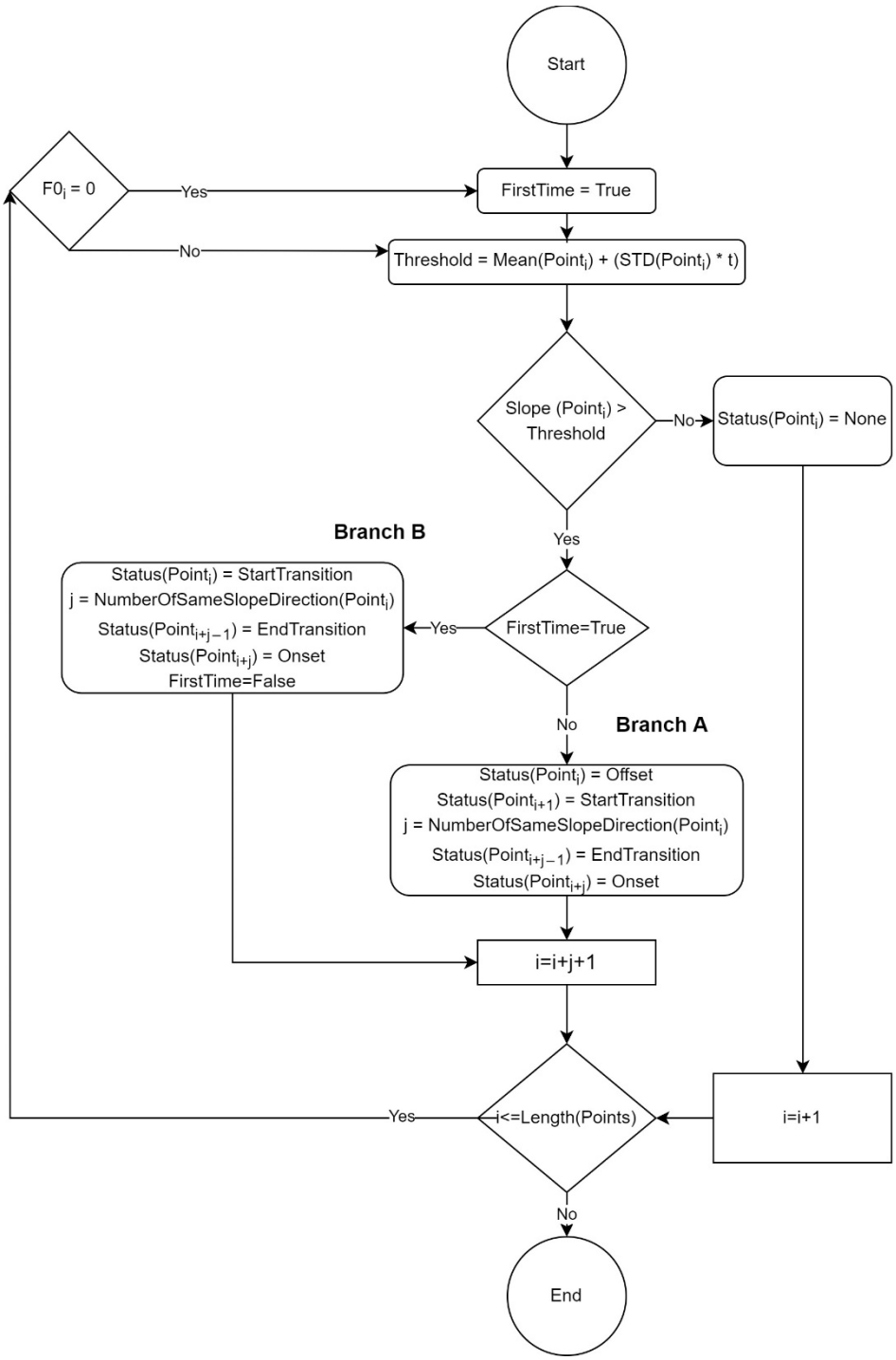


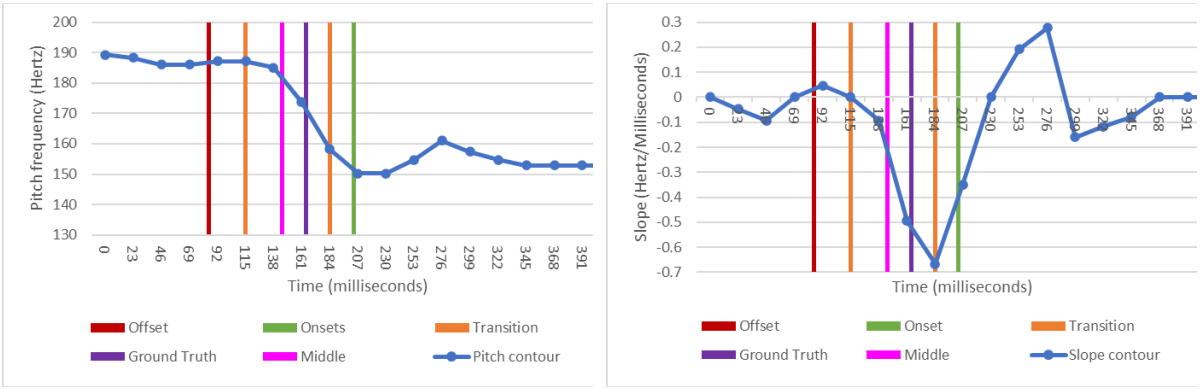
Figure 6-7 The algorithm for finding a significant change to find onset, offset, and transition.

6.3 Results and Discussion

This section provides the results and the details of the procedure for evaluating the proposed algorithm. It should be mentioned that the accuracy of the real-time proposed algorithm is compared against a set of real-time and offline algorithms. The delay buffer of the proposed algorithms depends on the window size to calculate the mean of the local slopes,

as mentioned in Section 6.2.5. Delays buffers of 230 and 46 ms are used for the Erkomaishvili and SVNote1 datasets, respectively.

Since the other onset detection algorithms mentioned in Section 6.1.2 only estimate onsets but not offsets and transitions, only onsets need to be extracted to evaluate and compare the proposed algorithm with them. Therefore, two types of onset times were considered: (1) First, only those points in the pitch contour are labelled as an onset. The green line illustrates these in Figure 6-8, and (2) the middle point between the start time of the transition and onset, illustrated by the pink lines in Figure 6-8, is considered the new onset point. The reason for considering the second type is to align with the approach used for ground truth datasets because they do not consider that transitions can exist between notes. Therefore, they would probably select a point between the red and green lines in Figure 6-8 as the onset. Therefore, considering the middle point should result in just a minor deviation from the ground truths.



(a) **(b)**

*Figure 6-8 An example illustrates the position of the onset point in the Erkomaishvili dataset (ground truth) compared to the onset, offset, and transition points indicated by the proposed algorithm. Panel (a) shows the pitch frequencies, and panel (b) depicts the slope contour according to panel (**a**).*

Generally, as shown in Figure 6-8, a range of plotted points between the offset and the start of the following note could be selected as an onset. Therefore, the algorithms were compared with different window sizes of 10, 50, 100, 150, 200, and 250 ms for calculating the F-measure. Table 6-1 and Table 6-2 display F-measures computed across all the algorithms in the six window sizes. A larger window size for F-measure shows more similarity since an enormous difference between the ground truth and the estimated onset would be acceptable in this case. However, as seen in Table 6-1 and Table 6-2, after applying the window size of 150 ms, the speed of improvement in F-measure values decreases. In addition, a window size

of more than 250 ms cannot be meaningful since it accepts more than a 250 ms difference between the ground truth and the estimated onset, which is too long. These tables provide the similarity between the ground truth's onsets times and the estimated onsets times by each algorithm. As mentioned above, two onset point selections are considered regarding the proposed algorithm. The rows titled "Pro Algorithm 1" in Table 6-1 and Table 6-2 consider the green line in Figure 6-8 as the onset, while the rows titled "Pro Algorithm 2" select the middle point, which is the pink line in Figure 6-8.

All the algorithms show better results on the SVNote1 dataset than on the Erkomaishvili dataset. One of the possible reasons for the better result could be the better audio quality of the SVNote1 dataset. In addition, there is a speaking introduction at the beginning of each audio file that is not included in their annotations. Nevertheless, since all the algorithms are working on the same audio files, they all have the same faulty sound, which will not affect the comparison.

As the result of the comparison, our proposed algorithm finds more correct onsets compared to the other algorithms when the window size is equal to or greater than 150 ms, as shown in the rows for Pro Algorithm 1 in Table 6-1 and Table 6-2. The bold numbers in these two tables highlight the performance of the best algorithm.

Selecting the average of the onset and the start of the transition as the onset leads to an increase in the accuracy of the proposed algorithm by 3.4% on average for the Erkomaishvili dataset. However, the opposite is the case for the SVNote1 dataset, in which the accuracy of the onset identification decreased by 3.8%. The reason for these opposing results is that the annotator of the Erkomaishvili dataset considered onsets to lie more closely to the middle, as depicted in Figure 6-8.

However, the SVNote1 dataset's annotators mostly considered onsets after the proposed algorithm's onset point, as shown in Figure 6-4. Both approaches can be interpreted as correct since the onset point is not universally agreed in a pitch contour, as mentioned above, but it is deemed to be valid over a range of points.

To check the meaningfulness of the averages of the F-measure values of each onset detection algorithm, the *p*-values for ANOVA were calculated for all the F-measure values calculated for every single file. The ANOVA's *p*-values for both Table 6-1 and Table 6-2 were

less than 0.0001, which means a significant difference exists between the accuracy of all evaluated algorithms.

Table 6-1 The average of the F-measures of all the algorithms on the Erkomaishvili dataset based on six window sizes, from 10 to 250 ms.

Algorithm \ Window size	10	50	100	150	200	250
Aubio *	0.072	0.295	0.415	0.480	0.523	0.553
EssCplx	0.076	0.304	0.444	0.508	0.541	0.557
EssHFC	0.065	0.297	0.452	0.533	0.58	0.611
LibBt	0.064	0.288	0.448	0.521	0.560	0.585
LibBtRMS	0.046	0.247	0.416	0.502	0.551	0.58
LibRaw	0.056	0.295	0.455	0.525	0.563	0.586
MadmomCNN *	0.086	**0.308**	0.42	0.479	0.516	0.543
MadmomSF *	**0.088**	0.287	0.392	0.450	0.488	0.515
Pro Algorithm 1 *	0.036	0.198	0.416	0.55	0.631	0.681
Pro Algorithm 2 *	0.059	0.274	**0.464**	**0.579**	**0.649**	**0.691**

* The algorithms marked with a star are real-time algorithms.

Table 6-2 The average of the F-measures of all the algorithms on the SVNote1 dataset based on six window sizes, from 10 to 250 ms.

Algorithm \ Window size	10	50	100	150	200	250
Aubio *	0.118	0.509	0.655	0.694	0.696	0.696
EssCplx	0.064	0.313	0.492	0.550	0.562	0.563
EssHFC	0.095	0.561	0.739	0.787	0.798	0.798
LibBt	0.045	0.371	0.611	0.737	0.779	0.786
LibBtRMS	0	0.111	0.498	0.697	0.761	0.783
LibRaw	**0.257**	**0.672**	0.763	0.784	0.785	0.785
MadmomCNN *	0.042	0.496	0.665	0.667	0.667	0.667
MadmomSF *	0.020	0.662	**0.779**	0.781	0.781	0.782
Pro Algorithm 1 *	0.089	0.469	0.704	**0.827**	**0.893**	**0.923**
Pro Algorithm 2 *	0.108	0.432	0.646	0.764	0.845	0.881

* The algorithms marked with a star are real-time algorithms.

As another result, the average and the standard deviation of the duration of the transitions are shown in Table 6-3. This table also provides the minimum (average minus standard deviation) and the maximum (average plus standard deviation) typical duration for the transitions. Therefore, the average transitions' duration in the datasets is almost the same. Overall, based on the results, the minimum and maximum duration of the transitions were approximately 16 and 98 ms, respectively. Therefore, since the proposed algorithm is based on the trajectory changes in a pitch contour and the transitions show these significant

changes, the minimum buffer delay required to find the onset, offset, and transition is 16 ms and the maximum of 98 ms. However, most events should be found correctly, with the average transition duration being around 57 ms. This delay would be acceptable for most real-time music information retrieval applications. For example, Henkel and Widner's real-time score-following system (Henkel and Widmer, 2021) requires a delay of around 56 ms. Generally, by examining more datasets, these numbers can be generalized and make them fixed for the algorithm instead of adjusting them according to the features of the input signals.

Table 6-3 The average, standard deviation, minimum, and maximum typical duration of transitions in both the datasets and overall.

	Average	STD	Min	Max
Erkomaishvili	57.44	40.77	16.67	98.21
SVNote1	56.25	44.68	11.57	100.93
Overall	57.4	40.91	16.49	98.31

Since the proposed algorithm is based on the changes in a pitch contour, when the intervals between notes are bigger and there are fewer soft onsets, the algorithm can estimate onsets more accurately.

6.4 Conclusions

This chapter has proposed a new algorithm for detecting onsets, offsets, and transitions between notes in singing. The algorithm can work in both offline and real-time environments. In the case of real-time, a 57-millisecond delay is needed to have adequate information for calculating the events. The proposed algorithm showed an improvement in accuracy when compared with eight well-known algorithms in two different datasets. When the window size for calculating the F-measure was considered between 100 ms and 250 ms, the proposed algorithm calculated the onsets better than other algorithms in the Erkomaishvili dataset. However, its accuracy was not the best for the window sizes 50 ms and 10 ms compared to the other algorithms. The comparison on the SVNote1 dataset was almost similar, but when the algorithm was the best one when the window size was 150 ms or more.

The onset detection algorithm introduced in this chapter will be employed in the following chapter to generate an annotated monophonic singing dataset.

Chapter 7
Generating an annotated singing dataset

This chapter introduces a new annotated singing dataset. Not only an explanation of the files in the dataset is provided, but also a review on the available dataset and the methodology of annotating audio files are discussed. This chapter's contents are according to the following journal paper.

- *Faghih, Behnam & Timoney, Joseph, "Annotated VocalSet: A Singing Voice Dataset". Applied Sciences. 12(18):9257. https://doi.org/10.3390/app12189257*

This chapter's goal is to generate a singing annotated dataset to provide information to be able to analyse the behaviour of trained-professional singers in their performances. Thus, the following section explains the approaches used to calculate the annotations, and a description of the Annotated-VocalSet is provided in section 7.2. After that, Section 7.4 introduces and compares four methods of selecting the onset, offset, and transition positions. Finally, this chapter will be closed with a conclusion in Section 7.5.

7.1 Steps to Generate the Dataset

Generally, four steps were followed to add the annotations: 1- estimation of F0, 2- detection of onsets, offsets, and transitions, 3- extraction of notes' features, and 4- adding the scores to the extracted notes. These steps are explained in the following.

7.1.1 Estimating Fundamental Frequencies

A state-of-the-art pitch detector algorithm, PYIN (Mauch and Dixon, 2014), was employed to estimate the fundamental frequencies of each file. The implementation of the PYIN in Librosa (McFee *et al.*, 2022) was used as it is one of the well-known Python libraries. According to Chapter 3, the PYIN algorithm is a reliable pitch estimator for singing signals. However, it still returns incorrect estimates for some F0s. Therefore, the Smart-Median pitch smoother algorithm, introduced in Chapter 5, was employed to smooth the pitch contours estimated by PYIN.

However, after plotting all the pitch contours, the authors reviewed all pitch trajectories and realized that some of the pitch contours were incorrect. We looked at the plotted data to find the incorrectly generated pitch contour. Where the plotted shape was not as expected, more investigations were done to find the reasons. There were two main reasons for incorrectly shaped pitch contours:

1- Wrongly estimated F0; for example, octave-doubling, as discussed in Chapter 3 and Chapter 4.
2- Singers' mistakes; for example, in some cases, such as in the files f7_scales_c_fast_forte_i, m6_scales_c_fast_piano_a, and

m11_scales_f_fast_forte_e.csv, the singers sang one note less than given in the musical scores, or they sang some extra notes, such as in the file m1_caro_straight.

Therefore, we removed these incorrect files from the Annotated-VocalSet to have a reliable set of pitch contours. In total, 24.5 per cent of files were discarded from the original dataset.

7.1.2 Detecting Onsets, Offsets, and Transitions

After preparing the pitch contours, a semi-automatic approach was used to annotate pitch contours with onset, offset, and transitions. First, the algorithm introduced in the previous chapter was used to estimate the onset, offset, and transition between two consecutive notes.

After labelling the events, all the detected events were graphed to double-check the accuracy of the annotations. Some errors observed among the estimated events were then corrected with a software tool developed by this thesis's author; Figure 7-1 depicts the tool. The software uses different colours for each event. If the user finds an incorrect event, they should edit the CSV file containing the values that will be explained in Section 7.2.1. Therefore, its users can change the CSV files and immediately see the results on the screen to ensure the events are labelled correctly. This tool was developed using the language C# and is available at https://doi.org/10.5281/zenodo.7061507, accessed on 14 September 2022. It should be mentioned that this tool works on CSV files in the "raw" folders.

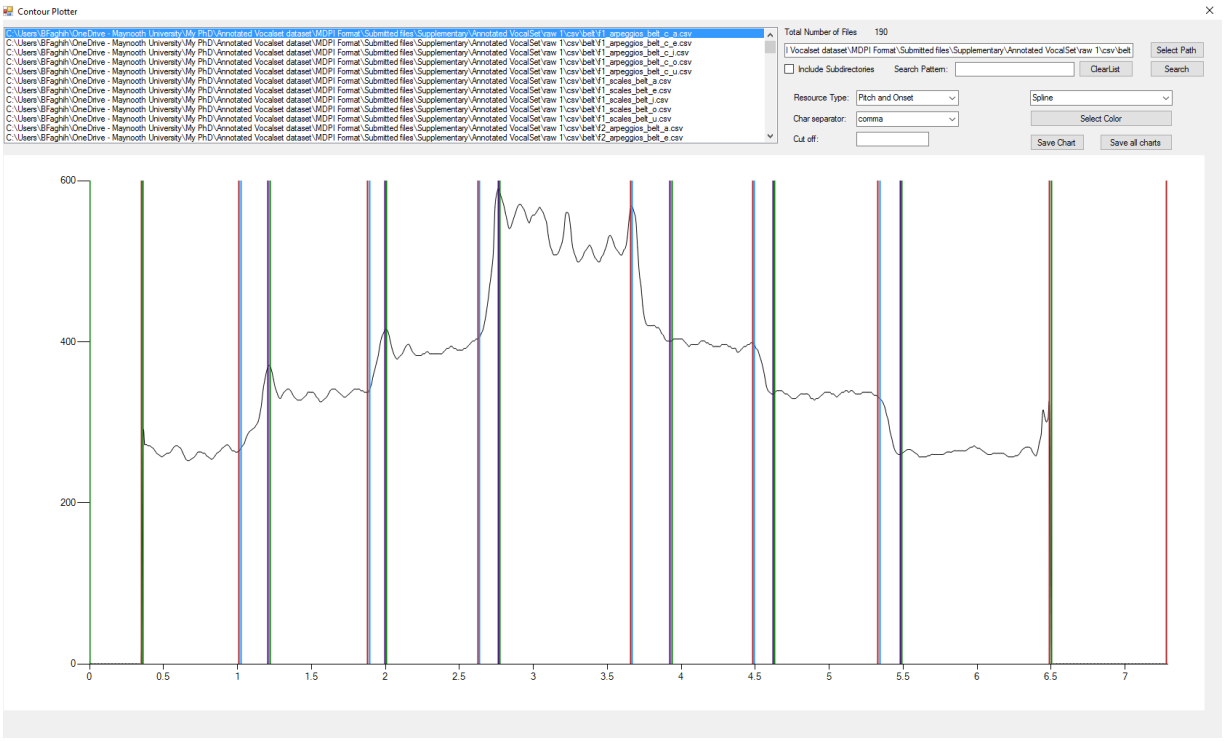


Figure 7-1. The software tool used to check and correct F0, onset, offset, and transition annotations by indicating them with different colours.

To adjust the estimated annotations, firstly, files were divided among five non-expert musicians (but they were trained on how to interpret a pitch contour to find events) to correct the estimated events. Finally, an expert musician with over 12 years of music training reviewed all the events and adjusted any incorrect ones.

Figure 7-2(a) depicts the onset, offset, and transition from a part of a pitch contour. The red line shows the offset, the green line is the onset, the blue line shows the start of the transition, and the purple line shows the end of the transition.

Since several software libraries and associated studies do not include transitions but only onsets and/or offsets, as discussed in Chapter 6, we have proposed two approaches, considering and not considering the existence of a transition, for annotating onsets, offsets, and transitions.

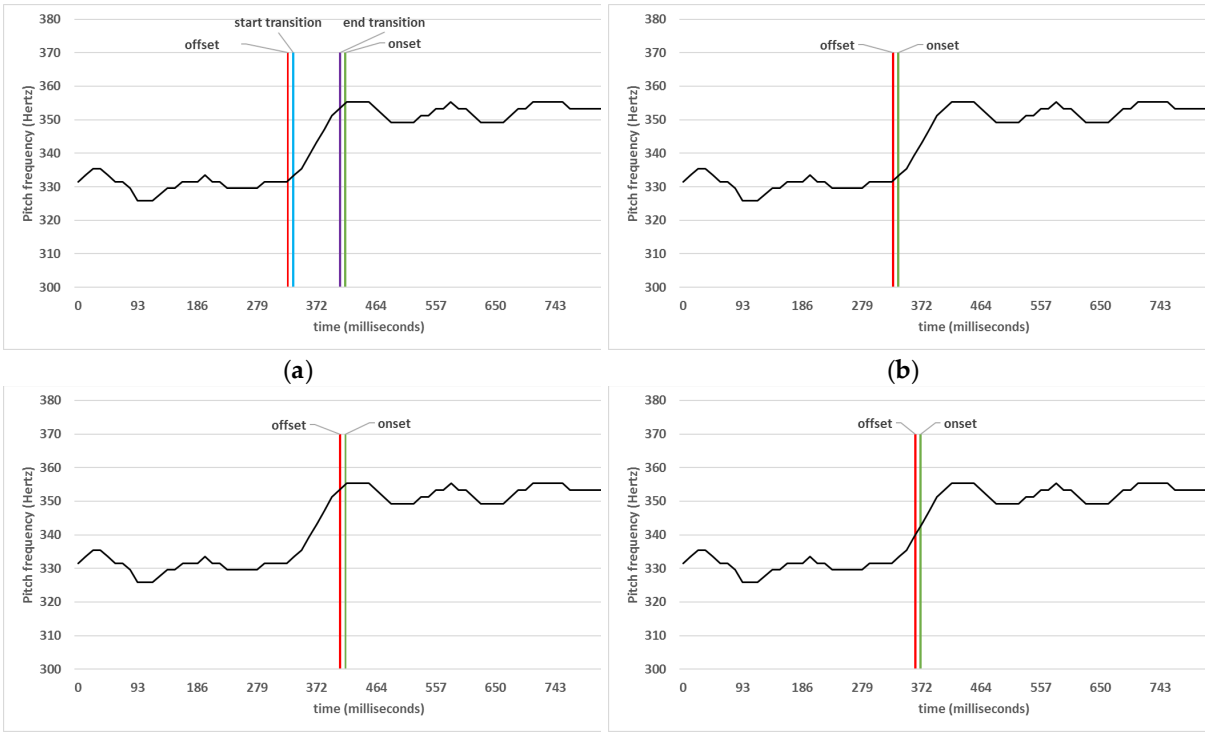


*Figure 7-2. Points' statuses on a pitch contour. Two notes, E4 and F4, are sung by a professional female singer. **(a)** Showing offset, the start of a transition, the end of a transition, and onset events in order. **(b)** The transition was not considered, and the onset started immediately after the offset point according to **(a)**. **(c)** Similarly, the transition was not considered, but the offset was annotated to lie immediately before the onset point in **(a)**. **(d)** Likewise, the transition was not considered, but the middle points between the onset and offset points in **(a)** are annotated as offset and onset.*

7.1.3 Extracting Notes Features

After annotating the pitch contours with the onset, offset, and transitions, as explained above, the following formulas were used to calculate each note's features.

