UPMC PARIS UNIVERSITAS

ircam Centre Pompidou

Master 1 Internship Report

Automatic Note Detection in Monophonic Sound Files

Author:
Laura Dale

Master 1 Candidate
Sciences de l'Ingénieur, Mécanique
Orientation Acoustique
UPMC

Supervisor:
Dr. Axel Roebel

Head Researcher
Analysis/Synthesis Team
IRCAM

August 21, 2012

Contents

List of Figures

List of Tables

1 Summary

Reliable fundamental frequency (f_0) estimation of monophonic sounds has existed for years. However, most of the existing algorithms require manual adjustment of input parameters for optimal performance on a given sound file. Additionally, though near-perfect accuracy has been attained for relatively clean sound files, performance is known to decline drasti-

cally for cases with background percussion, noise, or reverberation. The first goal of this work was thus to improve upon the current state of monophonic f_0 estimation, not by modifying the estimation algorithm itself, but instead by increasing its automaticity and robustness. By choosing optimal fundamental frequency estimation input parameters for a wide range of files or by setting them algorithmically in relation to the content of each file, the need for user interaction could be eliminated. Additionally, by using an eclectic and challenging database of sound files including many of which had been the source of complaint for previous users of f_0 estimation software, one could hope to improve performance in more complicated cases.

The output of f_0 estimation is an estimate of the fundamental frequency of an audio file at each point in time. In order to detect notes, this f_0 contour must be analyzed, segmented, and converted into musical notes. The second goal of this work was thus to devise an algorithm that would take, as input, the f_0 estimation of a given audio file, and output the set of musical notes contained in the file.

The fundamental frequency estimation software, the algorithms developed to automate the choice of some of its inputs, and the segmentation and note detection algorithms devised to process its outputs, all required the choice of parameters whose optimal settings were yet to be determined. Thus, a final algorithm was developed to evaluate the performance of the combined f_0 estimation and note segmentation protocols by comparing the detected note set with the notes in the original sound file. To rate the success of the complete note detection algorithm for a given parameter set, its results were averaged over the full set of audio files in the database. Finally, a genetic algorithm was employed so that the parameter settings resulting in the best average per-file performance could be found without testing every possible parameter set. The final product of this internship is a relatively robust, automatic note detection algorithm for monophonic and near-monophonic audio files.

This report is divided into the following sections. In the introduction, IRCAM's Analysis/Synthesis team will be presented, followed by some key concepts necessary for comprehension of the presented algorithms. Next, the completed project will be described in stages, from manual segmentation and database compilation to parameter choice. Finally, results will be presented and discussed.

Les logiciels qui estiment la fréquence fondamentale f_0 pour des sons monophoniques existent déjà depuis longtemps. Cependant, ces logiciels requièrent que l'utilisateur spécifie des paramètres d'entrés pour qu'ils fonctionnent de façon optimale pour chaque fichier audio. Même si la précision presque parfaite a été atteinte pour des fichiers sons relativement propres, la performance décline rapidement pour les cas où l'enregistrement est bruité, ou lorsqu'il existe un fond de percussion ou une réverbération. Le premier but de ce travail a donc été d'améliorer l'estimation f_0 monophonique en augmentant sa robustesse et son automaticité. En choisissant les entrées optimales pour une grande variété de fichiers audio, ou en les choisissant automatiquement avec un algorithme en relation avec le contenu de chaque fichier, le besoin de l'interaction de l'utilisateur pourrait être supprimé. De plus, en compilant une base de données éclectiques incluant les fichiers audio dont les utilisateurs des logiciels précédents se sont plaints pour l'estimation de f_0, on peut espérer améliorer la performance de cas plus complexes.

Le paramètre de sortie de l'analyse f_0 est une estimation de la fréquence fondamentale à chaque instant du fichier audio. La courbe f_0 est ensuite analysée, segmentée, et convertie en notes musicales. Le deuxième but de ce projet a donc été de créer un algorithme qui prendrait comme entrée l'estimation f_0 d'un tel fichier audio, et sortirait les notes musicales contenues dans le fichier.

Le logiciel d'estimation de la fréquence fondamentale, l'algorithme développé pour automatiser le choix de certains de ses paramètres d'entrés, et l'algorithme pour détecter les notes à partir de sa courbe f_0 de sortie, tous trois avaient encore des paramètres dont leurs réglages optimaux n'étaient pas encore trouvés. Alors un programme a été conçu pour comparer l'ensemble des notes détectées automatiquement avec les vraies notes du fichier son. Ainsi, il est possible d'évaluer la performance combinée des algorithmes d'estimation de f_0 et de sa segmentation en notes. Pour évaluer la performance d'un ensemble donné de paramètres, les résultats de la détection de notes ont été évalués et moyennés sur tous les fichiers de la base de données. Finalement, un algorithme génétique a été utilisé pour trouver les réglages de tous les paramètres donnant le meilleur résultat pour l'ensemble des fichiers, sans avoir besoin de tester toutes les combinaisons possibles. Le produit final de ce stage est un algorithme automatique et robuste qui détecte les notes musicales pour des fichiers sonores monophoniques et presque-monophoniques.

Ce rapport est divisé en différentes sections. Pour commencer, l'équipe Analyse/Synthèse de l'IRCAM sera d'abord présentée, suivi par la description des concepts clés nécessaires à la bonne compréhension des algorithmes conçus. Ensuite, le projet sera détaillé étape par étape, de la segmentation manuelle et la compilation de la base de données au choix des paramètres. Finalement, les résultats seront présentés et discutés.