1) *Start time*: is the onset time.
2) *End time*: the time of the offset.
3) *Duration*: calculated by subtracting the "*Start time*" from the "*End time*".
4) *Type*: can be a rest, sound, or transition according to Equation (7-1).

$$Type = \begin{cases} Rest & if\ the\ estimated\ pitches\ between\ start\ and\ end\ times\ are\ 0 \\ Sound & if\ the\ estimated\ pitches\ between\ start\ and\ end\ times\ are\ not\ 0 \\ Transition & if\ the\ start\ and\ end\ times\ are\ related\ to\ a\ transition \end{cases} \quad (7\text{-}1)$$

5) *Average F0*: contains the average of the F0s of the note.
6) *Median F0*: includes the median of the estimated F0s for the current note.
7) *Min F0*: contains the minimum estimated F0 in the current note.

8) *Max F0*: shows the maximum estimated F0 in the current note.
9) *The standard deviation of F0s (STD)*: this column shows the standard deviation of the estimated F0s of the current note.
10) *Average F0s in the range of STD (AverageStd)*: This column includes the average of the only estimated F0s that satisfy the following condition, Equation (7-2):

$$(Average\ F0) - (2 * STD) \leq F0_i \leq (Average\ F0) + (2 * STD) \quad (7\text{-}2)$$

This metric calculates the average by omitting the pitches that were determined to be outliers. Since, in some cases, especially when the note's duration was very short, none of the estimated pitches were within one standard deviation distance, and thus two standard deviation distances were considered instead.

11) *Estimated MIDI code*: includes the MIDI codes associated with the calculated AverageStd, as shown in Equation (7-3).

$$12 * log_2((AverageStd)/\ 440) + 69 \quad (7\text{-}3)$$

12) *Repetition No*: it is common that in a piece of music, some of the notes appear several times. This annotation indicates the repetition of the note in the song.

7.1.4 Combining Extracted Notes with Ground Truth Scores

After extracting the notes, they needed to be associated with the scores. The scores and lyrics of each file are available in the VocalSet dataset (Wilkins *et al.*, 2018). Therefore, the estimated notes and the scores were automatically aligned by a software program created by this thesis's author in C# programming language. For each musical score, an array that held the notes' information, such as name and duration, was created in the code. Then the two lists, estimated notes and scores, were aligned. To achieve this, the code iteratively walks through the lists, and when the notes are matched, that is, notes on each list were either a rest or an articulated sound, they were associated. Finally, the following columns were added to the files in the "extended" directories.

13) *Ground truth Note name*: their format is a capital letter + [#/b] + octave number—for example, C4, D#5, or Bb4.
14) *Ground Truth Frequency*: It includes the frequency of the note based on A4 = 440 Hz.
15) *Ground Truth MIDI code*: the MIDI note number according to the ground truth note name.

16) *Lyric*: if the file comes from an arpeggio, scale, or long-tone, this column includes the sung vowel. Otherwise, the syllable corresponding to the note according to the scripts provided in VocalSet [6] is used in this column.
17) *Ground Truth Note duration name*: it includes the name of the note's duration. Its value is one from this list: Whole_note, Half_note, Dotted_Quarter_note, Quarter_note, Dotted_Eighth_note, Eighth_note, Eighth_note_triplet, Sixteenth_note, or Grace_note.
18) *Ground Truth note duration*: the BPMs of two series of songs were indicated in VocalSet. The BPM of the songs marked as 'fast' is 330, and those marked 'slow' is 60. Therefore, the ground truth notes' duration of these two categories only were annotated in this study.
19) *Interval to the previous note*: this shows the number of semitones between the current note and the immediate previous note.
20) *Interval to the following note*: this shows the number of semitones between the current note and the note immediately following it.

7.1.5 Checking Annotation Correctness

After creating the annotations, several evaluations were used, as listed below, to check their correctness.

1) All the pitch contours and the events were plotted, similar to Figure 7-1, to manually double-check them. First, a non-expert but trained person checked and corrected the files. Then, another person with the same expertise double-checked them. Finally, an expert reviewed the annotations. As seen in Figure 7-1, it is possible to distinguish the played notes from a pitch contour. Therefore, when an event was not estimated correctly, the annotator could find and alter it.
2) A piece of software code was developed to check if the sequences of the onset, offset, and transition were correct. For example, an onset should be followed by an offset. In addition, the start and end of a transition should be between an offset and its consecutive onset. The list of the incorrect files was saved in a text file. Then, an expert corrected the erroneous files. These processes were repeated until the software code found no more errors.

3) In the process of combining the extracted notes with the scores as discussed in Section 7.1.4, if the number of extracted notes was not equal to the number of notes in the ground truth, the automatic tool listed the incorrect files to be investigated by the user. These steps were repeated until no error was reported by the tool.
4) Finally, with a piece of code, the information in all the files, including the header in the "extended 4" directory, were combined to have all the information in one CSV file. This file is available in the Annotated-VocalSet root directory and is named "all-files.csv". Then rows were sorted based on the column named Shifted_F0—Nominal_F0. The values in this column were expected to be between −1 and 1. Therefore, all the records that did not belong to this range were investigated manually to fix the inaccurate ones. Then, after this test, most of the notes were within the expected range, and the author became convinced that the out-of-range values were not errors due to the erroneous actions of the author but were due to errors in the notes produced by the singers.

After these checks, the correctness of the annotations satisfied the author.

7.2 Dataset Description

The directories' hierarchy of the Annotated-VocalSet is depicted in Figure 7-3. The notes in the files in the "raw 1" and "extended 1" directories were calculated by considering a transition between notes, as depicted in Figure 7-2(a). On the other hand, the notes in the other directories were estimated without considering a transition between notes. Therefore, for the estimated notes in the "raw 2" and "extended 2" directories, the onsets started immediately after the offsets, as shown in Figure 7-2(b). Moreover, the files belonging to the "raw 3" and "extended 3" directories were calculated by considering the points before onsets as offsets, as illustrated in Figure 7-2(c). In addition, the files in the "raw 4" and "extended 4" directories included the notes where their onset and offsets were estimated as the middle points between the offset and onset when a transition was considered between notes. As an illustration, the onset and offset points in Figure 7-2(d) are at the middle of the onset and offset points in Figure 7-2(a).

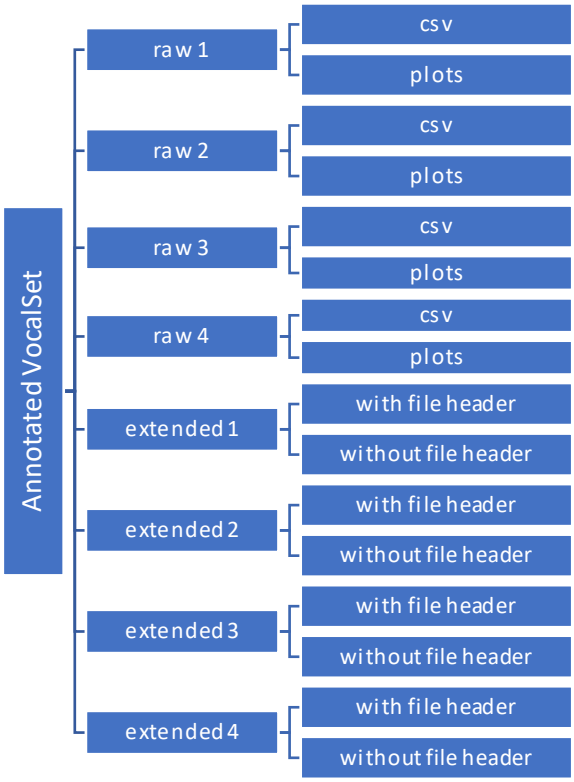


Figure 7-3. The directories' hierarchy of the Annotated-VocalSet.

7.2.1 Raw Directories

The "raw" directories shown in Figure 7-3 included each audio file's CSV and JPEG (plotted) files.

The CSV file columns in order are Time (in second), F0 (in Hertz), Amplitude (between 0 and 1), onset (true or empty), offset (true or empty), and Transition (true or empty). The Transition column indicates whether or not the detected onset/offset is related to a transition from one note to another note. In other words, if the transition column is true and the onset/offset is true, it means that this onset/offset shows the start/end of a transition.

The plot folders include the graphs of the pitch contours with the onset, offset, and transition events.

7.2.2 Extended Directories

The files in these directories are created based on the "raw" files. The extended directories include two subdirectories: with file header and without file header. The difference between these two subdirectories is that the folder named "with file header" contains files having a header, as described in the following.

The heading part is positioned at the top of the CSV files and has 15 lines, and the description of each line is as follows:

1) Filename;
2) Gender;
3) Singer name (f1, f2, etc.);
4) The technique (breathy, fast forte, fast piano, etc.);
5) Type of music (Scale, Arpeggios, etc.);
6) Vowel (a, e, I, o, u);
7) BPM;
8) File duration in milliseconds.

Lines 9 to 15 are reserved to allow the possibility of their use at some time in the future. In this case, the software developed for processing the current version of the dataset will not need to be changed to work with future versions.

After the heading (or at the top of the files in the without file header directories), the list of the columns in order are Sequence, Start time, End time, Duration, Type, Average F0, Median F0, Min F0, Max F0, Standard deviation F0, Average F0 in range of STD, Estimated MIDI code, Ground truth Note name, Ground Truth Frequency, Ground Truth MIDI code, Lyric, Ground Truth Note duration name, Ground Truth note duration, Interval to the previous note, and Interval to the following note. All the columns have been explained in Sections 7.2.1 and 7.2.2.

The original VocalSet dataset already provides the information in the heading section, but this study adds these details of the singing notes.

7.3 Summary of the generated data

This section provides statistical overviews of the generated annotations to make it clear for the reader to understand the dataset.

Figure 7-4 depicts the total number of annotated notes in each music type sung by men and women. The figure shows that the number of notes sung by males and females is almost the same. In addition, the notes mainly come from singing Scales and Arpeggio music type. On

the other hand, each of the musical excerpts (Caro, Dona, Row Row) has the lowest number of notes.

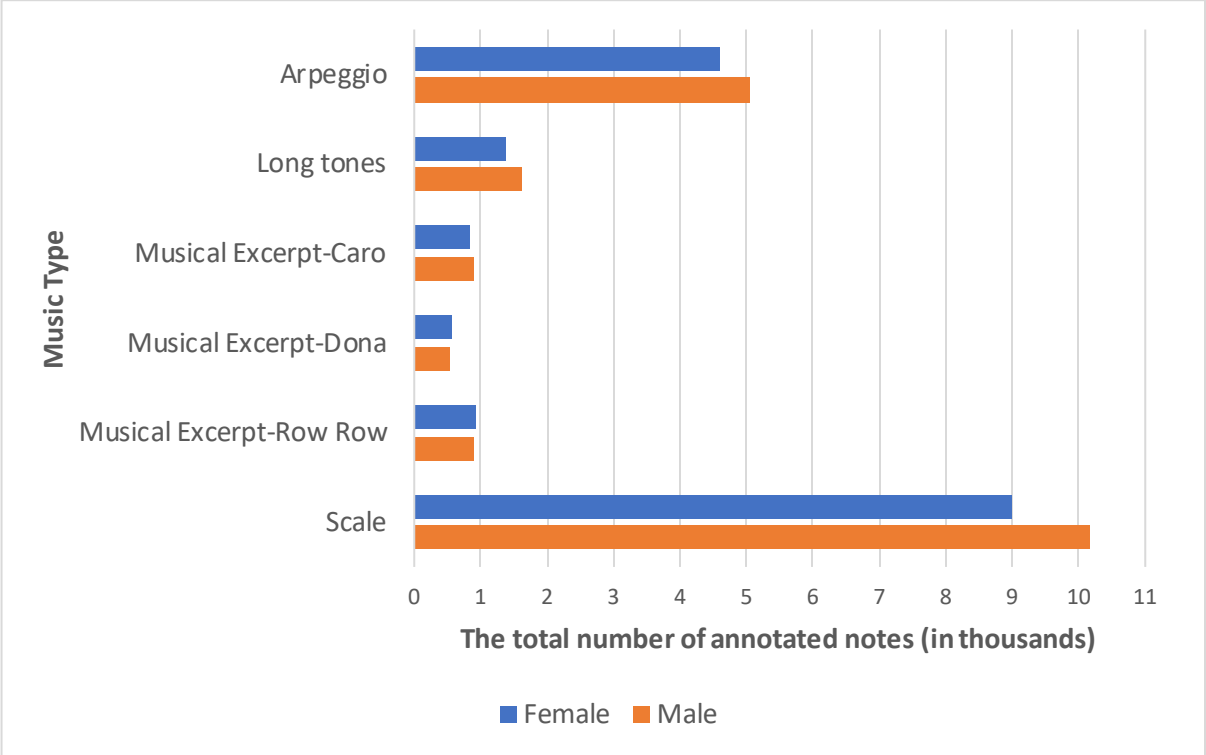


Figure 7-4 The total number of notes in each music type sung by males and females

Figure 7-5 provides a box and whisker plot of the pitch frequencies sung by the singers. As can be seen from the plot, the main distribution of the pitch frequencies is roughly between 50 Hz and 700 Hz, and the majority of notes are in the range of 200 Hz and 400 Hz.

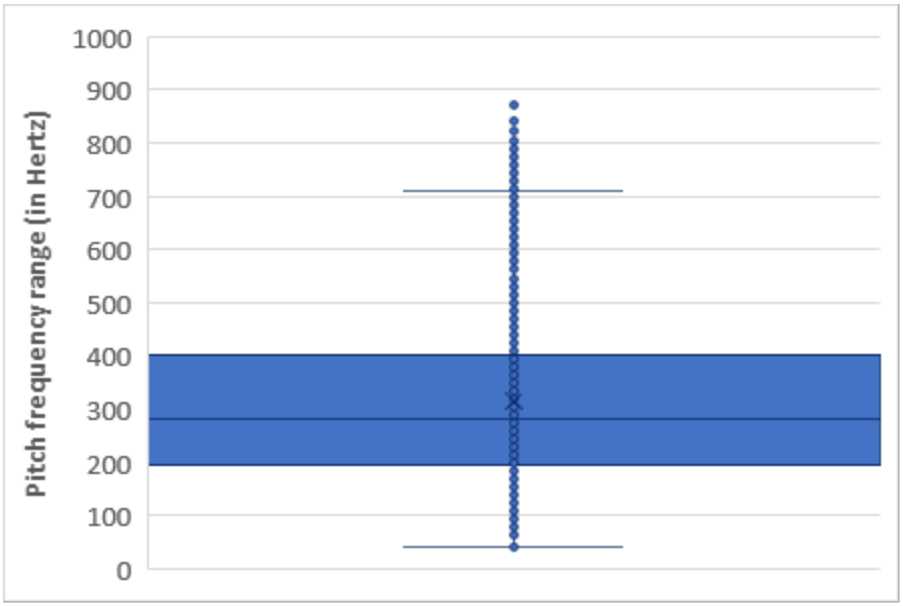


Figure 7-5 A box and whisker plot of the pitch frequencies sung by the singers.

Similarly, the range of the duration of the notes is plotted in Figure 7-6. As shown in the figure, the duration of most of the notes is less than two seconds, albeit some longer sounds also exist in the dataset.

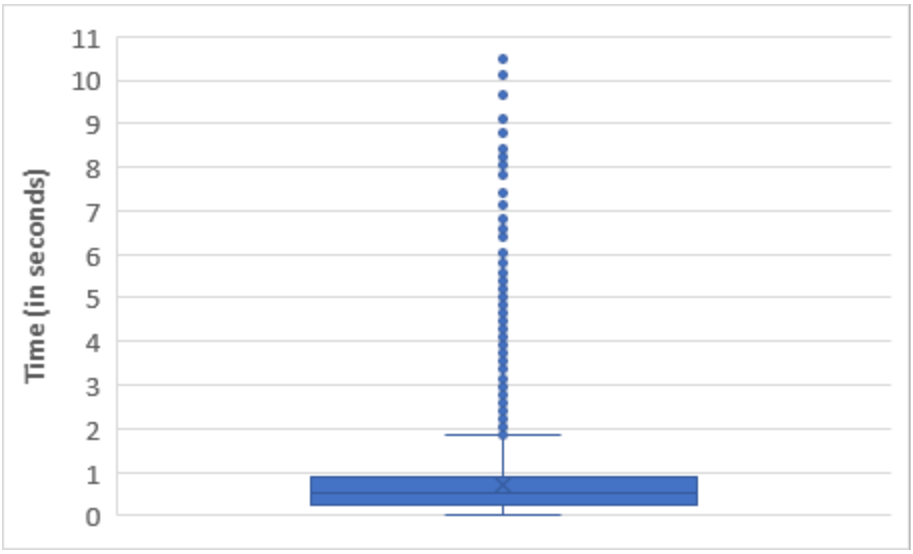
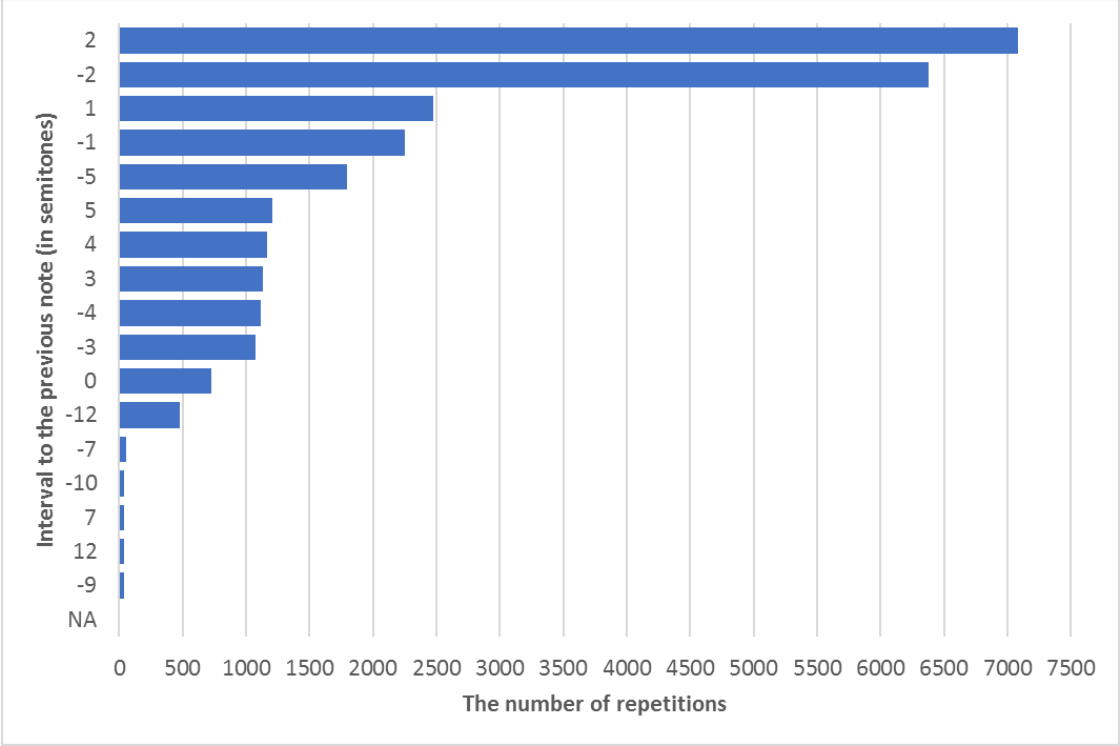


Figure 7-6 A box and whisker plot of the duration of the notes sung by the singers.

As depicted in Figure 7-7, the pitch intervals between notes in the database are limited to one octave, 12 semitones, with a significant majority in two semitones pitch intervals.

(a)



(b)

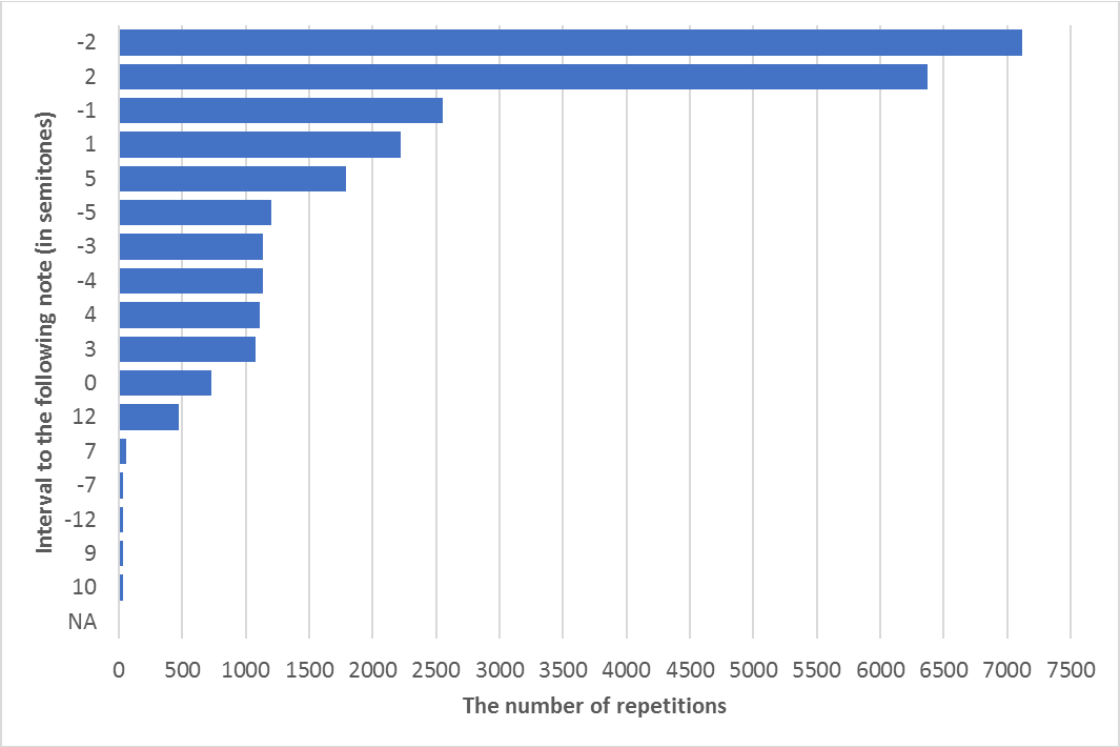


Figure 7-7 The total number of notes categorized by their pitch intervals to the previous note (a) and the following note (b).

Table 7-1 shows the percentage of notes' repetitions in a piece of music, as discussed in section 7.1.3. The table shows that most of the notes are in one or two repetitions.

Table 7-1 The percentages of the repetition of the notes in a piece of music out of total of 299117 notes

Repetition	Percentage of Repetition
1	52.78%
2	40.70%
3	2.30%
4	1.55%
5	1.43%
6	0.87%
7	0.37%

7.4 Comparing the Four Methods of Selecting the Positions of Onset, Offset, and Transition

Since this study provides four approaches for selecting the onset, offset, and transition discussed in Section 7.1, a comparison among them is provided to understand the differences

between them better. To compare them, the theoretical pitch frequencies of notes in the MIDI pitch code are considered to be the ground truth. In addition, each note's Average, AverageStd, and Median F0 values were converted to MIDI pitch codes. Therefore, each approach that can produce fewer differences from the ground truth is considered a better approach. Finally, all possible pair permutations of approaches were compared. Table 7-2 to Table 7-4 show the *p*-value for the *t*-test employed on each pair group to determine if the difference among the means of the distances between the estimated MIDI code and the theoretical MIDI code is meaningful. For example, the first row in Table 7-2 shows that there is no noticeable difference (*p*-value > 0.05) between the first approach (the files in the "extended 1" directory) and the second approach (the files in the "extended 2" directory).

As shown in Table 7-2 and Table 7-3, in some cases, there are statistically significant differences in the variation of the estimated pitch frequencies of notes when computed using the Average as compared to the AverageStd. However, the Median approach does not show a significant difference, as illustrated in Table 7-4. Nevertheless, according to Table 7-5 to Table 7-7, since the maximum difference between the average difference of F0s calculated by each approach is less than 0.2 MIDI pitch code, these differences across their averages are not meaningful in a musical sense. For example, based on Table 7-2, the *p*-value for comparing "extended 1" and "extended 3" is <0.01, which means that a significant difference between "extended 1" and "extended 4" exists statistically. Nevertheless, Table 7-5 shows that the range of the observed differences for "extended 1" is 0.9227 ± 3.075 MIDI pitch code and that the range for "extended 3" is 1.0386 ± 3.19 MIDI pitch code. This is a 0.1159 MIDI pitch code difference between the means of "extended 1" and "extended 3" (i.e., 1.0386–0.9227); this number of 0.1159 could not be considered to be a significant difference in pitch value, particularly when the range for "extended 1" is from -2.1523 to 3.9977 MIDI pitch code and for "extended 3" is between -2.1514 and 4.2286. Therefore, for estimating the fundamental pitch frequency of notes, there is no significant difference between selecting each point, as discussed in Section 7.1, to be the onset and offset. Similarly, by comparing Table 7-5 to Table 7-7, it can be concluded that there is not a considerable difference between the methods of calculating F0 (Average, AverageStd, and Median).

On the other hand, points selected as onsets and offsets can affect the duration of the notes.

Table 7-2. p-values of the t-test applied to the difference between average frequency and nominal frequency.

Series 1	Series 2	*p*-Value
extended 1	extended 2	0.084
extended 1	extended 3	<0.01
extended 1	extended 4	0.259
extended 2	extended 3	<0.01
extended 2	extended 4	0.567
extended 3	extended 4	<0.01

Table 7-3. p-values of t-test applied to the difference between AverageStd and nominal frequency.

Series 1	Series 2	*p*-Value
extended 1	extended 2	0.287
extended 1	extended 3	<0.01
extended 1	extended 4	0.784
extended 2	extended 3	<0.01
extended 2	extended 4	0.429
extended 3	extended 4	<0.01

Table 7-4. p-values of t-test on the difference between median frequency and nominal frequency.

Series 1	Series 2	*p*-Value
extended 1	extended 2	0.844
extended 1	extended 3	0.197
extended 1	extended 4	0.659
extended 2	extended 3	0.278
extended 2	extended 4	0.809
extended 3	extended 4	0.399

Table 7-5. The average and standard deviation of the difference between average frequency and nominal frequency.

Series	Average	Standard Deviation
extended 1	0.9227	3.075
extended 2	0.9627	3.113
extended 3	1.0386	3.19
extended 4	0.9492	3.194

Table 7-6. The average and standard deviation of the difference between AverageStd and nominal frequency.

Series	Average	Standard Deviation
extended 1	1.0317	2.518
extended 2	1.0539	2.52
extended 3	1.1094	2.511
extended 4	1.0374	2.516

Table 7-7. The average and standard deviation of the difference between median frequency and nominal frequency.

Series	Average	Standard Deviation
extended 1	0.8892	2.745
extended 2	0.8933	2.787
extended 3	0.9159	2.784
extended 4	0.8984	2.786

7.5 Conclusions

This chapter introduced an extended set of annotations for the solo singing files in the VocalSet dataset (Wilkins *et al.*, 2018). The provided annotations include F0, onset, offset, transition, note F0, note duration, Midi pitch, and lyric. In addition, four approaches for considering the onset and offset points in a pitch contour were compared, showing that the selected points for onset and offset cannot significantly affect the note's estimated F0. Moreover, calculating a note's F0 by average or median methods does not considerably affect the note's estimated F0. The annotated dataset is available online at https://doi.org/10.5281/zenodo.7061507, accessed on 14 September 2022.

After generating the dataset in this chapter, the next chapter aims to analyse it to figure out how to calculate a note's pitch frequency and duration according to its position in a piece of music.

Chapter 8

Models to estimate the pitch frequency and duration ranges for an acceptable note in singing

Performing musical notes correctly does not mean that all the performers will play the notes at the exact same pitch and duration. However, it does imply that they are performing the notes within acceptable psychoacoustic ranges. Therefore, this chapter aims to propose models for calculating these acceptable psychoacoustic ranges according to the position of the notes in a piece of music.

This chapter aims to investigate a dataset of recorded vocals to discover some particular aspects of the relationship between the performed F0 and duration against its written note and relative duration in a music score. In other words, this chapter introduces two novel models to simulate trained-professional singers' behaviours in singing notes' pitches and duration according to the position of the note in a piece of music and the singing technique applied. The annotated dataset introduced in the previous chapter is used for this study. Since the singers of the dataset were trained-professional singers, the mean of their singing behaviours is considered acceptable pitch frequency and duration.

The following section, 8.1, explains this study's steps in detail. Then, section 8.2 provides results, and a discussion follows that in section 8.3. Finally, this chapter is closed with a conclusion.

8.1 Materials and Methods

8.1.1 Dataset

The Annotated-VocalSet, generated in the previous chapter, was used to evaluate singers' patterns. All the singers had vocal training leading to a bachelor's or graduate degree in vocal performance. In addition, this dataset annotated nearly 7 hours of singing files comprising scales, arpeggios, long tones, and musical excerpts. Moreover, the singers used different techniques such as belt, vibrato, lip trill, straight, and breathy. Furthermore, they used all the vowels to sing arpeggios, scales, and long tones. Additionally, they sang three popular pieces: 1- Row, Row, Row Your Boat, 2- Caro Mio Ben, and 3- Dona Nobis Pacem. Figure 8-1 shows the music scores of the pieces available in Annotated-VocalSet.

As discussed in the previous chapter, the Annotated-VocalSet dataset provides four approaches for determining onset, offset, and transition between two consecutive notes.

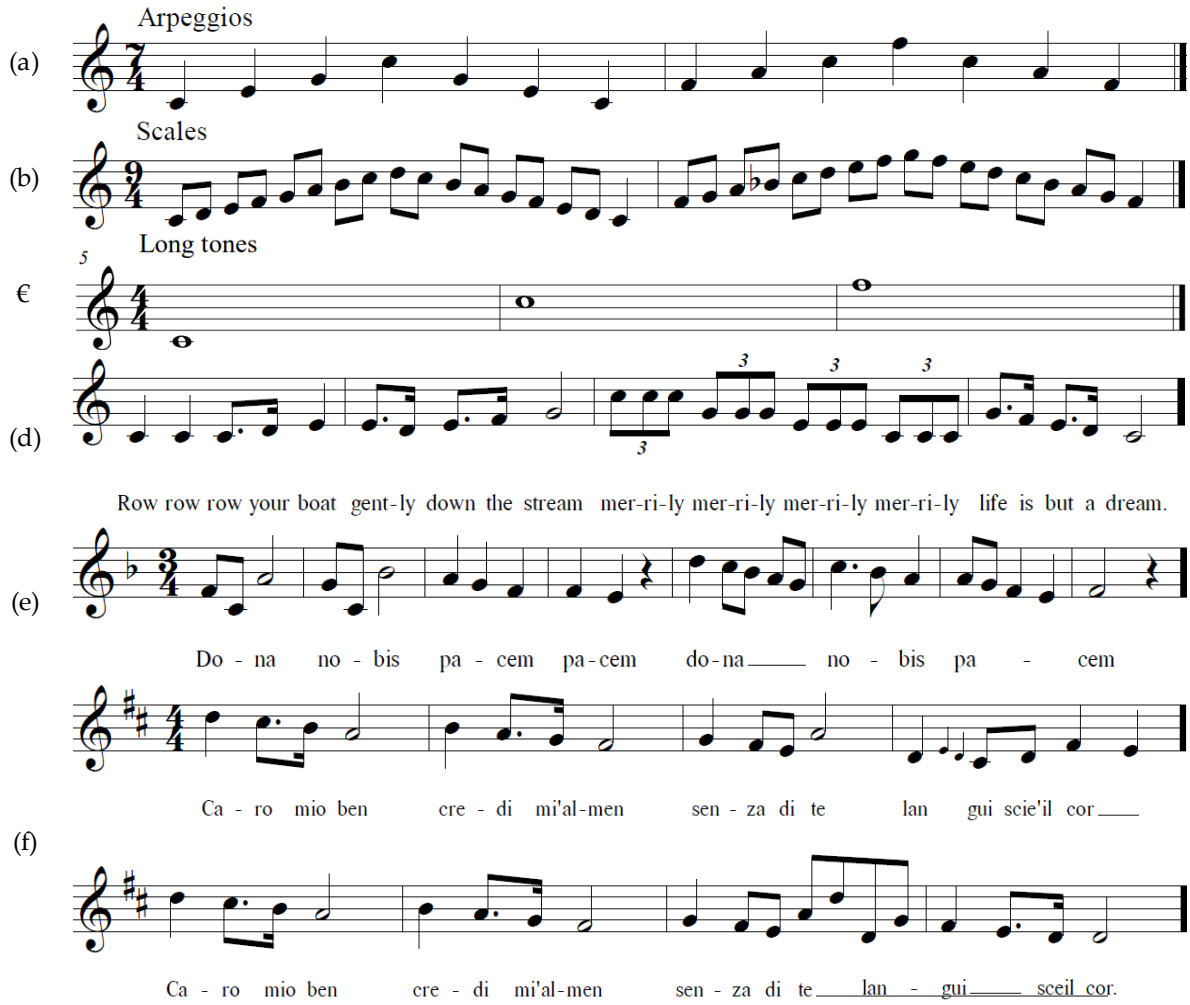


Figure 8-1 Music scores of the audio files in the Annotated-VocalSet dataset: (a) is arpeggios in C and F, (b) is C and F Scales, (c) is some long tones, (d) is the score "f "row row row your b"at", (e) is the score "f "Dona No"is", and (f) is the score "f "Caro mio"en".

As reported in the previous chapter, the estimated fundamental frequencies of notes are similar in all four approaches of selecting onset, offset, and transition. However, regarding the performed duration of notes, the selected points can affect the performed duration of the notes. This study selected the data from the dataset calculated by the fourth approach, which selects the middle points for onset and offset because considering transitions is not a common approach in datasets and onset detection. In addition, considering the transitions between notes could result in ignoring some part of the duration that the singers had intended to sing within the notes. Therefore, selecting the middle points can provide the best compromise with the singers' intentions.

In the VocalSet dataset, the singers in some performances only were asked to sing at some specific BPM, i. e., 60 and 330. Otherwise, in the remaining cases, the singers were free to sing in any BPM, which was not explicitly recorded in the dataset. Therefore, a ground truth is needed for comparisons to evaluate the effect of the BPM on the performed F0s and duration. Therefore, the two groups of files within which their BPM was indicated were employed in this study. The total number of notes played in each BPM is shown in Table 8-1 and categorised into slow and fast performances.

Table 8-1. The slow and fast categories with their BPM and the number of notes sung in each speed

Speed	BPM	Count
Slow	60	8619
Fast	330	9590
Grand Total		18209

8.1.2 Variables

According to the available annotations in the dataset, the variables evaluated to ascertain their effects or impact on the performed F0 and duration of notes are classified into eight groups. Table 8-2 and Table 8-3 provide a statistical overview of the data extracted from the dataset according to each variable. Table 8-2 details the types of variables: integer, fraction, or Boolean. These variables are explained in the following.

1. *Interval to the following note*: indicating the number of semitones between the current note and the following note. If the following note is a rest or there is no note after the current note, the value of this variable is null.
2. *Interval to the previous note*: shows the number of semitones between the current and previous note. If the previous note is a rest or does not exist, the value of this variable is null.
3. *Rest before*: it is a Boolean variable. If it is true, it means there is a rest before the current note or the current note is the first note of the music.
4. *Rest after*: this Boolean variable is true when a rest exists after the current note or the current note is the last note of the music. Otherwise, it is false.
5. *Ground truth duration*: indicating the theoretical duration of the notes in the music score.

6. *Ground truth MIDI pitch code*: shows the MIDI pitch code of the notes in the music score. The reason that the MIDI pitch code is used is that the MIDI pitch code maps the logarithmic pitch perception into a linear scale (Ishi, Hirose and Minematsu, 2003).
7. *Repeated note*: this variable has been considered to address pitch drift. If it is false, it indicates that this is the first time the note is played in the piece of music; otherwise, it is the second or more repetition of the note in the entire melody, or the note is a rest.
8. *Singing techniques*: another variable used in this study is the singing technique, which, according to the dataset, includes ten techniques, as shown in Table 8-3.

Table 8-2. A statistical summary of the notes categorised by intervals, rest, duration, and MIDI code variables

Variable	Type	Mean	Median	Standard deviation	Minimum	Maximum	Count
Interval to the following note	Integer	0.25	0	3.16	-12	12	26550
Interval to the previous note	Integer	-0.25	0	3.17	-12	12	26513
Rest after	Boolean	NA	NA	NA	NA	NA	6407
Rest before	Boolean	NA	NA	NA	NA	NA	6590
Ground truth duration (second)	Fractional	0.41	0.18	0.35	0.09	1	15493
Ground truth MIDI pitch code	Fractional	61.17	60	7.72	48	79	29267
Repeated note	Boolean	NA	NA	NA	NA	NA	15592

Several other variables could also be considered in an investigation of the parameters that affect the performed duration and F0. For example, some musical contexts could also have an impact, such as whether a note is at a beat vs non-beat position in the bar and its particular location when there is a change in the direction of the melody. However, including this information would require much effort to update the metadata of the dataset, which would be time-consuming and might also be impossible to do accurately since all the details were not available for this dataset.

Table 8-3. The list of the singing techniques and count of all the notes sung in each technique

Technique	Count
Belt Harsh	2661
Breathy	2446
Fast* Articulated	9590
Full Voice	537
Lip Trill	2040
Messa Di Voce	627
Molto Vibrato	4364
Slow* Legato	6962
Straight Tone	5350
Trill- Trillo	1128
Grand Total	35705

*Fast and Slow refer to the speed of singing the notes. The BPM of fast files were 330, and for the slow ones, it was 60.

8.1.3 Methods of evaluations

To assess the effect of various independent variables, outlined in the previous section, on the deviations in trained-professional singers' performance from the notes' duration and pitch as written in the original score and exactly equivalent to the MIDI pitch code description, Bayesian Hierarchical Linear Regression (BHLR) was used. The Bayesian model was used because it allowed us to account for the dependency of the individual-specific observations and infer both individual-level and population-level parameters. Furthermore, using the Bayesian approach, we were able to measure the uncertainty in our inferences using probability distributions (Gelman *et al.*, 1995; Dobson and Barnett, 2018).

Considering that our data are grouped by individual singers (each singer has sung various notes), we let the parameters in the linear regression to be random (varying according to the singers) and defined a hierarchical structure on the parameters to estimate the population effects. We denote y_i as the i^{th} measurement of the differences between the estimated value, the F0 or duration sung by the singers, and the ground truth value. Two different distributions for the observational model were tried. First, a Gaussian distribution was tried; however, the model was underfit due to the residuals' distribution having heavy tails. Then, the Logistic distribution was tried that had a better fit for our data. An explanation of the

model validation procedure is provided in section 8.1.5. Here, the model's description with Logistic distribution is given in the following, as the former was not selected for further analysis.

It is denoted $j = 1,..,J$ as individuals to distinguish the individual specific parameters. We write the model as follows (8-1):

$$y_i \sim Logistic(\mu_i, s_j) \tag{8-1}$$

where μ is the mean value for y and calculated by equation (8-2); and s is the scale parameter indicating a variance of $\sigma^2 = \frac{s^2\pi^2}{3}$ for the residuals.

$$\mu_i = tech_{(j,k)[i]} + \sum_{l=1}^{N} \beta_{(l,j)[i]} \times X_{l,i} \tag{8-2}$$

In equations (8-2), $tech_{(j,k)}$ is the singing technique effect for singer j and technique k, from the techniques listed in Table 8-3. In addition, $\beta_{(l,j)}$ is the individual specific effect of variable l, from the variables listed in Table 8-2. X is the design matrix, elements of which are indexed by i and l for its rows and columns respectively.

According to equation (8-2), two models were created: one for calculating the F0 deviation from the ground truth, i.e. represented by the MIDI pitch code that is the expected F0 deviation from the theoretical pitch that a trained-professional singer will sing; and the other is calculating the difference from the ground truth duration that is expected a trained-professional singer to sing. The variables considered for both models are the same as given in Table 8-2 and Table 8-3, except for the duration model, where the "Repeated note" variable was not employed since this variable was used only for considering pitch drift.

The JAGS software (Plummer, 2003) and R (R Core Team, no date) were used to fit the model. In addition, three Markov chain Monte Carlo chains were employed (typical simulation values were used, that is, 2000 iterations per chain with 1000 as burn-in, and a thinning value of 1). Finally, the R-hat diagnostic (Gelman *et al.*, 1992; Brooks and Gelman, 1998) was applied to assess the convergence. All R-hat values linked to the model parameters were close to 1, so the model was believed to be sampling from the posterior distribution.

Furthermore, we used the following prior distributions, equations (8-4) and (8-5), for the random effects:

$$tech_{(j,k)} \sim N\left(\mu'_{tech_k} \sigma^2_{tech_k}\right) \quad (8\text{-}3)$$

$$\beta_{l,j} \sim N(\mu_l, \sigma_l^2) \quad (8\text{-}4)$$

Where j is the index of the singer, k is the index of the singing technique, l points to the variables, and N refers to normal distribution.

The hyperprior distributions used to fit the model are as follows, equations (8-5)-(8-9):

$$\mu_{tech_k} \sim N(0,1) \quad (8\text{-}5)$$

$$\mu_l \sim N(0,1) \quad (8\text{-}6)$$

$$\sigma_k^2 \sim N(0,1)^+ \quad (8\text{-}7)$$

$$\sigma_l^2 \sim N(0,1)^+ \quad (8\text{-}8)$$

$$s \sim N(0,1)^+ \quad (8\text{-}9)$$

Considering the range of the dependent variables specified in Table 8-2 and Table 8-3, the above prior distributions have negligible effects on the posterior distribution of the parameters and thus can be considered uninformative. In other words, we let the data speak for themselves, and the results are not sensitive to these priors. As stated in Section 8.1.2, MIDI pitch codes are used to represent the pitches, and to reiterate, the reason was that humans' perception of pitch is not linear but logarithmic. That is, the human brain considers the 100 Hertz difference between 50 Hertz and 150 Hertz is perceptually much more significant than the same 100Hz difference between 1000 Hertz and 1100 Hertz. However, using the MIDI pitch code representation, the perceptual difference between say the 40 and 50 MIDI pitch codes is as same as that difference heard between say the 70 and 80 MIDI pitch codes.

To compare fundamental frequencies, the difference between the performed MIDI pitch code, P_{MIDI}, and the ground truth MIDI pitch code, GT_{MIDI}, code was calculated as shown in (8-10).

$$D_{MIDI} = P_{MIDI} - GT_{MIDI} \quad (8\text{-}10)$$

To compare the duration of the performed notes with their corresponding ground truth duration, the theoretical duration of notes in the music score is considered to be the ground

truth, and the difference between the performed duration, $P_{duration}$, and the ground truth, $GT_{duration}$, calculated as (8-11).

$$D_{duration} = P_{duration} - GT_{duration} \quad (8\text{-}11)$$

8.1.4 Estimating the range of a note's duration and F0 sung by trained-professional singers

Since all the singers of the VocalSet dataset were trained-professional and in the Annotated-VocalSet, any incorrect files were removed, we considered the range of the F0 and duration sung by these singers across all the recordings as a trustable resource. Thus, according to the resource, we can define a range of deviations of note's duration and F0 from their ground truth in trained-professional singers' performances. To do this, it is possible to use the models and sample from the posterior predictive distribution for a given note. Posterior predictive distribution is the distribution of the predicted values given the model parameters and the data. Nevertheless, since this approach is computationally demanding, we use an alternative empirical approach that is faster and easier to calculate and gives reasonable ranges. For this purpose, for each data point in our dataset, first, we calculate the length of the 80% prediction credible interval, that is, the 90% quantile minus the 10% quantile of the posterior predictive distribution. This decision is made based on experiments on our data showing that the 80% interval is neither too conservative nor too wide an acceptable range. As the second step, we take the average of all the intervals' lengths over the entire dataset as the average empirical prediction length (ePL). This is similar to the length of prediction confidence interval in the Frequentist statistical approaches.

Finally, we define the acceptable deviation from the ground truth to be $mu - ePL/2$ to $mu + ePL/2$, where mu is simply the mean deviation calculated using the equation (8-2). In other words, for a given note, we first calculate its mean deviation and then consider half ePL below and half ePL above that as the anticipated range of deviation in trained-professional singers' performance. Table 8-4 shows the ePL figures for 80% intervals for the deviations in MIDI pitch code and duration.

To calculate the range of a note's F0 and duration, we simply add the theoretical value of the note to the predicted deviation range of that note.

Table 8-4. The average empirical prediction length (ePL) to calculate the ranges of deviations in F0 and the duration of the notes sung by all the singers.

	80% *ePL*
F0	0.8977
Duration	0.5698

8.1.5 Model validation

To examine our model's validity and confirm its ability to mimic the true data generating process, we follow Gelman et al.'s (Gelman *et al.*, 1995, p. 143) suggested approach and conduct a posterior predictive check analysis. We generate 1000 samples from the posterior distribution of the model. We then compare the distribution of the predicted values with the observed values. The results are shown in Figure 8-2. The posterior predictive distributions are very similar to the distributions of the observations for both models of MIDI pitch code and duration deviations. Furthermore, we calculate the percentage of the occasions that true observations fall within the 95% credible interval of the posterior predictive distribution. This gives an insight into the performance of the model in uncertainty quantification. We found that for deviations of MIDI pitch code and duration models, the percentage was 95% and 96%, respectively, which indicates good uncertainty calibration.

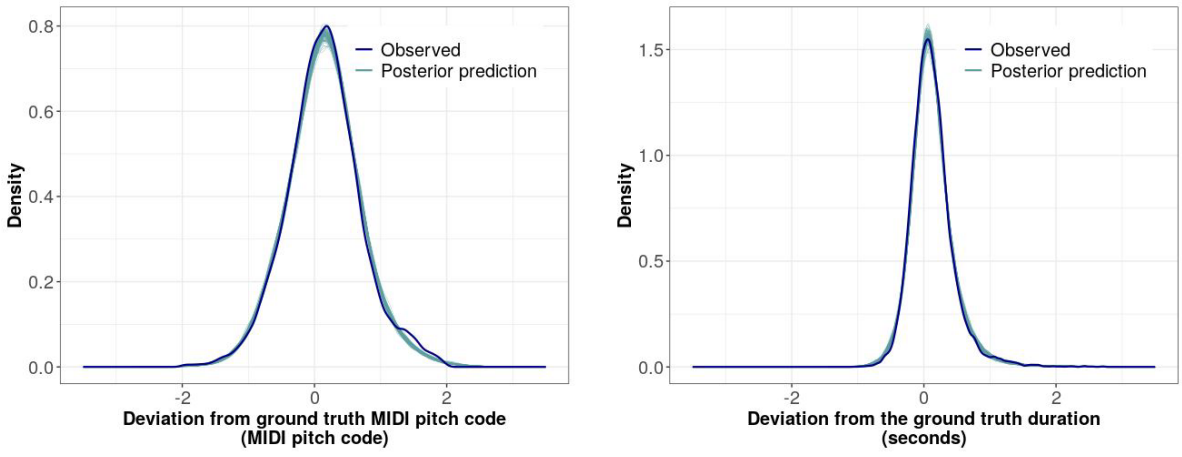


Figure 8-2 Comparison of the density of the distribution of the predicted values and the distribution of the observations. (a) Shows the comparison for the deviation in the MIDI pitch code model, and (b) shows the comparison for the deviation in the duration model.