2 Introduction

2.1 Analysis/Synthesis team and software development

The Analysis/Synthesis team at IRCAM is involved in the research and development of sound signal processing techniques aimed at analyzing and extracting relevant content, such as pitch, tempo, timbre, syllables, or expressivity, and in modifying that content in order to synthesize new sounds. Since its conception, the Analysis/Synthesis team has developed a wide range of software, allowing its analysis, synthesis, and transformation techniques to be used both internally at IRCAM and also externally. Three of these software packages were utilized in this project - *AudioSculpt*, *SuperVP*, and $F0$. $F0$, a software developed by Matthias Krauledat and Axel Roebel in 2003 [2], estimates the fundamental frequency at all points in an audio file. A more detailed description of the software will be given in section 2.4. *SuperVP* or Super Phase Vocoder, is a command line program that implements various analysis, filtering and re-synthesis techniques for uncompressed sound files. For example, the $F0$ software is included as one of the analysis modules. During this project, two analysis modules were used extensively: the short term Fourier transform (STFT) and, of course, $F0$. Additionally, the band pass filtering module and the noise extraction and re-synthesis module were employed in a window size choice algorithm described in section 3.3.1. *AudioSculpt* is a user-friendly program allowing most of the functionality of *SuperVP* in a more user-friendly format. In *AudioSculpt*, one

can visualize, sonorize, and edit the spectogram, perform f_0 estimations, view and edit results, and notate files with a variety of marker-types, including midi notes, temporal point and region markers, and text markers. Other Analysis/Synthesis software, such as *IrcamBeat*[1], are also included in *AudioSculpt*, such that the program truly incorporates a nice collection of the more mature sound analysis, synthesis, and transformation techniques developed at IRCAM. The output files of both *SuperVP* and *AudioSculpt* are in Sound Description Information Format (SDIF), an open source file type that consists of a series of frames, which contain matrices or single points of data with a customizable header designating the type of information[2]. Frame types containing f_0 estimation data and temporal marker frames were used throughout this project. The position of IRCAM and its Analysis/Synthesis group at the center of music technology research, and the opportunity to interact on a daily basis with the developers of the software utilized was most certainly advantageous to the author throughout her three month stay at IRCAM.

2.2 Harmonicity and fundamental frequency

All sound signals can be represented by a linear combination of sinusoids of different frequencies. What distinguishes a perfectly harmonic signal, however, is that it can be modeled by a finite set of sinusoids, whose frequencies are all multiples of the fundamental frequency, f_0.

$$x(t) = \sum_n A_n sin(2\pi n f_0 t) \tag{1}$$

where n is an integer, and A_n is the amplitude of a given sinusoid. Harmonic or near-harmonic signals are perceived as having a pitch, whereas inharmonic signals are not perceived to be pitched. Normally, the perceived pitch of a harmonic signal is that of a single sinusoid at the fundamental frequency, f_0.

An interesting property of harmonic signals is that despite the contributions of higher frequencies, they are periodic with period $T = 1/f_0$. The period of a higher frequency component is $T_n = 1/(nf_0)$. Thus, each higher frequency component will complete n full cycles during the period of the fundamental, T. It is only after the fundamental period, T, in which all frequency components will have completed an integral number of cycles, and thus the period of the harmonic signal becomes the fundamental period, $T = 1/f_0$. (See figure 1.)

As the perceived pitch is associated with the fundamental frequency, fundamental frequency estimation allows us to traverse between the scientific and the musical domain. Transcription of a musical source involves notation of the sequence and timing of its constituent notes. This is achieved by a variety of formats, including the musical score and the midi file. However, the most important features of any musical transcription are the pitch and the start and stop times of each note. Thus, to create an automatic transcription of a musical sound file, we must first estimate the fundamental frequency at each point in time. Secondly, we break up the fundamental frequency curve into notes.

[1] *IrcamBeat*, developed by Geoffroy Peeters and Frédéric Cornu, estimates the tempo, meter, and the temporal position of beats in an audio file.

[2] SDIF was also developed in part by Ircam, in collaboration with the Center for New Music and Audio Technologies at UC Berkeley and the Music Technology Group at the Universitat Pompeu Fabra.

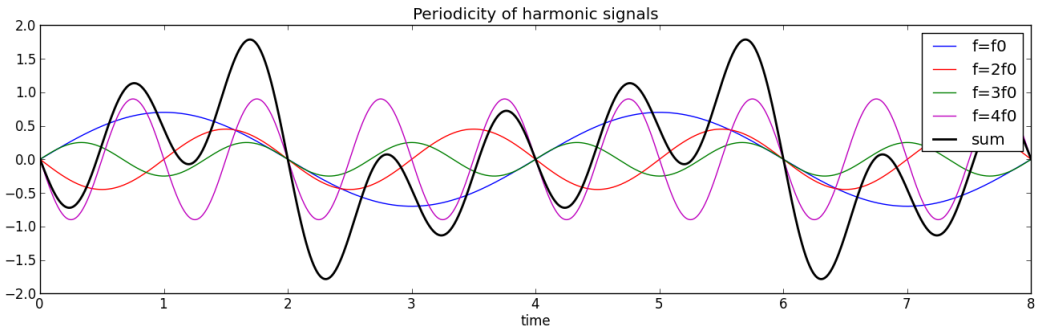

Figure 1: Demonstration of the period of a harmonic signal.
Four sinusoids with frequencies $f_0 = 0.25$ (blue), $2f_0$ (red), $3f_0$ (green), and $4f_0$ (magenta), with varying amplitudes, are displayed. Their sum (black), a harmonic signal, has a period equal to that of the fundamental component, $1/f_0 = 4$.