8.2 Results

In the following, the effect of each variable, mentioned in 8.1.2, on the deviation of performed duration and F0 from the ground truth in trained-professional singers' performance is presented . Thus, the posterior distribution of the parameters denoted in equations (8-5)-(8-6), which are the parameters of interest associated with the effects of the independent variables in the population will be reported.

8.2.1 Estimating the effect of the variables on deviation from ground truth F0

The effect of the variables on deviation from the ground truth MIDI pitch code reported in Table 8-5 and Table 8-6. The *Mean* column in Table 8-5 indicates the average change in the deviation from the ground truth MIDI pitch code when the variable is increased by one unit. Increasing by one unit means that, for example, when one second is added to the duration of the note for the "Duration" variable, or the number of the semitones between the current note and the following note increased by one for the "Interval to the following note" variable. In addition, Table 8-6 provides the expected effect of each singing technique. These tables also include the 95% credible interval for each effect. These credible intervals express that the true parameter falls within the interval with a probability of 95%. That is, 95% of the probability of the true effect is between the values in the columns titled *2.5%* and *97.5%* in Table 8-5 and Table 8-6. For example, according to Table 8-5, the 95% probability of the effect of the Ground truth duration variable is between -0.0474 and 0.0413. In other words, the 95% credible interval shows the range of uncertainty about the effect.

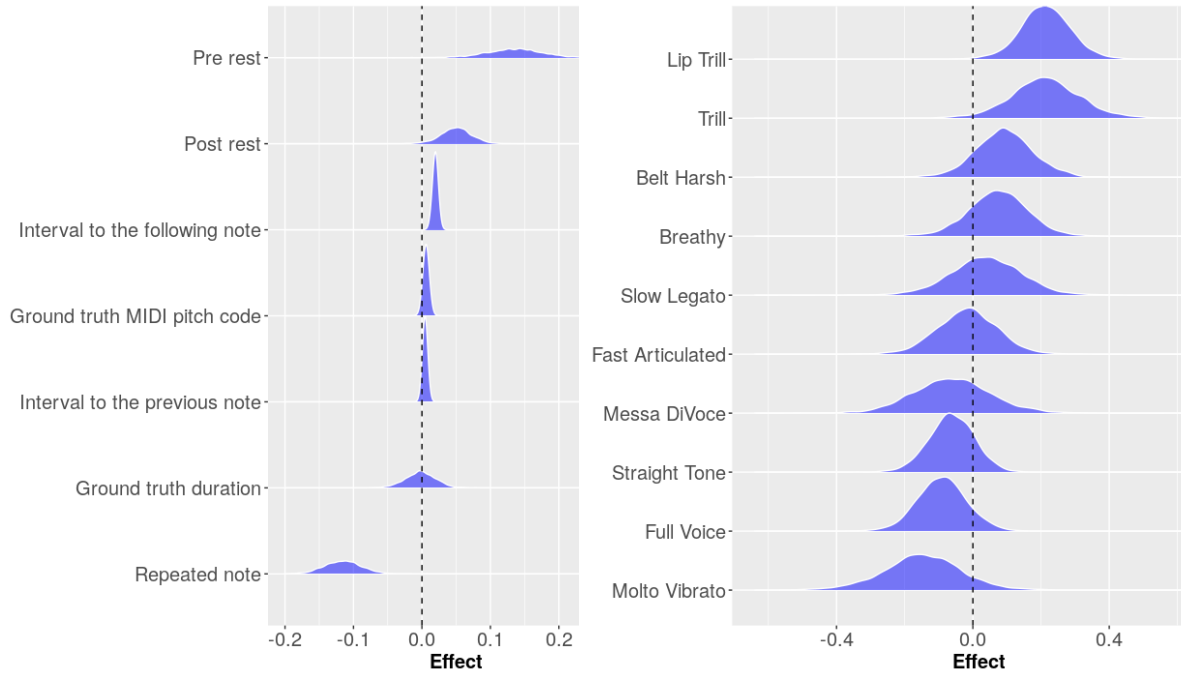
Furthermore, Figure 8-3 depicts the posterior distribution of the values shown in Table 8-5 and Table 8-6. According to this figure, we can see that different effects have different distributions indicating a varying level of uncertainty among them. For example, "Ground truth MIDI pitch code" and Interval to the effects of the previous and following notes have very narrow distributions, showing that the model is more certain about their effects than other effects with wider distributions.

Table 8-5. The mean and the 95% credible interval of the effect of the independent variables on deviation from the ground truth MIDI pitch code.

Variable	Mean	2.5%	97.5%
Ground truth duration	-0.0022	-0.0474	0.0413
Ground truth MIDI pitch code	0.0061	-0.0012	0.0134
Interval to the following note	0.0196	0.0132	0.0258
Interval to the previous note	0.0046	-0.0005	0.0099
Post rest	0.0499	0.0032	0.0933
Pre rest	0.1356	0.0504	0.2224
Repeated note	-0.1158	-0.1679	-0.0632

Table 8-6. The mean and the 95% credible interval of the effect of the singing techniques on deviation from the ground truth MIDI pitch code

Technique	Mean	2.5%	97.5%
Belt Harsh	0.0925	-0.0762	0.2656
Breathy	0.0732	-0.1034	0.2436
Fast Articulated	-0.0167	-0.1915	0.1535
Full Voice	-0.0903	-0.2311	0.0533
Lip Trill	0.2137	0.0641	0.3605
Messa DiVoce	-0.0609	-0.2802	0.1705
Molto Vibrato	-0.1482	-0.3833	0.0770
Slow Legato	0.0387	-0.1719	0.2406
Straight Tone	-0.0625	-0.1950	0.0643
Trill	0.2128	0.0110	0.4217



(a) (b)

Figure 8-3. The posterior distribution of the effects of the independent variables on deviation from the ground truth MIDI pitch code. (a) shows the impact of the numerical and Boolean variables, and (b) depicts the effect of the singing techniques.

8.2.2 Estimating the effect of the variables on the deviation from the ground truth duration

The theoretical duration of notes in the music score as the ground truth was needed to estimate the range of a note's duration in trained-professional singers' performances. However, in the VocalSet dataset, the BPM of all the files were not indicated. Thus, only the files with ground truth BPM were selected, and as a result, only some of the singing techniques were sung with a specific speed, as shown in Table 8-8. The mean value and its 95% credible interval of the effect of the independent variables and singing techniques on deviation from ground truth duration, equation (8-2), are presented in Table 8-7 and Table 8-8; the posterior distribution of mentioned parameters is provided in Figure 8-4.

The same approach as estimating a note's MIDI code, Section 8.2.1, should be used to calculate duration.

Table 8-7. The mean and the 95% credible interval of the effect of the independent variables on deviation from the ground truth duration.

Variable	Mean	2.5%	97.5%
Ground truth duration	-0.2054	-0.3913	-0.0174
Ground truth MIDI pitch code	0.0005	-0.0009	0.0020
Interval to the following note	-0.0005	-0.0032	0.0024
Interval to the previous note	0.0015	-0.0004	0.0034
Post rest	0.4112	0.2945	0.5242
Pre rest	0.0961	0.0682	0.1265

Table 8-8. The mean and the 95% credible interval of the effect of the singing techniques on deviation from the theoretical duration.

Variable	Mean	2.5%	97.5%
Fast Articulated	0.0031	-0.0782	0.0830
Molto Vibrato	0.2155	0.0727	0.3614
Slow Legato	0.1442	0.0544	0.2340
Straight Tone	0.1266	0.0170	0.2497

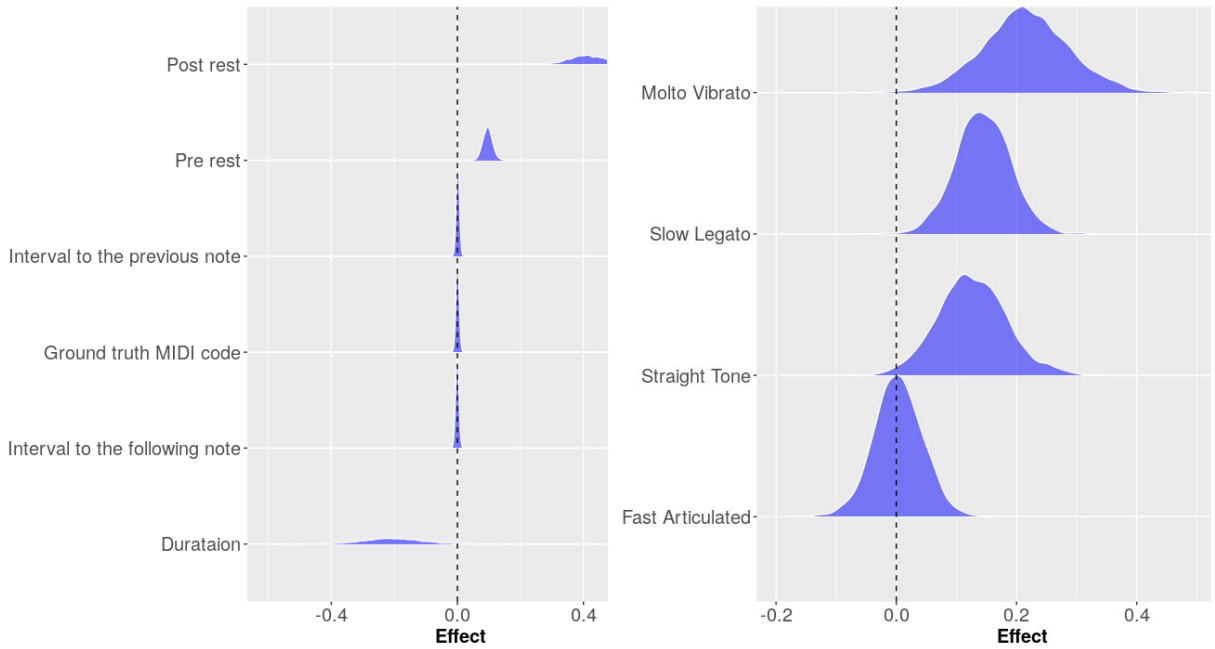


Figure 8-4. The posterior distribution of the effects of the independent variables on deviation from the ground truth duration. (a) shows the impact of the numerical and Boolean variables, and (b) depicts the effect of singing techniques.

8.3 Discussion

8.3.1 An illustration to show how to calculate the expected MIDI pitch code and duration of notes

This section provides an example to clarify how to calculate the expected notes' MIDI pitch code and duration in trained-professional singers' performances according to the results of this study. This example illustrates how to use the numbers in Table 8-5 to Table 8-8. If the musical scores are the notes shown in Figure 8-5, according to equations (8-2), and using the coefficients in Table 8-5 and Table 8-6, the expected MIDI pitch code and duration of the notes can be estimated. To calculate the expected values, first, the coefficients in equation (8-2) should be assigned by the values in the column title *Mean* in Table 8-5 and Table 8-6 to achieve the expected variation from ground truth MIDI pitch code or duration. Then the calculated expected variation should be added to the ground truth value to obtain the expected MIDI pitch code or duration of the note.

Table 8-9 shows the values of each variable, the X variable in equation (8-2), according to the score in Figure 8-5. When the following or previous notes are a rest, the interval to the following and previous notes should be considered to be 0. Nevertheless, two other variables, post- and pre-rest, calculate the effect of these situations.

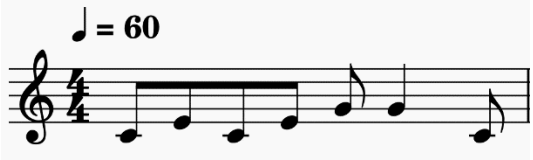


Figure 8-5. Musical scores as an example for estimating the pitch and duration of the notes

Table 8-9. The values of each of the variables for the notes in the score in the example provided in Figure 8-5.

Note	Ground truth MIDI pitch code	Ground truth Duration (in second)	Interval to the following note (in semitone)	Interval to the previous note	Post rest	Pre rest	Repeated note
C4	60	0.5	4	NA	False	True	False
E4	64	0.5	-4	-4	False	False	False
C4	60	0.5	4	4	False	False	True
E4	64	0.5	3	-4	False	False	True
G4	67	0.5	0	-3	False	False	False
G4	67	1	-7	0	False	False	True
C4	60	0.5	NA	7	True	False	True

Equation (8-2) needs to be used to calculate the expected pitch and duration. The effect of the singing techniques, $tech$, is provided in Table 8-6 and Table 8-8, the effect of each variable, β, shows in Table 8-5 and Table 8-7, and the value of the variables can be seen in Table 8-9. Thus, by assigning these values in equation (8-2), the expected deviations of each of the notes from their ground truth MIDI pitch code and duration can be calculated. Finally, the deviation should be added to the ground truth value. Therefore, Table 8-10 shows the expected MIDI pitch code for each note by employing the mean values of the effects and the Straight Tone as the singing technique.

In addition, the minimum and the maximum expected MIDI pitch code and duration of each note are calculated employing empirical prediction length (ePL), as discussed in section 8.1.4. In this example, to calculate the expected ranges, the ($ePL/2$) should be deducted and added to the expected values in Table 8-10. That is, the value $0.8977/2 = 0.4488$ based on Table 8-4 should be deducted and added to all values in the column titled "Expected MIDI pitch code" in Table 8-10 to calculate the minimum and maximum of anticipated MIDI pitch codes range. Similarly, the value $0.5698/2 = 0.2849$, according to Table 8-4, should be deducted and added to the values in the column titled "Expected duration" in Table 8-10.

Table 8-10. *The expected MIDI pitch code and duration of the notes and their anticipated ranges according to Figure 8-5 and Table 8-9. The Straight Tone was considered the singing technique.*

Note	Ground truth MIDI pitch code	Ground truth Duration (in second)	Expected MIDI pitch code	Minimum accepted MIDI pitch code	Maximum accepted MIDI pitch code	Expected duration (in second)	Minimum accepted duration (in second)	Maximum accepted duration (in second)
C4	60	0.5	60.5134	60.0646	60.9623	0.651	0.366	0.936
E4	64	0.5	64.2269	63.7781	64.6758	0.555	0.270	0.840
C4	60	0.5	60.2805	59.8316	60.7293	0.561	0.276	0.846
E4	64	0.5	64.2482	63.7993	64.6971	0.551	0.267	0.836
G4	67	0.5	67.3280	66.8792	67.7769	0.556	0.271	0.841
G4	67	1	67.0878	66.6390	67.5367	0.961	0.676	1.246
C4	60	0.5	60.2659	59.8170	60.7147	0.979	0.694	1.264

Figure 8-6 visualises the values in Table 8-10. The black-solid lines show the ground truth MIDI pitch code and duration, and the red-dashed lines show the expected MIDI pitch code and the expected range of MIDI pitch code for each note. The middle panel in Figure 8-6 shows

the expected duration of the notes, while the top and bottom panels show the maximum and minimum anticipated duration of notes, respectively.

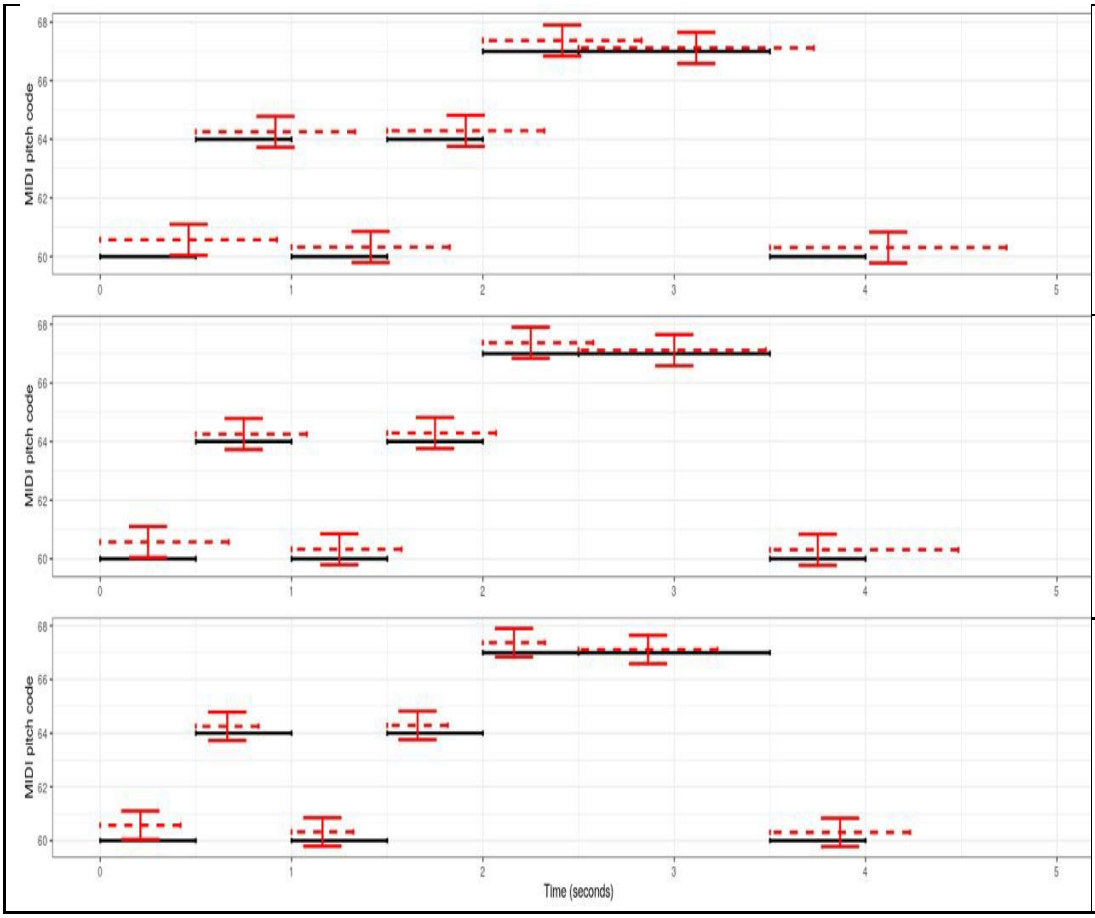


Figure 8-6. *A visual representation of the expected MIDI pitch code and duration of the notes and their anticipated ranges, according to Table 8-10. The black-solid lines show the ground truth MIDI pitch code and duration, and the red-dashed lines are the expected MIDI pitch code and the boxes around them show the anticipated range of each note. The top panel shows the maximum anticipated duration of the notes, the middle panel shows the expected duration of the notes, and the bottom panel show the minimum anticipated duration.*

This example illustrates how the position of the notes in a piece of music can affect their expected pitch and duration. For example, C4 is repeated three times in Figure 8-5, but their expected MIDI pitch code are different, 60.573, 60.325, and 60.308, respectively. Similarly, the estimated duration of each appearance of the C4 is different, in sequence, 0.672 sec, 0.575 sec, and 0.985 sec. Likewise, Table 8-10 shows that the expected MIDI pitch code and duration of the notes E4 and G4 are different in each appearance because of their positions in the musical score. In addition, the eighth notes have different expected duration.

8.3.2 The effect of rest before or after a note on the deviation of its performed F0 and duration from the ground truth

Table 8-5 and Figure 8-3 show that the existence of a rest before a note has a greater impact on the deviation from the theoretical note's pitch than when the rest comes after a note, resulting in a change to the MIDI pitch code value of the order of 0.1356 MIDI pitch code (or 13.56 cents) as compared to 0.0499 MIDI pitch code (or 4.99 cents).

An opposite observation was noticed for the effect of a rest on the deviation from the theoretical duration. As shown in Table 8-7 and Figure 8-4, a rest after a note has a more significant effect on the deviation than the rest before a note, that is, 0.4112 s and 0.0961 s, respectively.

8.3.3 The effect of the ground truth MIDI pitch code on the deviations of its performed F0 and duration from ground truth

According to Table 8-5 and Figure 8-3, although the effect of the MIDI pitch code on the deviation from ground truth pitch is a small value of 0.0061 MIDI pitch code (or 0.61 cents), since the pitch range of the notes to be sung is wide, generally from 77 Hz (almost D#2 or 39 MIDI pitch code) to 900 Hz (approximately A5 or 81 MIDI pitch code) (Heylen *et al.*, 2002), this difference in the higher note could be significant. For example, the pitch difference in MIDI pitch code 40 (note E2) is $40 * 0.0061 = 0.244$ MIDI pitch code (or 24.4 cents), but for MIDI pitch code 80 (note G#5) $80 * 0.0061 = 0.488$ MIDI pitch code (or 48.8 cents). Sundberg (Sundberg, 2013) had a similar finding that singers tend to sharp the higher pitch frequencies.

A similar observation can be seen in the effect of the ground truth notes' pitches on their performed duration. It is expected that by increasing one MIDI pitch code (one semitone), a singer sings the note 0.0005 seconds longer.

8.3.4 The effect of the ground truth note's duration on the deviations of its performed F0 and duration from the ground truth

As shown in Table 8-5, the ground truth duration of a note has a minimal impact on the deviation from the theoretical note's pitch, and the posterior distribution of its effect is centred on zero, as shown in Figure 8-3.

On the other hand, according to Table 8-7, the effect of the ground truth duration on the deviation from the ground truth duration is significantly negative, meaning that a unit increase (one second) in the theoretical duration of a note, is associated with a -205.45 milliseconds deviation from the ground truth on average when the singers sing the note.

8.3.5 The effect of the pitch interval to the previous and following notes on the deviations of performed F0 and duration from the ground truth

The interval to the following note has a significant positive effect on the deviation from the theoretical pitch of 0.0196 MIDI pitch code (or 1.96 cents), as shown in Table 8-5. Similarly, the interval to the previous note also has a positive effect on pitch but less than the interval to the following note, that is, 0.0046 MIDI pitch code (0.46 cent).

However, the effect of the pitch intervals on the deviation from the theoretical duration is negligible, and its posterior distribution is centred almost on zero, according to Figure 8-4. In addition, usually, the pitch intervals in singing are small. However, if there is a big interval with 24 semitones apart (two octaves), multiply the 24 by the expected interval effect to the following note, -0.0005 s, and the previous note, 0.0015 s, would be -12 milliseconds and 36 milliseconds, respectively.

8.3.6 The effect of the singing techniques on the deviation of the performed F0 and duration from the ground truth

For interpreting the effect of singing techniques, it should be noted that they are the intercepts of the linear equation (8-2). The numeric variables were centred at zero by removing their average before fitting the model, and hence these intercepts should be interpreted as the deviation from the ground truth when the other numeric independent predictors are at their average values, and the Boolean predictors are zero (meaning no pre-rest and post-rest). According to Figure 8-3 and Table 8-6, the Lip Trill technique has a significant positive deviation from the ground truth pitch, followed by the Trill technique, which has a lower probability of being greater than zero compared to the Lip Trill. On the other hand, the Molto Vibrato, Full Voice, and Straight Tone techniques have a considerable probability for their expected deviation from the ground truth pitch to be negative, although their 95% credible intervals contain zero. The other techniques have large probabilities on

either side of zero, and their effects can be assumed neither significantly positive nor significantly negative.

Regarding the effect of singing techniques on the deviation from the ground truth duration, according to Table 8-8 and Figure 8-4, the Molto Vibrato followed by Slow Legato and Straight Tone have significant positive effects. In contrast, the Fast Articulated technique has its posterior distribution centred on zero and thus cannot be assumed to be significantly different from zero.

According to the study by (Sundberg, 1994), the rate of the vibration in singing can vary the average of the fundamental frequencies. They found that the average pitch of a vibrato tone remains constant only if the vibrato rate is more than 4 Hz. In addition, they observed that the human pitch perception for vibrato and vibrato-free tones were the same.

Besouw et al. (Besouw, Brereton and Howard, 2008) presented three-tone ascending and descending arpeggios to musicians. The tuning of the middle tone, which either had or lacked vibrato, was varied, and the listeners were asked to decide which notes were in tune or untune. The results showed that the range of acceptable intonation of the middle tone was, on average, about 10 cents wider when it had vibrato than when it lacked vibrato. In addition, they found that if two voices sing perfectly "straight" (i.e., without vibrato), the demands on accuracy concerning the F0 are higher than if they sing with vibrato (Sundberg, 2013). Our model shows almost the same findings as these studies. According to Table 8-6, when singers sing the notes in the Straight Tone, the MIDI pitch code variation is -0.0625 MIDI pitch code (-6.25 cents), while the corresponding value for the Molto Vibrato technique is -0.1482 MIDI pitch code (-14.82 cents). Thus, singing in the Straight Tone is 8.6 cents more accurate than singing in the Molto Vibrato technique, which is close to the findings of the other studies.

8.3.7 The effect of the note's repetition on the performed F0 deviation from the ground truth pitch (pitch drift)

Pitch drift or intonation drift means changes in tuning over the course of a timescale of seconds or more during the playing of a piece of music (Seaton, Pim and Sharp, 2013). According to (Alldahl, 2006; Ryynänen and Klapuri, 2006), pitch drift mainly occurs in the downward direction, i.e., downward intonation drift. In another study done by Müller et al.

(Müller, Grosche and Wiering, 2010), it is observed that pitch drift is common in unaccompanied solo folk singing. Similarly, Mauch et al. (Mauch, Frieler and Dixon, 2014) also found evidence of pitch drift in solo singing. They (Mauch, Frieler and Dixon, 2014) also realised that the pitch drift extent is often small (less than 20 cents over 50 notes) and not correlated to pitch accuracy, interval accuracy, or musical background. Unlike the other studies, Mauch et al. (Mauch, Frieler and Dixon, 2014) observed that the most significant drifts are upward.

This chapter's finding is similar to most other studies that found that the pitch drift is downward. As shown in Table 8-5, the effect of a note's second or later appearing in the singers' performances is an average of -0.1158 MIDI pitch code (or -11.58 cents) deviation from the ground truth pitch with the 95% credible interval of -0.1679 to -0.0632, which makes the effect significantly negative. That is, the singer sang the note at a lower pitch than was used on the first occasion of singing these notes.

8.4 Conclusions

This chapter provides two models, one for calculating the expected F0 and another for estimating the expected duration of a note in a piece of music according to the note's conditions in trained-professional singers' performances. These models simulate trained-professional singers' behaviour in changing the pitches and duration of the notes according to their positions in a piece of music and the singing technique applied. The note's conditions considered in this study were the existence of a rest before or after the note, the pitch interval to the following or previous notes, the theoretical note's duration and MIDI pitch code in the music score, the singing techniques, and the repetition of the note. All these variables impact the expected pitch and duration, although their level of impact will depend on the interpretation the singer wants to impart to the music, both deliberately and subconsciously. For example, having a rest before a note impacts the expected pitch frequency more than when the rest is after the note. On the contrary, having a rest after a note has a higher impact on the note's duration than when the rest is before the note.

Chapter 9

Conclusion and Future work

This chapter makes a conclusion on the all the previous chapters. In addition, some suggestions for future work will be provided.

This thesis comprises several sub-studies on processing singing signals, especially in real-time environments. The studies are mainly related to estimating fundamental frequencies, smoothing F0 contour, estimating the onset, offset, and transitions between notes, generating an annotated singing dataset, and examining trained-professional singers' behaviour in singing notes' pitch frequencies and duration according to the position of the note in a piece of the music and the singing technique applied.

This thesis includes five principal objectives listed in section 1.4 that Chapters 3 - 8 provide answers for each of them. In the following, conclusions and future work for each of the objectives of this study are provided.

9.1 Conclusion and future work of investigating real-time singing pitch detector algorithms study

Two separate studies have been done in this thesis to find a reliable real-time pitch detection algorithm for singing signals from the existing pitch detection algorithms. The details of these two studies are provided in Chapters 3 and 4.

Chapter 3 compared two offline algorithms, PYIN and PRAAT, with two real-time ones, PLL and ECKF. I have experienced that these real-time algorithms did not generate a reasonable pitch estimation. On the other hand, the PYIN and PRAAT algorithms worked well, but they are offline.

Although Chapter 3's study could not find a reliable real-time pitch detector algorithm for singing signals, two reliable offline pitch detection algorithms, PYIN and PRAAT, were found that have been used in the other studies of this thesis.

Therefore, a comprehensive evaluation of seven real-time pitch detector algorithms was conducted in another study, explained in Chapter 4. Three measurements were considered to compare the functionality of the algorithms. The measurements were: 1- the number of pitches estimated correctly by categorising them based on gender, window size, the speed of the music, and post-processing, 2- the delay of each algorithm to estimate pitches correctly, and 3- the approaches to evaluate the accuracy of the estimated F0. Moreover, three methods for finding an acceptable range were assessed.

According to the results in Chapter 4, the overall best real-time algorithm from the seven tested algorithms for female voices was YinFFT, with a window size of 1024 when the sampling rate was 44100.

In addition, the speed of performance is not an issue. Moreover, the delay before starting to determine the correct pitches was found to be 25ms. Furthermore, the best real-time algorithm from the seven tested algorithms for male voices was Yin when the notes are playing fast, and in slow performance, the best one was YinFFT. The algorithms produced a more accurate pitch contour with a window size of 2048 compared to a window size of 1024 when the sampling rate was 44100. The delay before finding the correct pitches for the Yin algorithm was 107ms, and for YinFFT was 71ms. Additionally, the length of the intervals between notes does not impact the pitch accuracy of the delay.

Finally, the best method from the three presented methods to find the acceptable range for all the algorithms is the percentage, although, for FComb and MComb, significant differences between the three methods were not observed.

Therefore, these studies provided guidance for selecting a pitch detection algorithm according to the features of the singing signals.

However, the accuracy of the pitch detection algorithms was not as I expected. Before conducting the studies, I believed one might work better than another, but I did not expect the pitch contours generated by all the algorithms to be unreliable and untrustworthy.

Regarding future work, there are areas of immediate future work and other issues that require a more long-term strategy.

The studies in Chapter 3 and Chapter 4 could be more comprehensive. For example, more pitch detection algorithms/libraries, such as Librosa (McFee *et al.*, 2015), Madmom (Böck *et al.*, 2016), Essentia (Bogdanov *et al.*, 2013b), and TorchAudio (Yang *et al.*, 2021) in Python, could be evaluated. In addition, a wider range of window and hope sizes could be evaluated.

Another missing part of my study was to check if an adaptive window size (Nisar, Khan and Tariq, 2016) can improve robustness. In addition, different approaches to adaptive

window sizing should be evaluated in real-time environments to determine their performances in singing signals.

All the evaluations in this study were applied to human voices, albeit in a musical context. It would be valuable to see if the same experimental parameters applied to this study would produce comparable results when applied to the pitch determination of musical instruments.

9.2 Conclusion and future work of real-time smoothing pitch contours generated from singing signals study

Chapter 5 introduced a new pitch-contour-smoother targeted toward the singing voice in real-time environments. The proposed algorithm is based on the median filter and considers the features of fundamental frequencies in singing. The algorithm's accuracy was compared with 35 other smoother techniques, and four metrics evaluated their results: R-Squared, Root-Mean-Square Error, Mean Absolute Error, and F0 Frame Error. The proposed Smart-Median algorithm achieved better results across all the metrics in relation to the other smoother algorithms. According to this study, a buffer delay of 35 to 70 milliseconds was required for the algorithm to smooth the contour appropriately.

Most general smoother algorithms were not found to be suitable for smoothing the pitch contour of singing signals. A general observation is that in the ideal case, a smoother algorithm should be defined based on the essential features of the data in the contour and how that data is to be used after smoothing.

When I searched for a proper pitch smoother algorithm and implemented some of them, I was surprised to see why their results were unreliable and that still several errors existed after the algorithms smoothed the singing pitch contours. After designing my algorithm, I found that it is not an easy task because of the variety of possible errors in estimated pitch contours. Thus, although my algorithm could significantly improve the pitch contours in comparison to the other algorithms, the task of smoothing pitch contours requires further research to achieve the goal of error-free contours.

For future work, one short-term task is based on recognizing that the parameters of the Smart-Median can be set according to the specific properties of the sound input, such as those of particular musical instruments or their families, to improve accuracy in a targeted way.

Another task considers that the Smart-Median can determine an incorrect F0 based on its interval from the previous F0; this approach can be improved by considering a maximum randomness duration. For example, if there is a considerable frequency interval between the previous F0 and the current one, or if several immediately subsequent F0s are near the current F0, we may not consider the large jump to be noise but rather a new musical articulation. This requires the introduction of an extra decision-making stage into the algorithm.

In the longer term, further testing can be carried out on vocal material from various genres and techniques. This would require the creation of new, specialist corpora, requiring considerable manual effort in both the gathering and labelling. This can be supported by machine learning. Such a dataset would also benefit the research field at large.

9.3 Conclusion and future work of real-time onset, offset, and transition extraction from singing signals study

Estimating onset, offset, and transition between notes in singing was another main objective of this thesis. Unfortunately, Application of the existing onset detection algorithms could not reach an appropriate result. Thus, in Chapter 6, a novel algorithm has been introduced for detecting the beginnings, endings, and transitions between notes in singing performances. The algorithm is designed to operate effectively in both offline and real-time scenarios. A 57-millisecond delay is necessary for real-time applications to ensure sufficient information for accurate event calculation. It was shown to exhibit enhanced accuracy by comparing the proposed algorithm against eight established algorithms using two distinct datasets.

Nevertheless, estimating onsets was not as easy as I had expected. First, I considered evaluating the changes in the amplitude contours because I believed that when a singer starts singing a new note, there should be a clear change in the amplitude contour. However, when I plotted the amplitude contours, their shapes were much more complicated than those of the pitch contours. Thus, I started working on pitch contours as a means by which that onsets

could be estimated. After a while, I realized that in real singing, especially when some singing technique like vibrato is used, it is difficult to figure out where vibration changes the pitch frequency and where a new note appears. Therefore, onset detection requires more research to be able to detect onsets without error.

The accuracy of the proposed algorithm may be improved by considering more spectrogram channels, i.e., including other related frequency components from the spectrogram and not only the fundamental frequencies. In this way, a more comprehensive formula weighted together with the measurements for each channel could be fused to improve the overall measure. Therefore, a new series of numbers will be generated to find the onsets, offsets, and transitions from the trajectory changes in the new contour. In this approach, the adverse effect of the incorrect F0 estimation may be reduced, especially in a real-time environment.

Moreover, the accuracy of the proposed algorithm can be improved by incorporating a function tracking significant changes in the magnitudes of each spectral channel that are also associated with the onset.

Another possible approach instead of using the stretching pitch explained in Section 6.2.2 is to scale down all F0s to one specific octave and then use a log frequency axis. This approach may help in regularizing the slopes and making them comparable.

In addition, the algorithm is based on two parameters, window size (as explained in Section 6.2.5) and the proportion of the standard deviation to calculate the thresholds, as discussed in Section 6.2.7. By evaluating the algorithm on other larger datasets such as VocalSet (Wilkins *et al.*, 2018), these parameters could be fixed to be a constant value that is generally applicable to all singers or could be determined by a formula and therefore be made adaptive to the nature of the style of input singing.

Furthermore, the algorithm's efficiency and accuracy could be evaluated on notes performed by musical instruments to see if it is also applicable in that domain.

Lastly, making the algorithm more computationally efficient requires a smaller buffer size to work faster in real-time environments.

9.4 Conclusion and future work of generating an annotated singing dataset

Chapter 7 introduced an extended set of annotations for the solo singing files in the VocalSet dataset (Wilkins *et al.*, 2018). The provided annotations include F0, onset, offset, transition, note F0, note duration, Midi pitch, and lyric. In addition, four approaches for considering the onset and offset points in a pitch contour were compared, showing that the selected points for onset and offset cannot significantly affect the note's estimated F0. Moreover, calculating a note's F0 by average or median methods does not considerably affect the note's estimated F0. The annotated dataset is available online at https://doi.org/10.5281/zenodo.7061507, accessed on 14 September 2022.

It should be noted that annotating a dataset is a very time-consuming task. Although my software tool and onset detection algorithm helped me to prepare the annotations and estimate more than 80% of the onset, offset, and transitions correctly, manually checking and adjusting the events took me around six months. This difficulty should be the main reason that the number of comprehensive annotated available datasets is deficient. To the best of my knowledge, this annotated dataset is the largest and most comprehensive annotated singing dataset available.

However, as shown in Table 2-1, several singing datasets are available that can be expanded by adding more annotations. The same approach as has been done in this study, and has been explained in section 7.1, can be applied to annotate any other available singing or instrumental datasets. In this case, a significant amount of new data will be provided to researchers to evaluate musical performances.

9.5 Conclusion and future work of calculating notes' pitch frequencies and duration according to singing technique and their positions in a piece of music study

This thesis's last and most important aim was to understand the behaviour of trained-professional singers in changing notes' pitches and duration according to the notes' positions in a piece of music and the singing technique applied. Thus, Chapter 8 modelled these behaviours based on the parameters surrounding the pitch and duration of the notes.

To achieve the goal, the variations of ten variables on 2688 solo singing recorded files, obtained from Chapter 7, were investigated to find the relationships between a note's F0 and duration with these variables. The variables considered in this study are the interval to the following and previous notes, the existence of a rest before or after the note, duration, the MIDI pitch code, and the particular singing technique applied. The Bayesian hierarchical model was used to find the effect of the variables on pitch and duration. The investigation confirms that the pitch and duration of notes are based on all these parameters in trained-professional singers' performances. In addition, Chapter 8 proposed two formulas to calculate the pitch frequencies and duration of the notes according to the variables.

To the best of my knowledge, it is the first time that singers' behaviours in changing notes' pitches and duration according to several parameters have been modelled. In other words, these models can simulate how trained-professional singers perform a piece of music. However, these models have some limitations that need to be overcome. Suggestions for improving the models are provided in the following.

Regarding future work that could be carried out in the short term, the finding of this study can be implemented by some singing synthesisers, e. g. (Goto *et al.*, 2012; Jeerapradit, Suchato and Punyabukkana, 2018), to compare how subjective human evaluation actually behaves when listening to the theoretical pitch and duration against those calculated with the models presented in Chapter 8.

In addition, the models can be extended by augmenting them with more variables. For example, longer sequences of notes can be considered instead of only the immediate prior and following notes. Moreover, the loudness of the notes can be added to the model to ascertain if it has a noticeable effect. Another possible inclusion is the variation of the starting point of the singer starting singing of a note with respect to a metronome beat or accompanying note, as discussed by Sundberg and Bauer-Huppmann (Sundberg and Bauer-Huppmann, 2007). In addition, there are several other parameters related to the musical context, such as chord note vs non-chord note, a note's position along the circle of fifths, key or underlying harmony, beat vs non-beat note's position in the bar, syncopation, chord change, change of the direction of the melody, and phrase and sub-phrase structure, could be included to ascertain the significance of their effect on the performed duration and F0. This

would require a lot of annotation work on the current dataset, with caveats as to what it could and could not provide, and then, for some variables, it would demand a completely new dataset with the appropriate annotations. This certainly would be a time-consuming task in its preparation.

Another longer-term study would be to run the model on larger datasets with more singers and music excerpts to obtain more accurate results.

In conclusion, there is still a significant gap in our understanding of music performances and how humans perceive a piece of music. To develop advanced AI tools for music, much extra research is required to formulate all the possible parameters that can affect music performers in changing pitch, duration, amplitude, and timber of the tone, as well as how a listener perceives the music according to the parameters.

Appendix

Table A-1. Comparing pitch estimators and contour smoothers algorithms by ground truth based on the mean absolute error (MAE) metric. GT = Ground Truth, ES = Estimated pitch contour, SM = Smoothed contour.

Algorithm	Specacf			Schmitt			FComb			MComb			Yin			YinFFT			Praat			PYIN (GT)		
	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM
00	256	**123**	208	68	**61**	26	123	**42**	108	46	**32**	28	588	**114**	543	61	**26**	39	14	14	0.7	0	1.2	1.2
01	256	233	112	68	63	20	123	122	56	46	47	15	588	581	291	61	60	35	14	14	0.2	0	3	3
02	256	236	128	68	64	21	123	122	59	46	47	15	588	584	316	61	60	37	14	14	0.2	0	2.7	2.7
03	256	234	118	68	62	26	123	122	62	46	47	20	588	583	302	61	58	40	14	**13**	1.2	0	5.5	5.5
04	256	236	128	68	64	21	123	122	59	46	47	15	588	584	316	61	60	37	14	14	0.2	0	2.7	2.7
05	256	232	124	68	63	23	123	122	63	46	48	17	588	582	325	61	60	40	14	14	0.3	0	3.4	3.4
06	256	235	104	68	64	18	123	122	51	46	47	13	588	582	266	61	60	32	14	14	0.2	0	2.5	2.5
07	256	237	96	68	64	16	123	122	44	46	47	11	588	584	237	61	60	28	14	14	0.1	0	2	2
08	256	233	113	68	64	20	123	122	56	46	47	15	588	582	291	61	60	35	14	14	0.2	0	2.9	2.9
09	256	244	147	68	69	30	123	137	83	46	54	24	588	743	545	61	77	59	14	14	0.5	0	5.6	5.6
10	256	226	208	68	77	72	123	138	146	46	72	64	588	628	624	61	87	100	14	43	38.2	0	44.3	44.3
11	256	227	184	68	68	55	123	131	122	46	59	46	588	626	567	61	74	81	14	21	13.4	0	23.6	23.6
12	256	227	181	68	68	52	123	131	120	46	58	44	588	648	579	61	74	79	14	21	13.1	0	21.5	21.5
13	256	227	190	68	71	59	123	133	128	46	63	52	588	613	580	61	77	87	14	21	13.9	0	28.7	28.7
14	256	226	186	68	68	56	123	132	127	46	60	48	588	660	619	61	77	84	14	22	14.3	0	24.2	24.2
15	256	227	189	68	72	61	123	132	126	46	64	53	588	580	535	61	76	84	14	25	17.5	0	31.2	31.2
16	256	223	168	68	64	44	123	123	104	46	53	37	588	580	478	61	66	67	14	18	10	0	16.9	16.9
17	256	236	128	68	64	21	123	122	59	46	47	15	588	584	316	61	60	37	14	14	0.2	0	2.7	2.7
18	256	214	195	68	69	64	123	126	132	46	62	55	588	571	569	61	74	88	14	23	15.8	0	31.1	31.1
19	256	227	190	68	71	59	123	133	128	46	63	52	588	613	580	61	77	87	14	21	13.9	0	28.7	28.7
20	256	227	190	68	71	59	123	133	128	46	63	52	588	613	580	61	77	87	14	21	13.9	0	28.7	28.7
21	264	227	189	68	66	56	128	129	125	47	58	47	591	577	519	62	72	83	14	21	13.7	0	24.5	24.5
22	256	227	190	68	71	59	123	133	128	46	63	52	588	613	580	61	77	87	14	21	13.9	0	28.7	28.7
23	256	225	136	68	62	32	123	121	78	46	50	26	588	576	381	61	62	52	14	14	0.8	0	7.3	7.3
24	256	234	116	68	64	22	123	124	60	46	48	17	588	599	322	61	62	39	14	14	0.2	0	3.2	3.2
25	256	231	137	68	65	38	123	129	87	46	54	32	588	665	483	61	73	61	14	14	1.7	0	11.9	11.9
26	256	245	102	68	66	20	123	132	57	46	50	16	588	703	380	61	72	41	14	14	0.3	0	3.6	3.6
27	256	229	133	68	64	30	123	125	76	46	51	25	588	623	410	61	67	51	14	14	1.5	0	9	9
28	256	235	122	68	65	23	123	126	65	46	49	18	588	634	373	61	66	43	14	14	0.3	0	4.5	4.5
29	256	236	114	68	65	27	123	128	67	46	52	23	588	656	386	61	71	47	14	14	1.5	0	9	9
30	256	244	105	68	66	21	123	131	59	46	50	17	588	694	380	61	72	41	14	14	0.4	0	4.4	4.4
31	261	235	153	69	62	39	126	125	88	47	51	30	600	592	411	62	61	57	14	**13**	2	0	9.4	9.4
32	256	232	97	68	**61**	23	123	121	54	46	47	18	588	580	258	61	58	35	14	**13**	1.3	0	5.7	5.7
33	256	228	96	68	63	11	123	108	33	46	45	6	588	417	201	61	48	20	14	14	0	0	0.3	0.3
34	256	226	122	68	62	20	123	113	54	46	46	13	588	510	292	61	54	35	14	14	0.1	0	1.9	1.9
35	256	234	102	68	64	9	123	107	36	46	44	5	588	410	240	61	46	19	14	14	0	0	**0**	0

Table A-2. Comparing pitch estimators and contour-smoother algorithms by ground truth based on the R-squared (R2) metric. GT = Ground Truth, ES = Estimated pitch contour, SM = Smoothed contour.