2.3 Perception and musical distance

The *octave* is the basic unit of musical distance, spanning from a minimum frequency to twice that frequency. In most Western music, octaves are divided into 12 half-notes, and each half-note is subdivided into 100 musical cents, such that an octave spans 1200 cents. However, as perception of pitch and thus, of musical intervals, is logarithmic, these divisions are equally-spaced on a logarithmic scale, but not on a normal scale. Thus, the interval between two notes, with frequencies f_a and f_b is:

$$n = 1200 \cdot log_2(f_a/f_b) \tag{2}$$

where n is measured in cents. Four some instruments, the discretization of the range of possible notes is the half-note, such as woodwind and brass instruments in which the length of the resolating cavity is discretized (clarinet, trumpet, etc.). For others, there is no inherent discretization in the structure of the insrument and thus the range of possible pitches is infinite. Examples include the voice, the trombone, non-fretted stringed instruments like the violin and the cello, and musical synthesizers. In the first group, transcription in musical notation, with its half-note interval structure, is natural. For the second group, the half-note discretization of produced notes is often imposed by a well-trained musician, but as pitches between half-notes can still be produced, the automatic transcription of such instruments is challenging. On may ask whether another representation would be more adequate to represent these instruments. For the purposes of this project, however, the problem was not addressed. All notes were assumed to fit in a half note interval, with a central frequency, f_c and a range of ± 50 cents.

2.4 $F0$ estimation software

Although other reliable monophonic fundamental frequency estimation software exists, $F0$ was used exclusively throughout this internship. As other options were not explored, it is possible that use of another software could have produced better results. For those interested, an overview of available f_0 estimation algorithms and a successful alternative can be found in [1].

$F0$ can be described as follows. The short-term Fourier Transform (STFT) is calculated

for successive frames of an audio file, producing a series of spectra. The relative maxima of each spectrum that lie above a signal-noise threshold are classified as relevant spectral peaks (RSP). Next, a series of harmonic partial series (HPS), consisting of an f_0 candidate and its multiples, are tested for validity. Three criteria are used to quantify the likelihood that the given HPS is indeed the correct one–the harmonicity (HAR), the mean spectral bandwidth (MBW), and the spectral centroid (SCR). The total score, the linear sum of each criteria,

$$S = p_{HAR} \cdot HAR + p_{MBW} \cdot MBW + p_{SCR} \cdot SCR \tag{3}$$

is minimal for the optimal HPS.

The harmonicity criteria evaluates how closely the set of relevant spectral peaks (RSP) can be modeled by a given HPS. For each peak f_i of the RSP, the closest peak in the HPS, f_h is determined, and its degree of deviation is calculated:

$$d_i = \begin{cases} \frac{|f_i - f_h|}{\alpha f_h} & \text{if } |f_i - f_h| < \alpha f_h \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

where α defines a confidence interval, spanning the width of the fundamental frequency and centered around the given peak in the HPS, f_h. The peak salience, $spec_i$, defined as the amplitude of the maximum bin falling within the frequential bounds of the spectral peak, serves to quantify the size of the peak. The HAR measure is then calculated as the salience-weighted sum of the set of deviations, d_i:

$$HAR = \frac{\sum_{i=1}^{I} spec_i \cdot d_i}{\sum_{i=1}^{I} spec_i} \tag{5}$$

Since the HPS of subharmonics (f_0/n, where n is an integer) of the true fundamental frequency include all frequencies in the correct HPS, they can produce similarly low HAR values, as the salience and the deviation of each existing peak in the RSP are equivalent. In consequence, this effect must be countered by the inclusion of criteria that discourage the choice of subharmonics.

The MBW and SCR do just that. The MBW attempts to measure the smoothness of the spectral envelope of a given HPS. The spectral envelope connects the tops of the spectral peaks found at each value f_i in the HPS. The envelope for a subharmonic HPS alternates between found peaks and nonexistent peaks. Thus, even for a clarinet spectrum, in which the odd partials are considerably weaker than the even ones, the subharmonic's envelope is considerably rougher than that of the true fundamental (See figure 2). In turn, the MBW quantifies the inverse of the envelope smoothness, such that smoother HPS envelopes produce lower scores. The SCR, in turn, calculates the energy spread of the envelope of a given HPS. By weighting the energy explained by a given peak at f_h in the HPS by its index h, subharmonics produce higher values, since the index will be $2n$ times higher for each peak. For a more detailed description of these two criteria, refer to Yeh, et. al. [5].