Algorithm	Specacf			Schmitt			FComb			MComb			Yin			YinFFT			Praat			PYIN (GT)		
	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM
00	−28	**−3**	0	−0.5	−0.3	0.7	−22	**−1**	0.3	−1	**0.2**	0.7	−1153	**−3**	−0.4	−22	**1**	0.7	0.8	**0.84**	1	1	0.97	0.97
01	−28	−20	0.8	−0.5	−0.2	1	−22	−17	0.8	−1	−0.9	1	−1153	−462	0.7	−22	−10	0.9	0.8	0.81	1	1	0.99	0.99
02	−28	−20	0.7	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	0.9	−1153	−516	0.6	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
03	−28	−20	0.7	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	0.9	−1153	−526	0.6	−22	−11	0.9	0.8	0.81	1	1	0.98	0.98
04	−28	−20	0.7	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	0.9	−1153	−517	0.6	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
05	−28	−19	0.7	−0.5	−0.2	0.9	−22	−16	0.8	−1	−0.8	0.9	−1153	−428	0.6	−22	−10	0.9	0.8	0.81	1	1	0.99	0.99
06	−28	−20	0.8	−0.5	−0.3	1	−22	−17	0.9	−1	−0.9	1	−1153	−501	0.7	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
07	−28	−21	0.8	−0.5	−0.3	1	−22	−18	0.9	−1	−1	1	−1153	−561	0.8	−22	−12	0.9	0.8	0.81	1	1	**1**	1
08	−28	−20	0.7	−0.5	−0.3	1	−22	−17	0.8	−1	−0.9	1	−1153	−469	0.7	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
09	−28	−20	0.6	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	0.9	−1153	−490	0.5	−22	−11	0.8	0.8	0.81	1	1	0.99	0.99
10	−28	−15	0.4	−0.5	0	0.6	−22	−10	0.5	−1	−0.4	0.7	−1153	−191	0.2	−22	−5	0.6	0.8	0.6	0.7	1	0.79	0.79
11	−28	−17	0.5	−0.5	0	0.8	−22	−13	0.6	−1	−0.5	0.8	−1153	−239	0.3	−22	−6	0.7	0.8	0.8	1	1	0.93	0.93
12	−28	−17	0.5	−0.5	0	0.8	−22	−13	0.6	−1	−0.5	0.8	−1153	−267	0.4	−22	−7	0.7	0.8	0.8	1	1	0.94	0.94
13	−28	−16	0.4	−0.5	0	0.7	−22	−12	0.6	−1	−0.5	0.8	−1153	−204	0.3	−22	−6	0.7	0.8	0.8	1	1	0.9	0.9
14	−28	−16	0.4	−0.5	0	0.8	−22	−12	0.6	−1	−0.5	0.8	−1153	−245	0.3	−22	−6	0.7	0.8	0.79	1	1	0.93	0.93
15	−28	−16	0.4	−0.5	0	0.7	−22	−12	0.5	−1	−0.5	0.7	−1153	−238	0.3	−22	−6	0.6	0.8	0.77	0.9	1	0.85	0.85
16	−28	−17	0.5	−0.5	0	0.8	−22	−13	0.6	−1	−0.5	0.8	−1153	−259	0.4	−22	−6	0.7	0.8	0.81	1	1	0.95	0.95
17	−28	−20	0.7	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	0.9	−1153	−517	0.6	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
18	−28	−15	0.4	−0.5	**0.2**	0.7	−22	−10	0.5	−1	−0.3	0.7	−1153	−155	0.3	−22	−4	0.6	0.8	0.8	0.9	1	0.89	0.89
19	−28	−16	0.4	−0.5	0	0.7	−22	−12	0.6	−1	−0.5	0.8	−1153	−204	0.3	−22	−6	0.7	0.8	0.8	1	1	0.9	0.9
20	−28	−16	0.4	−0.5	0	0.7	−22	−12	0.6	−1	−0.5	0.8	−1153	−204	0.3	−22	−6	0.7	0.8	0.8	1	1	0.9	0.9
21	−30	−17	0.4	−0.6	0.1	0.7	−24	−12	0.5	−1	−0.4	0.8	−1208	−194	0.3	−23	−5	0.7	0.8	0.8	1	1	0.92	0.92
22	−28	−16	0.4	−0.5	0	0.7	−22	−12	0.6	−1	−0.5	0.8	−1153	−204	0.3	−22	−6	0.7	0.8	0.8	1	1	0.9	0.9
23	−28	−18	0.7	−0.5	−0.1	0.9	−22	−14	0.8	−1	−0.6	0.9	−1153	−288	0.6	−22	−7	0.8	0.8	0.81	1	1	0.98	0.98
24	−28	−20	0.7	−0.5	−0.3	0.9	−22	−17	0.8	−1	−0.9	1	−1153	−457	0.7	−22	−10	0.9	0.8	0.81	1	1	0.99	0.99
25	−28	−18	0.7	−0.5	0	0.9	−22	−14	0.8	−1	−0.6	0.9	−1153	−349	0.7	−22	−8	0.8	0.8	0.81	1	1	0.98	0.98
26	−28	−21	0.8	−0.5	−0.3	1	−22	−17	0.9	−1	−0.9	1	−1153	−584	0.8	−22	−13	0.9	0.8	0.81	1	1	**1**	1
27	−28	−18	0.7	−0.5	−0.1	0.9	−22	−14	0.8	−1	−0.6	0.9	−1153	−353	0.6	−22	−9	0.9	0.8	0.81	1	1	0.98	0.98
28	−28	−19	0.7	−0.5	−0.2	0.9	−22	−16	0.8	−1	−0.8	0.9	−1153	−449	0.7	−22	−10	0.9	0.8	0.81	1	1	0.99	0.99
29	−28	−19	0.8	−0.5	−0.1	0.9	−22	−15	0.9	−1	−0.7	0.9	−1153	−468	0.8	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
30	−28	−21	0.8	−0.5	−0.3	1	−22	−17	0.9	−1	−0.9	1	−1153	−563	0.8	−22	−13	0.9	0.8	0.81	1	1	0.99	0.99
31	−31	−22	0.6	−0.7	−0.3	0.8	−25	−18	0.7	−2	−1	0.8	−1308	−478	0.5	−25	−11	0.8	0.8	0.81	1	1	0.96	0.96
32	−28	−20	0.8	−0.5	−0.2	0.9	−22	−17	0.9	−1	−0.8	0.9	−1153	−501	0.8	−22	−11	0.9	0.8	0.81	1	1	0.99	0.99
33	−28	−23	0.6	−0.5	−0.4	1	−22	−19	0.8	−1	−1.1	1	−1153	−266	0.4	−22	−12	0.8	0.8	0.81	1	1	**1**	1
34	−28	−20	0.6	−0.5	−0.3	0.9	−22	−17	0.7	−1	−0.9	0.9	−1153	−389	0.5	−22	−9	0.8	0.8	0.81	1	1	0.99	0.99
35	−28	−22	0.5	−0.5	−0.4	0.9	−22	−20	0.7	−1	−1.1	0.9	−1153	−376	0.4	−22	−11	0.8	0.8	0.81	1	1	**1**	1

Table A-3. Comparing pitch estimators and contour-smoother algorithms by ground truth based on the Root-Mean-Square Error (RMSE) metric. GT = Ground Truth, ES = Estimated pitch contour, SM = Smoothed contour.

Algorithm	Specacf			Schmitt			FComb			MComb			Yin			YinFFT			Praat			PYIN (GT)		
	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM
00	394	**161**	370	111	96	73	258	**79**	240	96	**66**	62	2086	**153**	2077	194	**58**	159	21	21	3.1	0	4.5	4.5
01	394	307	188	111	97	36	258	206	112	96	84	32	2086	1342	1210	194	137	102	21	21	1.1	0	9.5	9.5
02	394	315	220	111	99	40	258	211	127	96	86	36	2086	1417	1405	194	143	117	21	21	1.1	0	10.2	10.2
03	394	315	201	111	96	48	258	210	131	96	84	43	2086	1427	1306	194	141	115	21	20	2.4	0	14.8	14.8
04	394	315	220	111	99	40	258	211	127	96	86	36	2086	1417	1405	194	143	117	21	21	1.1	0	10.2	10.2
05	394	302	207	111	96	40	258	202	124	96	84	36	2086	1291	1332	194	134	113	21	21	1.2	0	10.7	10.7
06	394	312	176	111	99	33	258	210	104	96	85	30	2086	1396	1130	194	142	95	21	21	1	0	8.6	8.6
07	394	321	165	111	101	30	258	216	95	96	87	27	2086	1475	1054	194	148	88	21	21	0.9	0	7.7	7.7
08	394	308	190	111	98	36	258	206	113	96	85	32	2086	1351	1221	194	138	103	21	21	1.1	0	9.5	9.5
09	394	311	228	111	101	47	258	211	141	96	87	42	2086	1373	1484	194	141	127	21	21	1.3	0	12.4	12.4
10	394	256	300	111	96	97	258	167	218	96	91	90	2086	818	1839	194	117	185	21	53	46.8	0	60.6	60.6
11	394	265	277	111	88	77	258	172	192	96	80	70	2086	936	1768	194	110	165	21	27	17.5	0	34.8	34.8
12	394	266	274	111	88	75	258	173	190	96	79	67	2086	990	1741	194	113	161	21	27	17.1	0	31.5	31.5
13	394	263	281	111	90	81	258	172	198	96	82	74	2086	851	1816	194	109	171	21	27	17.9	0	40.6	40.6
14	394	260	280	111	87	79	258	167	196	96	79	70	2086	940	1770	194	110	165	21	28	18.6	0	34.3	34.3
15	394	266	285	111	95	87	258	176	202	96	88	81	2086	906	1800	194	119	175	21	33	23.8	0	49.1	49.1
16	394	266	263	111	87	67	258	172	176	96	77	60	2086	964	1685	194	110	152	21	24	13.4	0	28	28
17	394	315	220	111	99	40	258	211	127	96	86	36	2086	1417	1405	194	143	117	21	21	1.1	0	10.2	10.2
18	394	243	288	111	86	86	258	155	205	96	80	79	2086	772	1836	194	102	175	21	29	20.5	0	44.2	44.2
19	394	263	281	111	90	81	258	172	198	96	82	74	2086	851	1816	194	109	171	21	27	17.9	0	40.6	40.6
20	394	263	281	111	90	81	258	172	198	96	82	74	2086	851	1816	194	109	171	21	27	17.9	0	40.6	40.6
21	403	259	283	110	**84**	78	265	164	200	96	78	72	2038	823	1739	199	105	172	21	27	17.8	0	36.7	36.7
22	394	263	281	111	90	81	258	172	198	96	82	74	2086	851	1816	194	109	171	21	27	17.9	0	40.6	40.6
23	394	279	216	111	89	49	258	183	138	96	78	44	2086	1071	1404	194	117	123	21	21	1.9	0	14.6	14.6
24	394	306	192	111	98	38	258	206	116	96	85	34	2086	1331	1235	194	137	105	21	21	1.2	0	9.9	9.9
25	394	284	203	111	88	53	258	183	137	96	78	48	2086	1199	1278	194	126	116	21	21	2.9	0	18.5	18.5
26	394	322	153	111	100	31	258	215	94	96	87	27	2086	1511	954	194	150	81	21	21	1	0	8	8
27	394	287	207	111	92	45	258	190	129	96	81	41	2086	1197	1334	194	127	116	21	21	2.7	0	15.5	15.5
28	394	304	197	111	97	39	258	204	119	96	84	35	2086	1332	1268	194	137	107	21	21	1.2	0	10.5	10.5
29	394	303	169	111	93	39	258	198	108	96	82	36	2086	1376	1052	194	140	93	21	21	2.5	0	14.3	14.3
30	394	319	158	111	99	32	258	212	96	96	86	28	2086	1495	978	194	149	84	21	21	1.1	0	8.7	8.7
31	397	303	247	112	93	65	261	200	168	97	82	59	2109	1294	1613	196	130	147	21	**19**	3.9	0	22	22
32	394	311	162	111	93	40	258	205	106	96	81	36	2086	1400	1044	194	138	93	21	20	2.5	0	13.4	13.4
33	394	319	250	111	103	33	258	219	127	96	90	26	2086	790	1591	194	124	123	21	21	0.6	0	1.5	1.5
34	394	305	249	111	97	46	258	206	144	96	86	39	2086	1143	1566	194	132	131	21	21	0.9	0	9.1	9.1
35	394	317	277	111	105	43	258	220	142	96	90	28	2086	729	1779	194	117	123	21	21	0.6	0	**0**	0

Table A-4. Comparing pitch estimators and contour-smoother algorithms by ground truth based on the F0 Frame Error (FFE) metric. GT = Ground Truth, ES = Estimated pitch contour, SM = Smoothed contour.

Algorithm	Specacf			Schmitt			FComb			MComb			Yin			YinFFT			Praat			PYIN (GT)		
	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM	GT-ES	GT-SM	ES-SM
00	40	**48**	61	67	**69**	89	77	**82**	84	84	**87**	92	45	**52**	63	88	**90**	97	95	95.1	99.8	100	99.8	99.8
01	40	33	60	67	64	86	77	63	72	84	77	86	45	36	77	88	81	85	95	95.1	100	100	95.7	95.7
02	40	35	61	67	66	90	77	66	77	84	79	91	45	39	83	88	84	91	95	95.1	100	100	98.3	98.3
03	40	36	64	67	67	87	77	67	78	84	80	91	45	40	83	88	84	91	95	95.1	100	100	98.3	98.3
04	40	35	61	67	66	90	77	66	77	84	79	91	45	39	83	88	84	91	95	95.1	100	100	98.3	98.3
05	40	34	61	67	65	88	77	64	74	84	78	88	45	37	79	88	82	88	95	95.1	100	100	97.4	97.4
06	40	35	65	67	65	89	77	65	76	84	79	89	45	38	81	88	82	88	95	95.1	100	100	97.4	97.4
07	40	36	69	67	66	92	77	66	80	84	79	92	45	39	85	88	84	91	95	95.1	100	100	98.3	98.3
08	40	34	63	67	65	89	77	65	75	84	78	89	45	38	80	88	82	88	95	95.1	100	100	97.4	97.4
09	40	20	41	67	52	70	77	47	54	84	65	73	45	15	43	88	64	68	95	95.1	100	100	84.6	84.6
10	40	21	32	67	46	51	77	41	42	84	54	58	45	13	33	88	55	57	95	83.3	86.9	100	72	72
11	40	21	36	67	49	58	77	44	46	84	60	65	45	17	44	88	61	63	95	95.2	99.6	100	80.1	80.1
12	40	21	36	67	49	59	77	44	47	84	61	66	45	17	42	88	62	64	95	95.2	99.6	100	81.3	81.3
13	40	21	35	67	47	56	77	42	44	84	57	62	45	16	41	88	58	60	95	95.2	99.6	100	76.8	76.8
14	40	20	35	67	48	58	77	43	45	84	60	65	45	13	35	88	60	62	95	95.2	99.5	100	80.6	80.6
15	40	27	42	67	55	64	77	51	54	84	65	70	45	26	56	88	68	70	95	94.5	98.8	100	83.2	83.2
16	40	27	44	67	56	68	77	52	55	84	67	73	45	26	58	88	70	72	95	95.2	99.7	100	86	86
17	40	35	61	67	66	90	77	66	77	84	79	91	45	39	83	88	84	91	95	95.1	100	100	98.3	98.3
18	40	22	34	67	47	54	77	43	44	84	56	61	45	17	41	88	58	60	95	**95.3**	99.4	100	75.6	75.6
19	40	21	35	67	47	56	77	42	44	84	57	62	45	16	41	88	58	60	95	95.2	99.6	100	76.8	76.8
20	40	21	35	67	47	56	77	42	44	84	57	62	45	16	41	88	58	60	95	95.2	99.6	100	76.8	76.8
21	38	21	36	66	50	58	76	45	47	83	60	65	43	18	47	88	62	64	95	**95.3**	99.6	100	79.8	79.8
22	40	21	35	67	47	56	77	42	44	84	57	62	45	16	41	88	58	60	95	95.2	99.6	100	76.8	76.8
23	40	22	45	67	52	70	77	50	56	84	64	72	45	22	59	88	67	71	95	95.1	100	100	85.1	85.1
24	40	23	49	67	53	74	77	53	61	84	67	75	45	25	64	88	70	74	95	95.1	100	100	86.7	86.7
25	40	21	43	67	51	67	77	47	53	84	63	70	45	16	43	88	63	66	95	95.1	99.9	100	83.3	83.3
26	40	21	51	67	53	76	77	51	61	84	66	76	45	19	54	88	66	70	95	95.1	100	100	84.8	84.8
27	40	22	45	67	52	71	77	50	57	84	65	74	45	21	56	88	67	71	95	95.1	99.9	100	84.7	84.7
28	40	22	47	67	53	73	77	51	59	84	66	74	45	22	59	88	68	72	95	95.1	100	100	84.6	84.6
29	40	21	49	67	52	74	77	50	59	84	66	74	45	19	52	88	66	70	95	95.1	99.9	100	84.8	84.8
30	40	21	50	67	53	75	77	50	60	84	66	76	45	19	53	88	66	71	95	95.1	100	100	84.9	84.9
31	39	34	57	66	66	80	76	64	72	83	78	87	44	37	76	88	82	87	95	95.2	99.9	100	97.2	97.2
32	40	30	60	67	60	80	77	61	70	84	74	82	45	32	72	88	77	82	95	95.1	100	100	92.8	92.8
33	40	42	80	67	**69**	95	77	79	94	84	84	98	45	45	95	88	89	99	95	95.1	100	100	**100**	100
34	40	34	61	67	63	81	77	68	76	84	77	85	45	36	78	88	81	85	95	95.1	100	100	94.7	94.7
35	40	42	82	67	**69**	96	77	80	94	84	84	98	45	45	95	88	89	99	95	95.1	100	100	**100**	100

References

Abeßer, J. *et al.* (2013) 'Automatic quality assessment of vocal and instrumental performances of ninth-grade and tenth-grade pupils', *Proceedings of the {10th International Symposium on Computer Music Multidisciplinary Research, CMMR} 2013. {Sound, Music & Motion}, 15-18th {October} 2013, {Marseilles}, {France}*, (January), pp. 975–988.

Abouzid, H. *et al.* (2019) 'Signal speech reconstruction and noise removal using convolutional denoising audioencoders with neural deep learning', *Analog Integrated Circuits and Signal Processing*. Springer US, 100(3), pp. 501–512. doi: 10.1007/s10470-019-01446-6.

Al-Hussaini, I. *et al.* (2018) 'Predictive Real-Time Beat Tracking from Music for Embedded Application', in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, pp. 297–300. doi: 10.1109/MIPR.2018.00068.

Alldahl, P.-G. (2006) *Choral Intonation*, *Gehrmans Musikforlag*. Available at: http://journals.sagepub.com/doi/10.2307/3383619.

Anand, S. *et al.* (2012) 'Acoustic parameters critical for an appropriate vibrato', *Journal of Voice*. Elsevier Ltd, 26(6), pp. 820.e19-820.e25. doi: 10.1016/j.jvoice.2012.06.004.

Aubio (no date) *Aubio*. Available at: https://aubio.org/.

Baracskay, I. *et al.* (2022) 'The Diversity of Music Recommender Systems', in *27th International Conference on Intelligent User Interfaces*. New York, NY, USA: ACM, pp. 97–100. doi: 10.1145/3490100.3516474.

Barupal, D. K. and Fiehn, O. (2011) 'Scikit-learn: Machine learning in Python', *The Journal of machine Learning research*, 2, pp. 2825–2830.

Bello Correa, J. P. (2003) *Towards the automated analysis of simple polyphonic music: A knowledge-based approach*. Queen Mary University of London.

Bello, J. P. *et al.* (2004) 'On the use of phase and energy for musical onset detection in the complex domain', *IEEE Signal Processing Letters*, 11(6), pp. 553–556. doi: 10.1109/LSP.2004.827951.

Bello, J. P. *et al.* (2005) 'A tutorial on onset detection in music signals', *IEEE Transactions on Speech and Audio Processing*, 13(5), pp. 1035–1047. doi: 10.1109/TSA.2005.851998.

Benetos, E. *et al.* (2019) 'Automatic Music Transcription: An Overview', *IEEE Signal Processing Magazine*. IEEE, 36(1), pp. 20–30. doi: 10.1109/MSP.2018.2869928.

Berger, J., Coifman, R. R. and Goldberg, M. J. (1994) 'Removing noise from music using local trigonometric bases and wavelet packets', *AES: Journal of the Audio Engineering Society*, 42(10), pp. 808–818.

Besouw, R. M. Van, Brereton, J. S. and Howard, D. M. (2008) 'Range of Tuning for Tones With and Without Vibrato', *Music Perception: An Interdisciplinary Journal*, 26(2), pp. 145–155. doi: 10.1525/mp.2008.26.2.145.

Bhalke, D. G., Rao, C. B. R. and Bormane, D. S. (2016) 'Automatic musical instrument classification using fractional fourier transform based- MFCC features and counter propagation neural network', *Journal of Intelligent Information Systems*. Journal of Intelligent Information Systems, 46(3), pp. 425–446. doi: 10.1007/s10844-015-0360-9.

Bharti, H., Singh, D. and Malik, M. (2022) 'E-Learning platform for music instruments: Advantages and challenges', *International journal of health sciences*, 6(March), pp. 5890–5898. doi: 10.53730/ijhs.v6nS1.6211.

Bittner, R. *et al.* (2014) 'MedleyDB: A multitrack dataset for annotation-intensive MIR research', in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pp. 155–160.

Bittner, R. M. *et al.* (2016) 'Medleydb 2.0 : New Data and a System for Sustainable Data Collection', in *International Conference on Music Information Retrieval (ISMIR-16)*, pp. 2–4.

Bittner, R. M. *et al.* (2021) 'vocadito: A dataset of solo vocals with F0, note, and lyric annotations', in *International Society for Music Information Retrieval*. Available at: http://arxiv.org/abs/2110.05580.

Bittner, R. M., Wang, A. and Bello, J. P. (2017) 'Pitch contour tracking in music using Harmonic Locked Loops', in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 191–195. doi: 10.1109/ICASSP.2017.7952144.

Bjørklund, A. (1961) 'Analyses of Soprano Voices', *The Journal of the Acoustical Society of America*, 33(5), pp. 575–582. doi: 10.1121/1.1908728.

Böck, S. *et al.* (2012) 'Online real-time onset detection with recurrent neural networks', *15th International Conference on Digital Audio Effects, DAFx 2012 Proceedings*, pp. 15–18.

Böck, S. *et al.* (2016) 'madmom: a new Python Audio and Music Signal Processing Library', in *Proceedings of the 24th ACM International Conference on Multimedia*. Amsterdam, The Netherlands, pp. 1174–1178. doi: 10.1145/2964284.2973795.

Boem, A. and Iwata, H. (2018) 'Encounter-Type Haptic Interfaces for Virtual Reality Musical Instruments', in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, pp. 1–2. doi: 10.1109/VR.2018.8446549.

Boersma, P. (1993) 'Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-To-Noise Ratio of a Sampled Sound', in *Proceedings of the institute of phonetic sciences*. Amsterdam: University of Amsterdam, pp. 97–110. Available at: http://www.cs.northwestern.edu/~pardo/courses/casa2009/casa_papers_2009/papers/pitch tracking - boersma.pdf.

Boersma, P. and van Heuven, V. (2001) 'PRAAT, a system for doing phonetics by computer', *Glot International*, 5(9–10), pp. 341–347.

Bogdanov, D. *et al*. (2013a) 'Essentia: An audio analysis library for music information retrieval', in *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pp. 493–498.

Bogdanov, D. *et al.* (2013b) 'Essentia: An audio analysis library for music information retrieval', *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, (November), pp. 493–498.

Böhler, J. and Zölzer, U. (2016) 'Monophonic pitch detection by evaluation of individually parameterized phase locked loops', in *Proceedings of the International Conference on Digital Audio Effects, DAFx*, pp. 247–253.

Bottalico, P., Graetzer, S. and Hunter, E. J. (2017) 'Effect of Training and Level of External Auditory Feedback on the Singing Voice: Pitch Inaccuracy', *Journal of Voice*. Elsevier Inc., 31(1), pp. 122.e9-122.e16. doi: 10.1016/j.jvoice.2016.01.012.

Bozkurt, B., Baysal, O. and Yüret, D. (2017) 'A Dataset and Baseline System for Singing Voice Assessment', in *nternational Symposium on Computer Music Multidisciplinary Research (CMMR)*. Matosinhos, Portugal, pp. 430–438. Available at: http://cmmr2017.inesctec.pt/wp-content/uploads/2017/09/43_CMMR_2017_paper_31.pdf.

Braun, S. (2001) 'WINDOWS', in *Encyclopedia of Vibration*. Elsevier, pp. 1587–1595. doi: 10.1006/rwvb.2001.0052.

Brooks, S. P. and Gelman, A. (1998) 'General Methods for Monitoring Convergence of Iterative Simulations', *Journal of Computational and Graphical Statistics*. Taylor & Francis, 7(4), pp. 434–455. doi: 10.1080/10618600.1998.10474787.

Brossier, P. *et al.* (2019) 'aubio/aubio: 0.4.9'. Zenodo, https://doi.org/10.5281/zenodo.2578765. doi: 10.5281/zenodo.2578765.

Brossier, P. M. (2005) 'Fast Onset Detection Using Aubio (Brossier), Mirex 2005', p. 2005.

Brossier, P. M. (2006) *Automatic Annotation of Musical Audio for Interactive Applications*. Queen Mary, University of London. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.169.4504&rep=rep1&type=pdf.

Brunkan, M. C. and Bowers, J. (2021) 'Singing with Gesture: Acoustic and Perceptual Measures of Solo Singers', *Journal of Voice*. Elsevier Inc., 35(2), pp. 325.e17-325.e22. doi: 10.1016/j.jvoice.2019.08.029.

Buitinck, L. *et al.* (2013) 'API design for machine learning software: experiences from the scikit-learn project', pp. 1–15. Available at: http://arxiv.org/abs/1309.0238.

Cannam, C., Landone, C. and Sandler, M. (2010) 'Sonic visualiser', in *Proceedings of the international conference on Multimedia - MM '10*. New York, New York, USA: ACM Press, p. 1467. doi: 10.1145/1873951.1874248.

Cano, E. *et al.* (2019) 'Musical Source Separation: An Introduction', *IEEE Signal Processing Magazine*, 36(1), pp. 31–40. doi: 10.1109/MSP.2018.2874719.

Cano, E., Dittmar, C. and Grollmisch, S. (2011) 'Songs2See : Learn to Play by Playing', in *AES 41st International Conference on Audio for games*, pp. 2–7.

Chan, T.-S. *et al.* (2015) 'Vocal activity informed singing voice separation with the iKala dataset', in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 718–722. doi: 10.1109/ICASSP.2015.7178063.

Chang, S. and Lee, K. (2014) 'A pairwise approach to simultaneous onset/offset detection for singing voice using correntropy', in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 629–633. doi: 10.1109/ICASSP.2014.6853672.

Chao-Ling Hsu and Jang, J.-S. R. (2010) 'On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset', *IEEE Transactions on Audio, Speech, and Language Processing*, 18(2), pp. 310–319. doi: 10.1109/TASL.2009.2026503.

Chatterjee, I. *et al.* (2018) *Pitch Tracking and Pitch Smoothing Methods-Based Statistical Approach to Explore Singers' Melody of Voice on a Set of Songs of Tagore*, *Lecture Notes in Electrical Engineering*. Springer Singapore. doi: 10.1007/978-981-10-7901-6_56.

Cheng, M. *et al.* (1976) 'Some comparisons among several pitch detection algorithms', in *ICASSP '76. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Institute of Electrical and Electronics Engineers, pp. 332–335. doi: 10.1109/ICASSP.1976.1170114.

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

Choi, S. *et al.* (2020) 'Children's Song Dataset for Singing Voice Research Soonbeom', in *International Society for Music Information Retrieval Conference (ISMIR)*.

Chuang, Y. C. and Su, L. (2020) 'Beat and Downbeat Tracking of Symbolic Music Data Using Deep Recurrent Neural Networks', *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2020 - Proceedings*, pp. 346–352.

Cleveland, W. S. (1979) 'Robust Locally Weighted Regression and Smoothing Scatterplots', *Journal of the American Statistical Association*, 74(368), pp. 829–836. doi: 10.1080/01621459.1979.10481038.

Cleveland, W. S. (1981) 'LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression', *The American Statistician*, 35(1), p. 54. doi: 10.2307/2683591.

Colin Cameron, A. and Windmeijer, F. A. G. (1997) 'An R-squared measure of goodness of fit for some common nonlinear regression models', *Journal of Econometrics*, 77(2), pp. 329–342. doi: 10.1016/S0304-4076(96)01818-0.

Collins, N. (2005a) 'A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions', *Audio Engineering Society - 118th Convention Spring Preprints 2005*. Audio Engineering Society, 1, pp. 34–45.

Collins, N. (2005b) 'A comparison of sound onset detection algorithms with emphasis on psychoacoustically motivated detection functions', in *Audio Engineering Society Convention 118*. Audio Engineering Society.

Cont, A. (2010) 'A Coupled Duration-Focused Architecture for Real-Time Music-to-Score Alignment', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6), pp. 974–987. doi: 10.1109/TPAMI.2009.106.

Craven, P. and Wahba, G. (1978) 'Smoothing noisy data with spline functions', *Numerische Mathematik*, 31(4), pp. 377–403. doi: 10.1007/BF01404567.

Creech, A. (ed.) (2020) 'Proceedings of the International Society for Music Education 34 th World Conference on Music Education', in.

Cuesta, H. *et al.* (2018) 'Analysis of Intonation in Unison Choir Singing', in *15th International Conference on Music Perception and Cognition (ICMPC)*. doi: https://doi.org/10.5281/zenodo.1286570.

D'Alessandro, C. and Castellengo, M. (1991) 'Etude, par la synthese, de la perception du vibrato vocal dans la transition de notes', in *International Voice Conference*. Besancon, pp. 551–564.

Dagum, E. B. (2010) 'Time series modeling and decomposition', *Statistica*, 70, pp. 433–457. doi: 10.6092/issn.1973-2201/3597.

Dai, W. *et al.* (2017) 'A nonlinear generalization of the Savitzky-Golay filter and the quantitative analysis of saccades', *Journal of Vision*, 17(9), p. 10. doi: 10.1167/17.9.10.

Dalla Bella, S. *et al.* (2007) 'Singing proficiency in the general population', *The Journal of the Acoustical Society of America*, 121(2), pp. 1182–1189. doi: 10.1121/1.2427111.

Das, O., Smith, J. O. and Chafe, C. (2017) 'Real-time pitch tracking in audio signals with the extended complex kalman filter', in *International Conference on Digital Audio Effects (DAFx-17)*. Edinburgh,, UK, pp. 118–124. Available at: http://www.dafx17.eca.ed.ac.uk/papers/DAFx17_paper_21.pdf.

Das, O., Smith, J. O. and Chafe, C. (2020) 'Improved Real-Time Monophonic Pitch Tracking with the Extended Complex Kalman Filter', *Journal of the Audio Engineering Society*, 68(1/2),

pp. 78–86. doi: 10.17743/jaes.2019.0053.

Défossez, A. *et al.* (2019) 'Music Source Separation in the Waveform Domain'. Available at: http://arxiv.org/abs/1911.13254.

Degara, N. *et al.* (2011) 'Onset Event Decoding Exploiting the Rhythmic Structure of Polyphonic Music', *IEEE Journal of Selected Topics in Signal Processing*, 5(6), pp. 1228–1239. doi: 10.1109/JSTSP.2011.2146229.

Deng, G. and Cahill, L. W. (1994) 'Adaptive Gaussian filter for noise reduction and edge detection', in *IEEE Nuclear Science Symposium & Medical Imaging Conference*. San Francisco, CA, USA: IEEE, pp. 1615–1619. doi: 10.1109/nssmic.1993.373563.

Devaney, J. *et al.* (2011) 'Automatically extracting performance data from recordings of trained singers.', *Psychomusicology: Music, Mind and Brain*, 21(1–2), pp. 108–136. doi: 10.1037/h0094008.

Dixon, S. (2006) 'Onset detection revisited', in *Proceedings of the 9th International Conference on Digital Audio Effects*. Citeseer, pp. 133–137.

Dixon, S. and Effects, D. A. (2005) 'LIVE TRACKING OF MUSICAL PERFORMANCES USING ON-LINE TIME WARPING Simon Dixon Austrian Research Institute for Artificial Intelligence', *DAFx*, pp. 1–6.

Dobson, A. J. and Barnett, A. G. (2018) *An Introduction to Generalized Linear Models, Fourth Edition*. Chapman and Hall/CRC. doi: 10.1201/9781315182780.

Dong, M. *et al.* (2010) 'Aligning singing voice with MIDI melody using synthesized audio signal', in *7th International Symposium on Chinese Spoken Language Processing*. Tainan: IEEE, pp. 95–98. doi: 10.1109/ISCSLP.2010.5684843.

Dorfer, M., Arzt, A. and Widmer, G. (2017) 'Learning Audio - Sheet Music Correspondences for Score Identification and Offline Alignment', (2). Available at: http://arxiv.org/abs/1707.09887.

Dressler, K. (2016) 'Automatic Transcription of the Melody from Polyphonic Music', 24(5), pp. 901–913.

Drugman, T. *et al.* (2018) 'Traditional Machine Learning for Pitch Detection', *IEEE Signal Processing Letters*. IEEE, 25(11), pp. 1745–1749. doi: 10.1109/LSP.2018.2874155.

Drugman, T. and Alwan, A. (2011) 'Joint robust voicing detection and pitch estimation based on residual harmonics', *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, (January), pp. 1973–1976. doi: 10.1111/j.1096-3642.2009.00621.x.

Dzhambazov, G. *et al.* (2017) 'Metrical-accent Aware Vocal Onset Detection in Polyphonic Audio', in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, pp. 702–708. Available at: http://arxiv.org/abs/1707.06163.

Ewert, S., Muller, M. and Grosche, P. (2009) 'High resolution audio synchronization using chroma onset features', in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 1869–1872. doi: 10.1109/ICASSP.2009.4959972.

Eyben, F. *et al.* (2010) 'Universal onset detection with bidirectional long-short term memory neural networks', in *Proc. 11th Intern. Soc. for Music Information Retrieval Conference, ISMIR, Utrecht, The Netherlands*, pp. 589–594.

Faghih, B. and Timoney, J. (2019a) 'An investigation into several pitch detection algorithms for singing phrases analysis', in *2019 30th Irish Signals and Systems Conference (ISSC)*. Maynooth, Ireland: IEEE, pp. 1–5. doi: 10.1109/ISSC.2019.8904943.

Faghih, B. and Timoney, J. (2019b) 'Considerations for the Next Generation of Singing Tutor Systems', in *Audio Engineering Society Convention 146*. Dublin: Audio Engineering Society. Available at: http://www.aes.org/e-lib/browse.cfm?elib=20364.

Faghih, B. and Timoney, J. (2022a) 'Real-time monophonic singing pitch detection', *Preprint*. doi: 10.13140/RG.2.2.22054.19526.

Faghih, B. and Timoney, J. (2022b) 'Smart-Median: A New Real-Time Algorithm for Smoothing Singing Pitch Contours', *Applied Sciences*, 12(14), p. 7026. doi: 10.3390/app12147026.

Ferro, M. and Tamburini, F. (2019) 'Using Deep Neural Networks for Smoothing Pitch Profiles in Connected Speech', *Italian Journal of Computational Linguistics*, 5(2), pp. 33–48. doi: 10.4000/ijcol.476.

'Finale' (no date). Available at: https://www.klemm-music.de/makemusic/finale/.

Gawlik, M. and Wszołek, W. (2018) 'Modern pitch detection methods in singing voices analyzes', in *Euronoise 2018*, pp. 247–254. Available at: http://www.euronoise2018.eu/docs/papers/42_Euronoise2018.pdf.

Gelman, A. *et al.* (1992) 'Inference from Iterative Simulation Using Multiple Sequences', *Statistical Science*, 7(4), pp. 457–472.

Gelman, A. *et al.* (1995) *Bayesian Data Analysis*. Chapman and Hall/CRC. doi: 10.1201/9780429258411.

Gómez, E. and Bonada, J. (2013) 'Towards Computer-Assisted Flamenco Transcription: An Experimental Comparison of Automatic Transcription Algorithms as Applied to A Cappella Singing', *Computer Music Journal*, 37(2), pp. 73–90. doi: 10.1162/COMJ_a_00180.

Gong, R. and Serra, X. (2018) 'Towards an efficient deep learning model for musical onset detection'. doi: arXiv:1806.06773v1.

Gonzalez, S. and Brookes, M. (2014) 'PEFAC - A Pitch Estimation Algorithm Robust to High Levels of Noise', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 22(2), pp. 518–530. doi: 10.1109/TASLP.2013.2295918.

Goto, M. (2001) 'An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds', *Journal of New Music Research*, 30(2), pp. 159–171. doi: 10.1076/jnmr.30.2.159.7114.

Goto, M. *et al.* (2012) 'VocaListener and VocaWatcher: Imitating a human singer by using signal processing', in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5393–5396. doi: 10.1109/ICASSP.2012.6289140.

Gruenz, O. O. and Schott, L. O. (1949) 'Extraction and Portrayal of Pitch of Speech Sounds', *The Journal of the Acoustical Society of America*, 21(5), pp. 487–495. doi: 10.1121/1.1906538.

Gupta, C., Li, H. and Wang, Y. (2017) 'Perceptual evaluation of singing quality', in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, pp. 577–586. doi: 10.1109/APSIPA.2017.8282110.

Gupta, C., Li, H. and Wang, Y. (2018) 'A technical framework for automatic perceptual evaluation of singing quality', *APSIPA Transactions on Signal and Information Processing*, 7. doi: 10.1017/ATSIP.2018.10.

Henkel, F. and Widmer, G. (2021) 'Real-Time Music Following in Score Sheet Images via Multi-Resolution Prediction', *Frontiers in Computer Science*, 3(November), pp. 1–13. doi: 10.3389/fcomp.2021.718340.

Hennequin, R. *et al.* (2020) 'Spleeter: a fast and efficient music source separation tool with pre-trained models', *Journal of Open Source Software*, 5(50), p. 2154. doi: 10.21105/joss.02154.

Henry, M. (2015) 'Vocal Sight-Reading Assessment', *Update: Applications of Research in Music Education*, 33(2), pp. 58–64. doi: 10.1177/8755123314547908.

Henry, M. L. (2011) 'The effect of pitch and rhythm difficulty on vocal sight-reading performance', *Journal of Research in Music Education*, 59(1), pp. 72–84. doi: 10.1177/0022429410397199.

Heylen, L. *et al.* (2002) 'Normative voice range profiles of male and female professional voice users', *Journal of Voice*, 16(1), pp. 1–7. doi: 10.1016/S0892-1997(02)00065-6.

Hoon Heo, Dooyong Sung and Kyogu Lee (2013) 'Note onset detection based on harmonic cepstrum regularity', in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, pp. 1–6. doi: 10.1109/ICME.2013.6607461.

Hoppe, D., Sadakata, M. and Desain, P. (2006) 'Development of real-time visual feedback assistance in singing training: A review', *Journal of Computer Assisted Learning*, 22(4), pp. 308–316. doi: 10.1111/j.1365-2729.2006.00178.x.

Howard, D. M. *et al.* (2004) 'WinSingad: A real-time display for the singing studio', *Logopedics Phoniatrics Vocology*, 29(3), pp. 135–144. doi: 10.1080/14015430410000728.

Hutchinson, M. F. and de Hoog, F. R. (1985) 'Smoothing noisy data with spline functions', *Numerische Mathematik*, 47(1), pp. 99–106. doi: 10.1007/BF01389878.

Ishi, C. T., Hirose, K. and Minematsu, N. (2003) 'Mora F0 representation for accent type identification in continuous speech and considerations on its relation with perceived pitch values', *Speech Communication*, 41(2–3), pp. 441–453. doi: 10.1016/S0167-6393(03)00014-1.

Jeerapradit, L., Suchato, A. and Punyabukkana, P. (2018) 'HMM-based Thai Singing Voice Synthesis System', in *2018 22nd International Computer Science and Engineering Conference (ICSEC)*. IEEE, pp. 1–4. doi: 10.1109/ICSEC.2018.8712801.

Jehan, T. (2005) *Creating music by listening (Doctoral dissertation)*, *Media Arts and Sciences*. Media Arts and Sciences Department, Massachusetts Institute of Technology.

Jitendra, M. S. N. V. and Radhika, Y. (2021) 'Singer Gender Classification using Feature-based and Spectrograms with Deep Convolutional Neural Network', *International Journal of Advanced Computer Science and Applications*, 12(2), pp. 135–144. doi: 10.14569/IJACSA.2021.0120218.

Jlassi, W., Bouzid, A. and Ellouze, N. (2016) 'A new method for pitch smoothing', in *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE, pp. 657–661. doi: 10.1109/ATSIP.2016.7523161.

Jones, W. M. P. (1995) *Kernel Smoothing*. London: Chapman & Hall.

Jouvet, D. and Laprie, Y. (2017) 'Performance analysis of several pitch detection algorithms on simulated and real noisy speech data', in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1614–1618. doi: 10.23919/EUSIPCO.2017.8081482.

Karatana, A. and Yildiz, O. (2017) 'Music genre classification with machine learning techniques', in *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE, pp. 1–4. doi: 10.1109/SIU.2017.7960694.

Kasi, K. and Zahorian, S. A. (2002) 'Yet Another Algorithm for Pitch Tracking', in *IEEE International Conference on Acoustics Speech and Signal Processing*. IEEE, pp. I-361-I–364. doi: 10.1109/ICASSP.2002.5743729.

Kawahara, H., Estill, J. and Fujimura, O. (2001) 'Aperiodicity extraction and control using mixed mode excitation and group delay manipulation for a high quality speech analysis, modification and synthesis system STRAIGHT', *International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications (MAVEBA)*, (May 2014), pp. 59–64.

Khadem-hosseini, M. *et al.* (2020) 'Error Correction in Pitch Detection Using a Deep Learning Based Classification', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 28(Ml), pp. 990–999. doi: 10.1109/TASLP.2020.2977472.

Kim, J. W. *et al.* (2018) 'Crepe: A Convolutional Representation for Pitch Estimation', *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. IEEE, 2018-April, pp. 161–165. doi: 10.1109/ICASSP.2018.8461329.

Klapuri, A. (2000) 'Qualitative and quantitative aspects in the design of periodicity estimation algorithms', in *2000 10th European Signal Processing Conference*, pp. 1–4.

Klingbeil, M. (2005) 'Software for Spectral Analysis, Editing, and Synthesis', *International Computer Music Conference*, (1).

Krige, W. A. and Niesler, T. R. (2006) 'An HMM Based Singing Transcription System', *Pattern Recognition Association of South Africa*, (May).

Kroher, N. and Díaz-Báñez, J.-M. (2019) 'Modelling melodic variation and extracting melodic templates from flamenco singing performances', *Journal of Mathematics and Music*, 13(2), pp. 150–170. doi: 10.1080/17459737.2019.1610194.

Kroher, N. and Gomez, E. (2016) 'Automatic Transcription of Flamenco Singing From Polyphonic Music Recordings', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE, 24(5), pp. 901–913. doi: 10.1109/TASLP.2016.2531284.

Lacoste, A. and Eck, D. (2006) 'A supervised classification algorithm for note onset detection', *EURASIP Journal on Advances in Signal Processing*. Springer, 2007, pp. 1–13.

Lal, P. (2006) 'A Comparison of Singing Evaluation Algorithms', *Computer*, pp. 2298–2301.

Lang, M. (2003) *TuneIt, a simple command-line instrument tuner for Linux*. Available at: https://github.com/mlang/tuneit (Accessed: 20 June 2023).

Lepain, P. (1999) 'Polyphonic Pitch Extraction from Musical Signals', *Journal of New Music Research*. Routledge, 28(4), pp. 296–309. doi: 10.1076/0929-8215(199912)28:04;1-O;FT296.

Lewis-Beck, M. S. and Skalaban, A. (1990) 'The R -Squared: Some Straight Talk', *Political Analysis*, 2, pp. 153–171. doi: 10.1093/pan/2.1.153.

Li, Y. and Li, C. (2018) 'Singer Recognition Based on Convolutional Deep Belief Networks', *IOP Conference Series: Materials Science and Engineering*, 435(1), p. 012005. doi: 10.1088/1757-899X/435/1/012005.

Lin, C. H. *et al.* (2014) 'Automatic singing evaluating system based on acoustic features and rhythm', *IEEE International Conference on Orange Technologies, ICOT 2014*, pp. 165–168. doi: 10.1109/ICOT.2014.6956625.

Lin, H., Wu, H.-H. and Kao, Y.-T. (2008) 'Geometric measures of distance between two pitch contour sequences', *Journal of Computers*, 19(2), pp. 55–66.

Lindblom, B. and Sundberg, J. (2007) 'The Human Voice in Speech and Singing', in *Springer Handbook of Acoustics*. New York, NY: Springer New York, pp. 669–712. doi:

10.1007/978-0-387-30425-0_16.

Liu, L. and Benetos, E. (2021) 'From Audio to Music Notation', in Miranda, E. R. (ed.) *Handbook of Artificial Intelligence for Music*. Cham: Springer International Publishing, pp. 693–714. doi: 10.1007/978-3-030-72116-9_24.

Liu, Q. *et al.* (2013) 'A Pitch Smoothing Method for Mandarin Tone Recognition', *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(4), pp. 245–254.

Luers, J. K. and Wenning, R. H. (1971) 'Polynomial Smoothing: Linear vs Cubic', *Technometrics*, 13(3), p. 589. doi: 10.2307/1267170.

Luo, Y.-J. *et al.* (2018) 'Singing Voice Correction Using Canonical Time Warping', in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 156–160. doi: 10.1109/ICASSP.2018.8461280.

Makov, S. *et al.* (2019) 'A spectral-based pitch detection method', in *AIP Conference Proceedings*, pp. 050005-1-050005–6. doi: 10.1063/1.5138432.

Malloch, J. *et al.* (2019) 'A Design Workbench for Interactive Music Systems', in Holland, S. et al. (eds) *New Directions in Music and Human- Computer Interaction*. Springer, pp. 23–40. doi: 10.1007/978-3-319-92069-6_2.

Manternach, J. N. (2016) 'Effects of varied conductor prep movements on singer muscle engagement and voicing behaviors', *Psychology of Music*, 44(3), pp. 574–586. doi: 10.1177/0305735615580357.

Masri, P. and Bateman, A. (1996) 'Improved modelling of attack transients in music analysis-resynthesis', *Proceedings of the International Computer Music Conference*, pp. 100–103. Available at: http://hans.fugal.net/comps/papers/masri_1996.pdf.

Mauch, M. *et al.* (2015) 'Computer-aided Melody Note Transcription Using the Tony Software : Accuracy and Efficiency', *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR 2015)*, p. 8. doi: 10.1121/1.4881915.

Mauch, M. and Dixon, S. (2014) 'PYIN: A fundamental frequency estimator using probabilistic threshold distributions', *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. IEEE, (1), pp. 659–663. doi: 10.1109/ICASSP.2014.6853678.

Mauch, M., Frieler, K. and Dixon, S. (2014) 'Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory', *The Journal of the Acoustical Society of America*, 136(1), pp. 401–411. doi: 10.1121/1.4881915.

Mayor, Oscar., Bonada, Jordi., Loscos, A. (2009) 'Performance Analysis and Scoring of the Singing Voice', *Proc 35th AES Intl Conf London UK*, pp. 1–7. doi: 10.1007/s10958-005-

0306-9.

Mayor, O., Bonada, J. and Loscos, A. (2006) 'The Singing Tutor: Expression Categorization and Segmentation of the Singing Voice', in *AES 121st Convention*. San Francisco.

McFee, B. *et al.* (2015) 'librosa: Audio and Music Signal Analysis in Python', *Proceedings of the 14th Python in Science Conference*, (Scipy), pp. 18–24. doi: 10.25080/majora-7b98e3ed-003.

McFee, B. *et al.* (2020) 'librosa/librosa: 0.8.0'. Zenodo. doi: 10.5281/zenodo.3955228.

McFee, B. *et al.* (2022) 'librosa/librosa: 0.9.1'. doi: 10.5281/ZENODO.6097378.

McKinney, M. F. *et al.* (2007) 'Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms', *Journal of New Music Research*, 36(1), pp. 1–16. doi: 10.1080/09298210701653252.

McLeod, A. *et al.* (2017) 'Automatic Transcription of Polyphonic Vocal Music', *Applied Sciences*, 7(12), p. 1285. doi: 10.3390/app7121285.

Meseguer-Brocal, G., Cohen-Hadria, A. and Peeters, G. (2018) 'DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm', in *The 19th International Society for Music Information Retrieval Conference, ISMIR 2018*. Paris, France: International Society for Music Information Retrieval, pp. 431–437. doi: 10.5281/zenodo.1492443.

Molina, E. (2012) 'Automatic Scoring of Singing Voice Based on Melodic Similarity Measures', pp. 1–48. doi: 10.1016/j.econlet.2011.10.013.

Molina, E. *et al.* (2013) 'Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment', *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 744–748. doi: 10.1109/ICASSP.2013.6637747.