$F0$'s input parameters include:

- the minimum and maximum f_0 candidate frequencies ($f_{0,min}$ and $f_{0,max}$)
- the maximum frequency in the analysis range (F_{max})

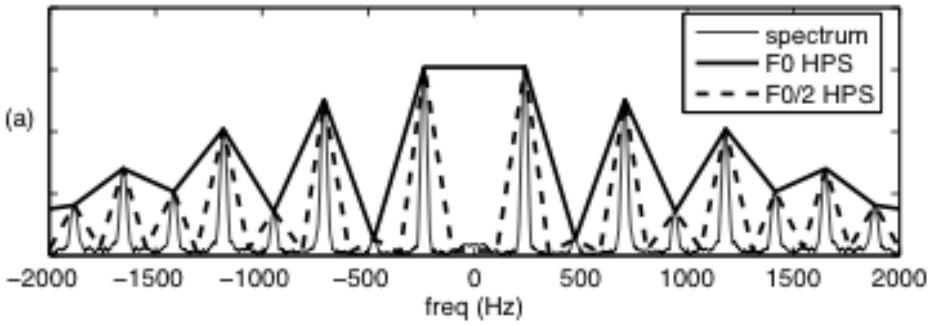

Figure 2: Spectral smoothness.
Comparison of spectral smoothness between the envelope of the HPS of the correct f_0 (solid line) and that of the HPS of the subharmonic $f_0/2$ (dashed line). The spectrum shown is that of a clarinet playing the note Bb3. Taken from [5].

- the window size, M
- the signal noise threshold, sn
- the relative weights, p_{HAR}, p_{MBW}, and p_{SCR} of each criteria in the scoring algorithm (equation. 3)

$F0$ outputs to an .sdif file containing 3 relevant parameters for each frame:

- the estimated f_0, the fundamental frequency of the HPS which obtained the lowest score.
- the harmonicity value, $harm$, calculated by demodulating the signal within a temporal window of $4/f_0$ such that the f_0 is constant over the segment, calculating the unbiased, normalized estimation of the autocorrelation of the demodulated segment, and finally, selecting the local maximum in the vicinity of $t = 1/f_0$. As a harmonic function is periodic with period $T = 1/f_0$ (see section 2.2), the autocorrelation function of a correct estimate will have a local maximum at $t = T$. Normalization of autocorrelation function is performed to ensure that the degree of this periodicity, and thus, the degree of harmonicity of the signal, is expressed as a value between 0 and 1. However, if f_0 varies over the analysis frame, the period T will also vary, and thus the desired peak in the autocorrelation function will be diffused over the range of fundamental periods, T. Thus, the signal is demodulated over the analysis frame so that f_0 remains constant.
- the energy amplitudes value, $ener$, defined as the sum of the squared amplitudes of the saliencies of the RSP.
$$ener = \sum_i spec_i^2 \tag{6}$$

2.5 Importance of window size choice

The choice of window size for a STFT analysis frame has a direct impact on the minimum detectable fundamental frequency of the outputted spectra. This minimum fundamental

frequency (f_{min}) is approximately equal to the bandwidth (B_w) of the main lobe of the window function:

$$f_{min} = B_w = c/M \tag{7}$$

where c is a factor which depends on the windowing function chosen and M is the window size, in seconds. The windowing function, $w[n]$, is applied to the signal in the current frame, $s[n]$, to give the output in the temporal domain, x[n]:

$$x[n] = w[n] \cdot s[n]$$

Application of the discrete Fourier transform gives the convolution product of the window and the signal in the frequency domain:

$$X[k] = W[k] * S[k]$$

For a perfectly harmonic, monophonic source with fundamental frequency f_0, the spectral peaks of $S[k]$ can be approximated as delta functions, spaced f_0 apart. Convolution replicates the window spectrum $W[k]$ at each of the delta functions, to form the output spectrum, X[k]. If $f_0 = f_{min}$, the main lobes centered at each harmonic (mf_0, where m is an integer) will not overlap, and the correct fundamental frequency, f_0, will be discernible. However, if f_0 is much less than f_{min}, the main lobes will overlap considerably and thus the fundamental frequency will not be detected. (See figure 3.)

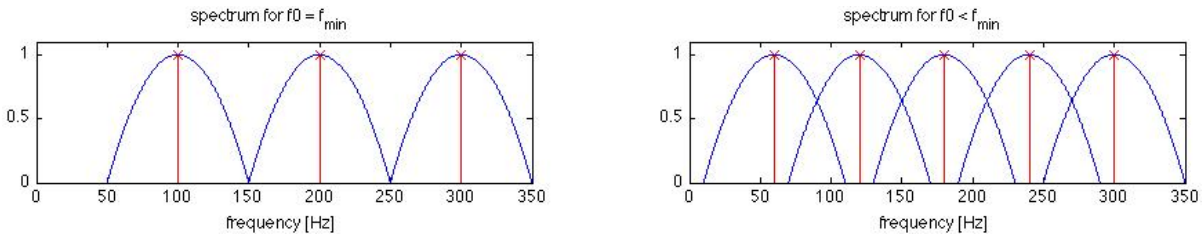

Figure 3: Effect of window size on minimum discernible fundamental frequency.
Idealized frequency spectra for a harmonic source of frequency f_0 (red) and the output spectrum after the application of a window function (blue). When $f_0 = f_{min}$ (see equation 7) (left), adjacent lobes do not overlap, whereas when f_0 is much less than f_{min} (right), the lobes overlap considerably and thus the harmonic structure is no longer distinguishable.[3]

The Hann window, with its main lobe bandwidth factor $c = 4$, was used throughout this project. Thus, the lowest frequency for which there will be no overlap is:

$$f_{min} = 4/M \tag{8}$$

However, as a small amount of overlap between adjacent lobes will not immediately inhibit detection, the true minimum detectable fundamental frequency may be slightly lower than this value.

If the window size is not properly set, and f_{min} lies above some of the notes in the file, these notes cannot be detected, no matter how effective the following note segmentation algorithm may be. Thus, it is of utmost importance to choose an adequate window size before further analysis is performed.

[3]Only the main lobe of the window function is represented.

3 Project Description

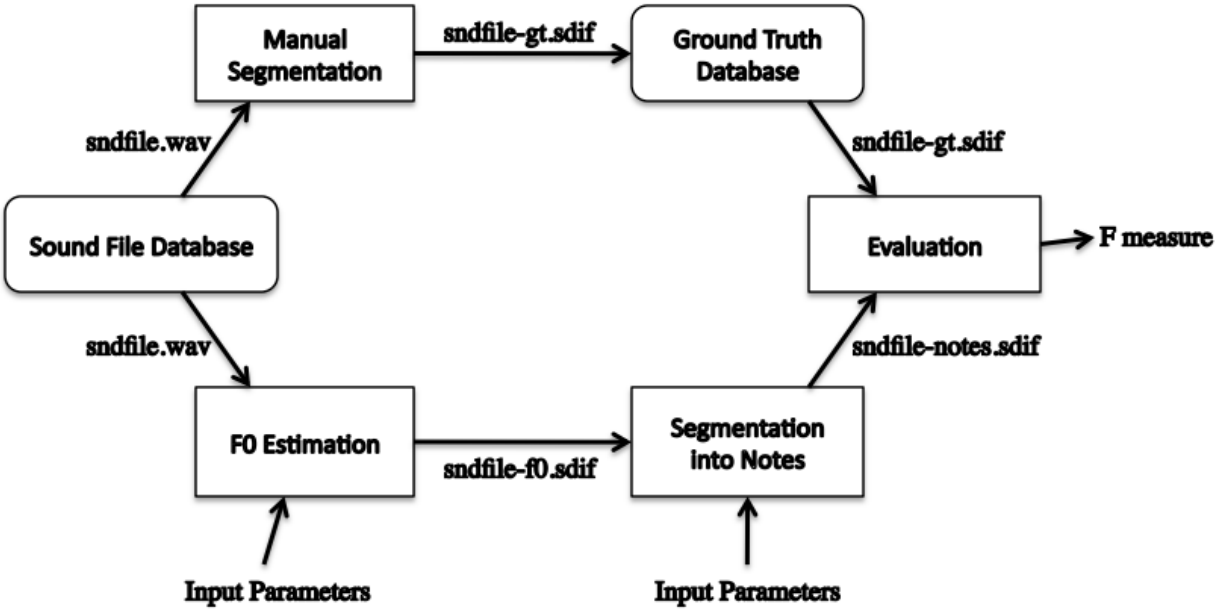

Figure 4: Schematic representation of project

The project can be summarized as follows. First, a database of 132 monophonic and nearly monophonic audio files was compiled (described in section 3.1). These files were manually segmented to create a ground truth database comprised of .sdif files containing the start times and pitches of each note (see section 3.2). Once the ground truth database was created, an automatic note segmentation scheme was developed, consisting of performing an f_0 estimation on a given sound file from the database, segmenting the f_0 output curve into notes, and transferring the relevant note start and pitch information into an equivalent file format so as to allow for easy comparison with the corresponding .sdif file in the ground truth database (described in sections 3.3 and 3.4). An evaluation routine was written which compares the pitches and onset times detected by the automatic pathway (f_0 estimation and note segmentation), against the true pitches and onset times contained in the ground truth database, outputting a score (F-measure) from 0 to 1 (described in detail in section 3.5). As both the $F0$ software and the note segmentation algorithm require a large set of input parameters, a genetic adaptation algorithm was used, optimizing parameter choice to achieve the highest possible average per-file F-measure (see section 3.6). A schematic representation of the project can be viewed in figure 4.

3.1 Sound file database

The database of .wav and .aif audio files compiled for this project includes excerpts from a wide variety of instruments and musical styles, such as computer synthesized sounds, vocal extracts with and without effects, whistling, classical and electric guitar, piano, xylophone, jazz trumpet and saxophone, and tribal chants. Many of the sound files were submitted by prior users of various f_0 estimation programs, complaining that performance was inadequate. Effects such as reverberation and note overlap, and the existence of background percussion and speech mean that many of these files could be classified, strictly

speaking, as polyphonic. The final database aims to push the limits of monophonic f_0 estimation.

3.2 Manual segmentation

The first phase of the project consisted of manually segmenting each audio file in the database. *AudioSculpt 3.0*, a software developed by IRCAM, was used to create .sdif files containing each note's start time and fundamental frequency. Two approaches were used. The first, the *marker* approach, consisted of performing a fundamental frequency analysis, and for each note, placing a marker at the onset, and specifying a range over which the fundamental frequency estimation had been successful for that note. The output of this approach was two .sdif files, one containing the fundamental frequency estimation at each point in time, and the other containing the marker information. However, for certain files, the fundamental frequency estimation was largely unsuccessful, and thus, another approach was necessary. In the *midi* approach, midi notes were drawn on the spectrogram over the frequential and temporal bounds of each note. The output of this approach was, again, an .sdif file, but with a different internal format. Thus, a python program was written that read the output of the two different approaches and wrote an sdif file of a standardized format, including a frame for the start time of each note, followed by a frame containing its fundamental frequency. The final product of the manual segmentation process, the completed ground truth database, comprised one standard-format .sdif file, sndfile-gt.sdif, for each file in the sound file database.