Molina, E. *et al.* (2014) 'Evaluation framework for automatic singing transcription', in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pp. 567–572.

Moore, B. C. J. (2013) *An Introduction to the Psychology of Hearing*. 6th edn. Brill. doi: 10.1121/1.4898050.

Mora, J. *et al.* (2010) 'Characterization and melodic similarity of a cappella flamenco cantes', *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, pp. 351–356.

Müller-Rakow, A. and Flechtner, R. (2017) 'Designing Interactive Music Systems with and for People with Dementia', *The Design Journal*, 20(sup1), pp. S2207–S2214. doi: 10.1080/14606925.2017.1352736.

Müller, M. (2007) *Dynamic time warping*, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-74048-3_4.

Müller, M., Grosche, P. and Wiering, F. (2010) 'Automated analysis of performance variations in folk song recordings', in *International conference on Multimedia information retrieval*. New York, New York, USA: ACM Press, pp. 247–256. doi: 10.1145/1743384.1743429.

Muller, M., Kurth, F. and Röder, T. (2004) 'Towards an Efficient Algorithm for Automatic Score-to-Audio Synchronization', in *Ismir*. Barcelona, Spain. Available at: http://ismir2004.ismir.net/proceedings/p067-page-365-paper136.pdf.

Nakamura, E. *et al.* (2014) 'Outer-Product Hidden Markov Model and Polyphonic MIDI Score Following', *Journal of New Music Research*, 43(2), pp. 183–201. doi: 10.1080/09298215.2014.884145.

Nakamura, E., Yoshii, K. and Katayose, H. (2017) 'Performance Error Detection and Post-Processing for Fast and Accurate Symbolic Music Alignment', in *18th International Society for Music Information Retrieval Conference*. Suzhou: ISMIR, pp. 347–353.

Nakamura, T., Nakamura, E. and Sagayama, S. (2013) 'Acoustic Score Following To Musical Performance With Errors and Arbitrary Repeats and Skips', *Proceedings of the Sound and Music Computing Conference 2013, SMC 2013*, pp. 299–304. Available at: http://www.logos-verlag.de/cgi-bin/buch/isbn/3472.

Nakamura, T., Nakamura, E. and Sagayama, S. (2016) 'Real-Time Audio-to-Score Alignment of Music Performances Containing Errors and Arbitrary Repeats and Skips', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(2), pp. 329–339. doi: 10.1109/TASLP.2015.2507862.

Nakano, T., Goto, M. and Hiraga, Y. (2006) 'An Automatic Singing Skill Evaluation Method for Unknown Melodies Using Pitch Interval Accuracy and Vibrato Features', in *Ninth International Conference on Spoken Language Processing*. Pittsburgh, pp. 1706–1709. Available at: https://www.isca-speech.org/archive/interspeech_2006/i06_1854.html.

Nakano, T., Goto, M. and Hiraga, Y. (2007) 'MiruSinger: A singing skill visualization interface using real-time feedback and music CD recordings as referential data', *Proceedings ISM Workshops 2007 9th IEEE International Symposium on Multimedia - Workshops*, pp. 75–76. doi: 10.1109/ISMW.2007.4475948.

Nakatani, T. *et al.* (2008) 'A method for fundamental frequency estimation and voicing decision: Application to infant utterances recorded in real acoustical environments', *Speech Communication*, 50(3), pp. 203–214. doi: 10.1016/j.specom.2007.09.003.

Nisar, S., Khan, O. U. and Tariq, M. (2016) 'An Efficient Adaptive Window Size Selection Method for Improving Spectrogram Visualization', *Computational Intelligence and Neuroscience*, 2016, pp. 1–13. doi: 10.1155/2016/6172453.

Nishikimi, R. *et al.* (2019) 'Automatic Singing Transcription Based on Encoder-decoder

Recurrent Neural Networks with a Weakly-supervised Attention Mechanism', in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 161–165. doi: 10.1109/ICASSP.2019.8683024.

Nishikimi, R. *et al.* (2021) 'Audio-to-score singing transcription based on a CRNN-HSMM hybrid model', *APSIPA Transactions on Signal and Information Processing*, 10(1). doi: 10.1017/ATSIP.2021.4.

Noll, A. M. (1967) 'Cepstrum Pitch Determination', *The Journal of the Acoustical Society of America*, 41(2), pp. 293–309. doi: 10.1121/1.1910339.

de Obaldía, C. and Zölzer, U. (2019) 'Improving Monophonic Pitch Detection Using the ACF and Simple Heuristics', in *Proceedings ofthe 22nd International Conference on Digital Audio Effects (DAFx-19)*. Birmingham, pp. 1–7. Available at: http://dmtlab.bcu.ac.uk/conf-sites/dafx-2019-html/papers/DAFx2019_paper_54.pdf.

Okada, M., Ishikawa, T. and Ikegaya, Y. (2016) 'A Computationally Efficient Filter for Reducing Shot Noise in Low S/N Data', *PLOS ONE*. Edited by T. Abraham, 11(6), p. e0157595. doi: 10.1371/journal.pone.0157595.

Orfanidis, S. J. (2018) 'Local Polynomial Filters', in *Applied Optimum Signal Processing*, pp. 119–163. Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://eceweb1.rutgers.edu/~orfanidi/aosp/aosp-ch03.pdf.

Orio, N., Lemouton, S. and Schwarz, D. (2003) 'Score following: state of the art and new developments', in *New interfaces for musical expression*, pp. 36–41. Available at: http://dl.acm.org/citation.cfm?id=1085724.

Paliwal, K. and Wojcicki, K. (2008) 'Effect of Analysis Window Duration on Speech Intelligibility', *IEEE Signal Processing Letters*, 15, pp. 785–788. doi: 10.1109/LSP.2008.2005755.

Pardo, B. and Birmingham, W. (2005) 'Modeling form for on-line following of musical performances', in *The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, pp. 1018–1023. Available at: papers2://publication/uuid/B72E0187-AB89-47D4-B97D-BA9E91F8DB9E.

Pelchat, N. and Gelowitz, C. M. (2020) 'Neural Network Music Genre Classification', *Canadian Journal of Electrical and Computer Engineering*, 43(3), pp. 170–173.

Pesek, M., Leonardis, A. and Marolt, M. (2017) 'Robust real-time music transcription with a compositional hierarchical model', *PLoS ONE*, 12(1). doi: 10.1371/journal.pone.0169411.

Plante, F., Meyer, G. and Ainsworth, W. A. (1995) 'A Pitch Extraction Reference Database', in *Fourth European Conference on Speech Communication and Technology, EUROSPEECH*. Madrid.

Plummer, M. (2003) 'JAGS : A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling JAGS : Just Another Gibbs Sampler', in *3rd international workshop on distributed statistical computing*. Vienna, Austria, pp. 1–10.

Podder, P. *et al.* (2014) 'Comparative Performance Analysis of Hamming, Hanning and Blackman Window', *International Journal of Computer Applications*, 96(18), pp. 1–7. doi: 10.5120/16891-6927.

Powers, D. M. W. (2020) 'Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation', pp. 37–63. doi: https://doi.org/10.48550/arXiv.2010.16061.

Prame, E. (1997) 'Vibrato extent and intonation in professional Western lyric singing', *The Journal of the Acoustical Society of America*, 102(1), pp. 616–621. doi: 10.1121/1.419735.

R Core Team (no date) 'R: A Language and Environment for Statistical Computing'. Vienna, Austria. Available at: https://www.r-project.org.

Rabiner, L. *et al.* (1976) 'A comparative performance study of several pitch detection algorithms', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(5), pp. 399–418. doi: 10.1109/TASSP.1976.1162846.

Rabiner, L. R. and Sambur, M. R. (1975) 'An Algorithm for Determining the Endpoints of Isolated Utterances', *Bell System Technical Journal*, 54(2), pp. 297–315. doi: 10.1002/j.1538-7305.1975.tb02840.x.

Racharla, K. *et al.* (2020) 'Predominant Musical Instrument Classification based on Spectral Features', in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, pp. 617–622. doi: 10.1109/SPIN48934.2020.9071125.

Raffel, C. *et al.* (2014) 'mir_eval: A transparent implementation of common MIR metrics', in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pp. 367–372.

Raffel, C. and Ellis, D. P. W. (2014) 'Intuitive analysis, creation and manipulation of MIDI data with pretty_midi', in *15th International Society for Music Information Retrieval Conference*. Taipei, Taiwan. Available at: http://craffel.github.io/pretty-midi/.

Rafii, Z. and Pardo, B. (2012) 'Music/voice separation using the similarity matrix', in *13th International Society for Music Information Retrieval Conference, ISMIR 2012*, pp. 583–588.

Rafii, Z. and Pardo, B. (2013) 'REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation', *IEEE Transactions on Audio, Speech, and Language Processing*. IEEE, 21(1), pp. 73–84. doi: 10.1109/TASL.2012.2213249.

Reback, J. *et al.* (2020) 'pandas-dev/pandas: Pandas 1.0.3'. doi: 10.5281/ZENODO.3715232.

Rej, R. (2003) *NIST/SEMATECH e-Handbook of Statistical Methods*. American Association for Clinical Chemistry. doi: https://doi.org/10.18434/M32189.

Rosenzweig, S., Cuesta, H., *et al.* (2020) 'Dagstuhl ChoirSet: A Multitrack Dataset for MIR Research on Choral Singing', *Transactions of the International Society for Music Information Retrieval*, 3(1), pp. 98–110. doi: 10.5334/tismir.48.

Rosenzweig, S., Scherbaum, F., *et al.* (2020) 'Erkomaishvili Dataset: A Curated Corpus of Traditional Georgian Vocal Music for Computational Musicology', *Transactions of the International Society for Music Information Retrieval*, 3(1), pp. 31–41. doi: 10.5334/tismir.44.

Ryynänen, M. (2006) 'Singing transcription', *Signal Processing Methods for Music Transcription*. Edited by A. Klapuri and M. Davy. Boston, MA: Springer US, pp. 361–390. doi: 10.1007/0-387-32845-9_12.

Ryynänen, M. P. (2008) *Automatic Transcription of Pitch Content in Music and Selected Applications*, *PhD Thesis*. doi: 10.1017/CBO9781107415324.004.

Ryynänen, M. P. and Klapuri, A. (2006) 'Transcription of the singing melody in polyphonic music', *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pp. 222–227. doi: 10.1149/06001.1239ecst.

Salamon, J. and Gomez, E. (2012) 'Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics', *IEEE Transactions on Audio, Speech, and Language Processing*. IEEE, 20(6), pp. 1759–1770. doi: 10.1109/TASL.2012.2188515.

Salazar, S. *et al.* (2015) 'Continuous score-coded pitch correction'. Google Patents.

Sampaio, M. da S. (2018) 'Contour Similarity Algorithms', *MusMat - Brazilian Journal of Music and Mathematics*, 2(2), pp. 58–78. Available at: https://musmat.org/wp-content/uploads/2018/12/08-contour-similarity-algorithm.pdf.

Savitzky, A. and Golay, M. J. E. (1964) 'Smoothing and Differentiation of Data by Simplified Least Squares Procedures.', *Analytical Chemistry*, 36(8), pp. 1627–1639. doi: 10.1021/ac60214a047.

Schedl, M. (2017) 'Investigating country-specific music preferences and music recommendation algorithms with the LFM-1b dataset', *International Journal of Multimedia Information Retrieval*. Springer London, 6(1), pp. 71–84. doi: 10.1007/s13735-017-0118-y.

Schedl, M., Knees, P. and Gouyon, F. (2017) 'New Paths in Music Recommender Systems Research', in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. New York, NY, USA: ACM, pp. 392–393. doi: 10.1145/3109859.3109934.

Schindler, A., Lidy, T. and Böck, S. (2020) 'Deep Learning for MIR Tutorial'. doi: https://doi.org/10.48550/arXiv.2001.05266.

Schluter, J. and Bock, S. (2014) 'Improved musical onset detection with Convolutional Neural Networks', in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6979–6983. doi: 10.1109/ICASSP.2014.6854953.

Schlüter, J. and Böck, S. (2013) 'Musical onset detection with convolutional neural networks', in *6th international workshop on machine learning and music (MML), Prague, Czech Republic*.

Schmid, M., Rath, D. and Diebold, U. (2022) 'Why and How Savitzky–Golay Filters Should Be Replaced', *ACS Measurement Science Au*, 2(2), pp. 185–196. doi: 10.1021/acsmeasuresciau.1c00054.

Schramm, R., Jung, C. R. and Miranda, E. R. (2015) 'Dynamic Time Warping for Music Conducting Gestures Evaluation', *IEEE Transactions on Multimedia*, 17(2), pp. 243–255. doi: 10.1109/TMM.2014.2377553.

Schramm, R., Nunes, H. D. S. and Jung, C. R. (2015) 'Automatic solfège assessment', *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pp. 183–189.

Schramm, R., Nunes, H. D. S. and Jung, C. R. (2016) 'Audiovisual Tool for Solfège Assessment', *ACM Transactions on Multimedia Computing, Communications, and Applications*, 13(1), pp. 1–21. doi: 10.1145/3007194.

Schreiber, H. and Müller, M. (2018a) 'A crowdsourced experiment for tempo estimation of electronic dance music', in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 409–415.

Schreiber, H. and Müller, M. (2018b) 'A single-step approach to musical tempo estimation using a convolutional neural network', *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 98–105.

Schreiber, H., Urbano, J. and Müller, M. (2020) 'Music Tempo Estimation: Are We Done Yet?', in *Transactions of the International Society for Music Information Retrieval*, p. 111. doi: 10.5334/tismir.43.

Schroeder, M. R. (1968) 'Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement', *The Journal of the Acoustical Society of America*, 43(4), pp. 829–834. doi: 10.1121/1.1910902.

Schwarz, D., Orio, N. and Schnell, N. (2004) 'Robust Polyphonic Midi Score Following with Hidden Markov Models', *International Computer Music Conference*, pp. 1–4. Available at: https://hal.archives-ouvertes.fr/hal-01161328/%5Cnhttp://recherche.ircam.fr/anasyn/schwarz/publications/icmc2004/scofo-midi.pdf.

Seashore, C. E. (1967) *Psychology of Music*. New York: Dover. Available at: http://store.doverpublications.com/0486218511.html.

Seaton, R., Pim, D. and Sharp, D. (2013) 'Pitch Drift in A Cappella Choral Singing', in *Institute of Acoustics Annual Spring Conference*. Nottingham, pp. 358–364.

Sebastian, B., Krebs, F. and Schedl, M. (2012) 'Evaluating the online capabilities of onset detection methods', in *13th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 49–54.

Serra, X. and Smith, J. (1990) 'Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition', *Computer Music Journal*, 14(4), p. 12. doi: 10.2307/3680788.

Shetty, S. S. and R, R. S. (2014) 'Audio Noise Removal – The State of the Art', *International Journal of Computational Engineering Research (IJCER)*, 04(12), pp. 2250–3005. Available at: www.ijceronline.com.

'Sibelius' (no date). Available at: https://www.avid.com/de/sibelius.

Simpson, R. E. (1987) *Introductory Electronics for Scientists and Engineers*. Allyn and Bacon (Introductory Electronics for Scientists and Engineers). Available at: https://books.google.ie/books?id=OjMfAQAAIAAJ.

Smith, S. W. (1999) 'Moving Average Filters', in *The Scientist & Engineer's Guide to Digital Signal Processing*. Second. California Technical Publishing, pp. 277–284. Available at: https://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch15.pdf.

So, Y., Jia, J. and Cai, L. (2012) 'Analysis and Improvement of Auto-correlation Pitch Extraction Algorithm Based on Candidate Set', in *Lecture Notes in Electrical Engineering*. Berlin/Heidelberg, Germany: Springer, pp. 697–702. doi: 10.1007/978-3-642-25792-6_106.

Stables, R., Athwal, C. and Bullock, J. (2011) 'Towards a model for the humanisation of pitch drift in singing voice synthesis', in *International Computer Music Conference*, pp. 555–558. doi: 10.1021/jp076092x.

Stöter, F.-R. *et al.* (2019) 'Open-Unmix - A Reference Implementation for Music Source Separation', *Journal of Open Source Software*, 4(41), p. 1667. doi: 10.21105/joss.01667.

Sundberg, J. (1972) 'Pitch of synthetic sung vowels', *STL-QPSR*, 13(1), pp. 34–44.

Sundberg, J. (1979) 'Maximum speed of pitch changes in singers and untrained subjects', *Journal of Phonetics*. Elsevier Masson SAS, 7(2), pp. 71–79. doi: 10.1016/S0095-4470(19)31040-X.

Sundberg, J. (1992) 'Breathing behavior during singing', *STL-QPSR*, 33(1), pp. 49–64. Available at: http://www.speech.kth.se/qpsr/.

Sundberg, J. (1994) 'Acoustic and psychoacoustic aspects of vocal vibrato', *StL-QPSR*, 35(2–3), pp. 45–68.

Sundberg, J. (2011) 'Some observations on operatic singer's intonation', *Interdisciplinary Studies in Musicology*, 10, pp. 47–60.

Sundberg, J. (2013) 'Perception of Singing', in *The Psychology of Music*. Third Edit. Elsevier, pp. 69–105. doi: 10.1016/B978-0-12-381460-9.00003-1.

Sundberg, J. and Bauer-Huppmann, J. (2007) 'When Does a Sung Tone Start?', *Journal of Voice*, 21(3), pp. 285–293. doi: 10.1016/j.jvoice.2006.01.003.

Sundberg, J. and La, F. (2011) 'Is Intonation Expressive?', in *40th Annual Symposium on Care of the Professional Voice*. Philadelphia.

Sundberg, J., Lã, F. M. B. and Himonides, E. (2013) 'Intonation and Expressivity: A Single Case Study of Classical Western Singing', *Journal of Voice*. Elsevier Ltd, 27(3), pp. 391.e1-391.e8. doi: 10.1016/j.jvoice.2012.11.009.

Sundberg, J., Prame, E. and Iwarsson, J. (1995) 'Replicability and accuracy of pitch patterns in professional singers', *STL-QPSR*, 36(2–3), pp. 51–62.

Talkin, D. (1995) 'A Robust Algorithm for Pitch Tracking (RAPT)', in Kleijn, W. B. and Paliwal, K. K. (eds) *Speech Coding and Synthesis,*. Amsterdam, Netherlands: Elsevier, pp. 495–518.

Tardón, L. J. *et al.* (2018) 'Music Learning: Automatic Music Composition and Singing Voice Assessment', in *Springer Handbook of Systematic Musicology*. Springer, Berlin, Heidelberg, pp. 873–883. doi: 10.1007/978-3-662-55004-5_42.

Team, C. (COmputational analysis of Fla. music) (2013) 'TONAS: a dataset of flamenco a cappella sung melodies with corresponding manual transcriptions'. doi: 10.5281/ZENODO.1290722.

Technologies, A. A. (2022) 'Auto-tune'. Available at: https://www.antarestech.com/%0Amediafiles/documentation_records/10_%0AAuto-Tune_Live_Manual.pdf.

Toh, C. C., Zhang, B. and Wang, Y. (2008) 'Multiple-feature fusion based onset detection for solo singing voice', *ISMIR 2008 - 9th International Conference on Music Information Retrieval*, (January 2008), pp. 515–520.

Tong Zhang (2003) 'Automatic singer identification', in *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*. IEEE, pp. I–33. doi: 10.1109/ICME.2003.1220847.

Valero-Mas, J. J., Salamon, J. and Gómez, E. (2015) 'Analyzing the influence of pitch quantization and note segmentation on singing voice alignment in the context of audio-based Query-by-Humming', *Proceedings of the 12th International Conference in Sound and Music Computing, SMC 2015*, pp. 371–378. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84988521905&partnerID=40&md5=16afa9d1e013a793825160c196042898.

Villavicencio, F. *et al.* (2015) *Efficient Pitch Estimation on Natural Opera-Singing by a Spectral Correlation based Strategy*, *IPSJ SIG Technical Report*. doi: 10.13140/RG.2.1.2694.5362.

Virtanen, P. *et al.* (2020) 'SciPy 1.0: fundamental algorithms for scientific computing in Python', *Nature Methods*, 17(3), pp. 261–272. doi: 10.1038/s41592-019-0686-2.

Wager, S. *et al.* (2020) 'Deep Autotuner: A Pitch Correcting Network for Singing Performances', in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 246–250. doi: 10.1109/ICASSP40776.2020.9054308.

Wei-Ho Tsai and Hsin-Min Wang (2006) 'Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals', *IEEE Transactions on Audio, Speech and Language Processing*, 14(1), pp. 330–341. doi: 10.1109/TSA.2005.854091.

Wei Chu and Alwan, A. (2009) 'Reducing F0 Frame Error of F0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend', in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 3969–3972. doi: 10.1109/ICASSP.2009.4960497.

Weis, C., Schreiber, H. and Muller, M. (2020) 'Local Key Estimation in Music Recordings: A Case Study Across Songs, Versions, and Annotators', *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 2919–2932. doi: 10.1109/TASLP.2020.3030485.

Welch, G. F. (1979) 'Vocal range and poor pitch singing', *Psychology of Music*, 7(2), pp. 13–31. doi: 10.1177/030573567972002.

Welch, G. F. (2021) 'Kalman Filter', in *Computer Vision*. Cham: Springer International Publishing, pp. 1–3. doi: 10.1007/978-3-030-03243-2_716-1.

Wen, Q. *et al.* (2020) 'Fast RobustSTL: Efficient and Robust Seasonal-Trend Decomposition for Time Series with Complex Patterns', in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Virtual Event, CA, USA: ACM, pp. 2203–2213. doi: 10.1145/3394486.3403271.

Wilkins, J. *et al.* (2018) 'VocalSet: A Singing Voice Dataset', *ISMIR*, pp. 468–472. doi: https://doi.org/10.5281/zenodo.1193957.

Wu, Y. D. (2013) 'A New Similarity Measurement of Pitch Contour for Analyzing 20th- and 21st-Century Music: The Minimally Divergent Contour Network', *Indiana Theory Review*, 31(1), pp. 5–51. Available at: http://www.jstor.org/stable/10.2979/inditheorevi.31.1-2.0005.

Yang, Y.-Y. *et al.* (2021) 'TorchAudio: Building Blocks for Audio and Speech Processing', *arXiv preprint arXiv:2110.15018*.

Yu, Y. *et al.* (2016) 'Performance scoring of singing voice', *Proceedings of 2015 International*

Conference on Asian Language Processing, IALP 2015, pp. 119–122. doi: 10.1109/IALP.2015.7451546.

Yu, Y. *et al.* (2020) 'Deep attention based music genre classification', *Neurocomputing*. Elsevier B.V., 372, pp. 84–91. doi: 10.1016/j.neucom.2019.09.054.

Zhang, J. and Bryan-Kinns, N. (2022) 'QiaoLe: Accessing Traditional Chinese Musical Instruments in VR', in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, pp. 357–362. doi: 10.1109/VRW55335.2022.00080.

Zhao, X., O'Shaughnessy, D. and Nguyen, M. Q. (2007) 'A processing method for pitch smoothing based on autocorrelation and cepstral F0 detection approaches', *Conference Proceedings of the International Symposium on Signals, Systems and Electronics*, pp. 59–62. doi: 10.1109/ISSSE.2007.4294413.

Zhou, Y. *et al.* (2020) 'Mandarin Singing Synthesis Based on Generative Adversarial Network', in *2020 IEEE 3rd International Conference on Information Communication and Signal Processing (ICICSP)*. IEEE, pp. 139–142. doi: 10.1109/ICICSP50920.2020.9232118.

Zolzer, U., Sankarababu, S. V. and Moller, S. (2012) 'PLL-based Pitch Detection and Tracking for Audio Signals', in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, pp. 428–431. doi: 10.1109/IIH-MSP.2012.110.