3.3 Fundamental frequency estimation

3.3.1 Window size and frequency range choice algorithm

In order to run an f_0 estimation, the window size (M) and the frequency range parameters ($f_{0,min}$,$f_{0,max}$, and F_{max}) must be specified, among other parameters (see section 2.4) Ideally, the chosen window size should be just large enough that the minimum detectable frequency, f_{min}, falls right below the minimum f_0 in the sound file (see equation 7). Once an appropriate window size is chosen, $f_{0,min}$ can be set to any value greater or equal to f_{min}. Thus, $f_{0,min}$ was automatically set equal to f_{min}. For the purposes of comparing window size choice algorithms, $f_{0,max}$ was set 4 octaves above $f_{0,min}$, and F_{max} was set to twice the value of $f_{0,max}$. To limit program run times, the number of window sizes, N_{ws}, was limited. The minimum window size, $M_{min} = 4/800$ corresponds to a minimum frequency of 800 Hz for a Hann window, with $c = 4$. The maximum window size, $M_{max} = 4/25$ corresponds to a minimum frequency of 25 Hz. The window sizes were equally spaced on a logarithmic scale:

$$M_p = exp\left(\frac{p(logM_{max} - logM_{min})}{N_{ws}}\right) \tag{9}$$

where $p \leq N_{ws}$ is an integer. In order to test window size choice algorithms, N_{ws} was set to 6, corresponding to f_{min}'s, one octave apart. However, the number of window sizes (N_{ws}), the window size factor (c in equation 7), the number of octaves between $f_{0,min} and \mathrm{f}_{0,max}$ (N_{oct}), and the multiplicative factor between $f_{0,max}$ and F_{max} (FF), were eventually taken as variables to be set by the *Genadapt* parameter choice program.

Two algorithms were developed to automatically choose an appropriate window size, the *signal energy* algorithm and the *harmonicity-energy amplitudes* algorithm.

The *signal energy* algorithm performs an STFT, applies a band pass filter ranging from $f_{0,min}$ to F_{max}, and creates a new sound file after having eliminated all noise, for each possible window size (see eq. 9). The noise elimination algorithm consists of making a distinction between noise peaks and harmonic peaks in each spectrum, eliminating the noise peaks, and resynthesizing a soundfile by combining the inverse DFTs of the set of noise-eliminated spectra [6]. Next, the signal energy, E, of each resynthesized file is calculated:

$$E = \sum_{n}^{N} |x[n]|^2 \tag{10}$$

where N is the length, in samples, of the sound file. As the re-synthesized file with the correct frequency range and window size will contain the energy of as many of the notes in the original file as possible, the window size (and corresponding frequency range parameters) which produces the maximum signal energy is selected.

Likewise, the *harmonicity-energy amplitudes* algorithm performs an STFT and then a noise-eliminating re-synthesis for each window size. However, instead of calculating the signal energy, an f_0 estimation is performed for each window size. Then, the product of the harmonicity (*harm*, see section 2.4) and energy amplitude (*ener*, see equation 6) outputs for each frame is summed and normalized by the number of frames, to determine the HE value:

$$HE = \sum_{j}^{J} \frac{harm[j] \cdot ener[j]}{J} \tag{11}$$

where J is the number of frames. The window size (and corresponding frequency range parameters) producing the maximum HE value becomes the optimal choice. Notes are often correlated with periods of high harmonicity values; consequently, a high average harmonicity per frame is desirable. However, very weak signals, such as electrical noise at 25 and 50 Hz, can be highly harmonic and encourage a choice of window size that allows its detection. Thus, the energy amplitudes value was included to ensure that the harmonic sources biasing the window size choice were truly significant. Moreover, as different window sizes result in a different number of output frames for the same sound file, normalization by J becomes necessary. With these modifications made, it was hypothesized that the f_0 estimation with the highest HE value would correspond to an intelligent choice of window size.

Once designed, the two algorithms were applied to the entire sound file database and evaluated based on their ability to choose a window size large enough for the minimum fundamental frequency of the sound file to be detected, and ideally, the largest window size that would allow such detection, corresponding to the highest possible f_{min} which lies below the minimum f_0 in the sound file. The ideal window size is thus the one that allows detection of all notes in the file and minimizes the possibility of subharmonic errors. The signal energy approach succeeded in choosing a window size that allowed detection of the full range of f_0's in the file 68.9% of the time, 35.7% of which a smaller window size could have been selected, while the harmonicity-energy amplitudes approach succeeded for up to 94.6% of the files, 13% of which a suboptimal but nonetheless functional window size

was selected. For the 5.4% for which a non-ideal window size was selected, some but not all of the notes could be detected. Thus the percentage of notes rendered detectable by the harmonicity-energy amplitudes alrogithm could be considerably higher than 94.6%. Though the signal energy approach was somewhat faster than the harmonicity-energy amplitudes approach, as only one fundamental frequency estimation was performed as opposed to one per window size choice, the 30% increase, on average, in run time was deemed worth the increased precision, and the harmonicity-energy amplitudes approach was selected.

As the harmonicity-energy amplitudes approach includes f_0 estimation, no further f_0 analysis is necessary. The sndfile-f0.sdif file derived from the optimal window size is then passed on to the note segmentation algorithm.

3.4 Segmentation into notes

The note segmentation algorithm is primarily based on two limits, the *harmonicity threshold* and the *relative derivative limit*. The use of a harmonicity threshold is a direct product of this project, whereas the idea for the use of a relative derivative limit can be attributed to Rossignol, et. al. [3].

Harmonicity is the degree to which a sound can be modeled by a discrete set of harmonics (see section 2.2). Notes are thus characterized by high harmonicity values, whereas noise is characterized by low harmonicity values. Additionally, for instruments in which the length of the resonating cavity is discretized, harmonicity should decline when changing from one note to the next. Thus, by setting a harmonicity threshold, $harm_{th}$, for the f_0 curve, one can effectively eliminate noise sections in relatively clean sound files. f_0 values for which the harmonicity is above this level are kept, while f_0 values for which the harmonicity level is below the threshold are thrown out. However, for instruments which do not produce a discrete set of pitches, another parameter is necessary to differentiate between adjacent notes.

The relative derivative measures the speed at which the fundamental frequency changes. A note change is often the cause of a relatively quick change in the fundamental frequency. Thus, by setting an upper limit on the magnitude of the relative derivative, one can hope to segment the f_0 curve into notes. The relative derivative is defined as follows:

$$rd = \frac{f_0[j+1] - f_0[j]}{dt \cdot (f_0[j] \cdot f_0[j+1])^{1/2}} \tag{12}$$

where dt is the time between two samples in the sndfile-f0.sdif file, which is equivalent to the window size divided by the hop size between adjacent frames (set to $M/8$). Division by dt is necessary as different window sizes will produce different time steps, and the optimal window size for each file will vary based on the contents of the soundfile. Normalization by the geometric mean of the two samples in question is performed to ensure that a jump upwards and a jump downwards perceived as equal in musical distance (see section 2.3) would result in relative derivatives of the same magnitude. Apart from helping to detect note changes, the relative derivative threshold also serves to eliminate f_0 estimates that vary extremely quickly in frequency, often faster than would be physically possible. Possible

sources of such a variation include the rapid fluctuation between a subharmonic and the true f_0, and f_0 estimates deriving from noise. In both cases, variation of this type should be removed from analysis.

Two additional parameters, the minimum segment length (msl) and the minimum note length (mnl) also serve to eliminate undesirable f_0 estimates. As extremely short segments are probably the result of either noise or poor f_0 estimation, segments shorter than the minimum segment length are disregarded. Likewise, notes shorter than a certain threshold often have a similar origin. Moreover, there exists a temporal minimum in which instrumentalists can change notes, and also a perceptual limit to which humans can perceive note changes. Thus, even in the case of computer synthesized music, it is unlikely that notes shorter than this perceptual limit would exist. For these reasons, both a minimum segment length and a minimum note length were utilized.

3.4.1 Combining fixed and adaptive parameters

It is unclear, at first glance, whether the inputs to the note segmentation algorithm ($harm_{th}$, rd_{lim}, msl, and mnl) should be the same for all files, or whether they should be somehow adapted in relation to the content in each audio file. If these thresholds represent true physical limits, then a constant value for all files is desirable. For example, if there exists a harmonicity value above which it becomes highly unlikely that a fundamental frequency estimate could be the result of noise, then that value would be the ideal harmonicity threshold. Along those lines, if there exists a relative derivative corresponding to a certain steepness of curve that can be most likely attributed to a note change, then that fixed value should be the relative derivative limit. However, if this value is below the average —rd— values resulting from vibrato, perhaps the rd_{lim} should be set just above the vibrato-—rd— values, such that vibrato is averaged out over the length of a segment. Thirdly, if there is indeed a perceptual or physical limit to note change durations, then we should set our minimum segment and note lengths based on this value.

On the other hand, one can argue that adjusting these parameters in relation to each audio file might produce better results. If in each audio file, there is a clustering of higher harmonicity values representing notes and a clustering of lower harmonicity values corresponding to the times when no note is present, then we should set our harmonicity threshold between these two clusters. As a vocalist is more inclined to slide between pitches than a flutist, the spectrum of rd values produced by a flutist will be different than that of a vocalist. Additionally, as two instrumentalists may have different tendencies as to the amplitude of vibrato, or whether they often glide upwards or downwards within a single note, their rd spectrums will reflect those differences. Consequently, it might be better to set the relative derivative limit based on the specific rd distribution of a given file. Finally, if the tempo of an audio file can be reliably detected, the minimum note length could be set as just inferior to some fraction of the duration of a beat.

To confront this problem, a combined approach was utilised. The distribution of harmonicity values in each sound file is fit by a two-component Gaussian mixture model (GMM). The Gaussian with the lower mean was expected to model the lower cluster of harmonicity values representative of noise and the lack of notes, and the Gaussian with the higher

mean is expected to model the upper cluster, representing the presence of notes. The modeled probability distribution is then the sum of the two component Gaussians. The local minimum of the modeled probability distribution that lies between the two component Gaussians is then used as a reference point (ref). When no local minimum exists, the intersection point of the two component Gaussians is used. Finally, an additional parameter, the distance from this reference point ($dist$) at which to set the harmonicity threshold, is included, such that the final harmonicity threshold becomes:

$$harm_{th} = ref + dist \tag{13}$$

A depiction of the algorithm is provided in figure 5.

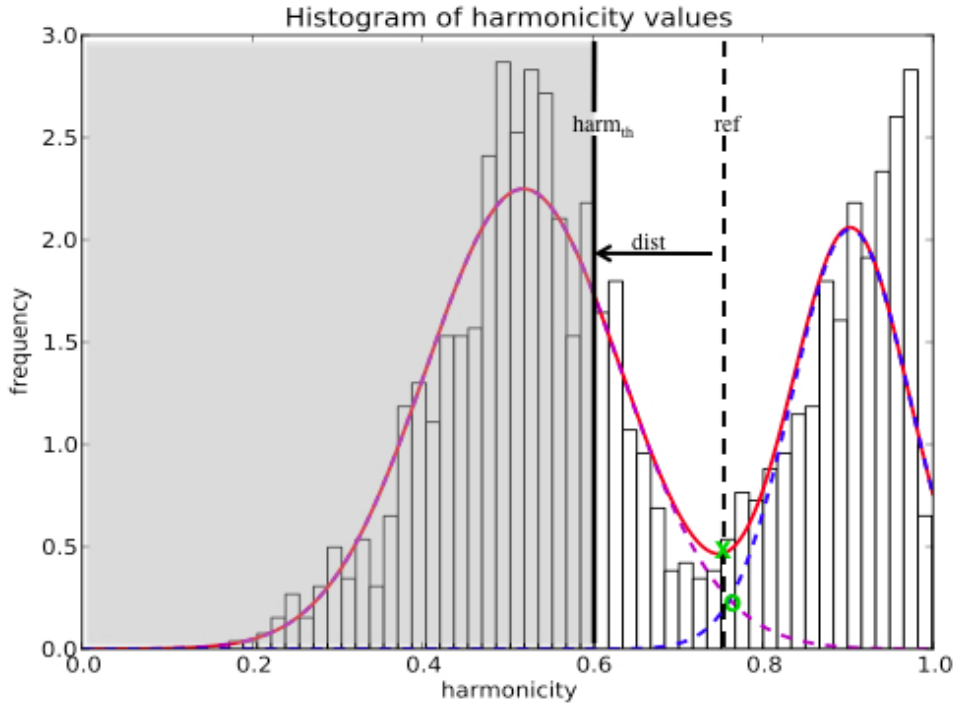

Figure 5: Depiction of algorithm used to set harmonicity threshold.
A two-component Gaussian mixture model is used to model the harmonicity distribution of a given sound file. The sum of the upper Gaussian (dashed line, blue) and the lower Gaussian (dashed line, magenta) is the modeled probability distribution (red). The local minimum between the two Gaussians (x, green) is then used as a reference value (ref). When no local minima exists, the intersection of the two Gaussians (o, green) is used instead. The $dist$ parameter (negative in the diagram) determines the distance between this reference value and the harmonicity threshold ($harm_{th}$). All harmonicity values below this value are excluded from further analysis (shaded in grey).

However, both a strict upper limit ($harm_{max}$) and a strict lower limit ($harm_{min}$) are included. If the calculated harmonicity threshold lies above $harm_{max}$ or below $harm_{min}$, $harm_{th}$ defaults to the closer of the two strict limits. Overlapping ranges of the possible values for $harm_{max}$ and $harm_{min}$ were provided to the genetic parameter choice algorithm. Thus, if a fixed harmonicity threshold were indeed optimal, then the genetic algorithm could set the two parameters as equal.

A similar, but somewhat simpler approach is used to set the relative derivative limit. First, the harmonicity threshold is employed to eliminate certain undesirable portions of the f_0 curve. As the rejected portions are not included in the final segmentation, only the rd

values of the remaining portions of the curve are included when a single-Gaussian is fit to the $\{rd\}$ distribution. Finally, a new parameter, the number of standard deviations, $N_{std,rd}$, is used to set the rd limit, as follows:

$$rd_{lim} = N_{std,rd} \cdot std(\{rd\}) \tag{14}$$

However, using the same reasoning as for the harmonicity threshold, strict upper (rd_{max}) and lower limit (rd_{max}) parameters are included as well. Therefore, if a fixed rd_{lim} for all files were optimal, then the genetic algorithm could set these two parameters equal. In this case, the results of the Gaussian description of the rd values would essentially be ignored as the default value would always be chosen.

As the tempo cannot be reliably detected for the given set of files in the soundfile database, the minimum note and segment lengths are not adapted to each file, and are instead lest as fixed parameters to be set by the genetic adaptation algorithm.

3.4.2 The case of vocal files

As previously mentioned, the voice is a particularly challenging instrument for automatic note detection as it doesn't produce a discretized set of fundamental frequencies. The voice can produce a continuous set of pitches between the upper and lower limits of the singer's range.[4] Although unfretted stringed instruments and the trombone also share this quality, most instrumentalists constrain the set of notes produced to half-note intervals to conform with current musical culture. However, it is both more challenging for singers to hit half-note intervals perfectly, and also more culturally and stylistically acceptable for singers to slide into and out of notes. Thus the magnitude of the relative derivative would be, on average, considerably less at note changes in vocal files than for other instruments. Additionally, of all instruments, vibrato extent (amplitude) reaches its maximum in vocal music, ranging from up to $\pm$ 100 cents, whereas for bowed strings is is usually contained to $\pm$ 50 cents [4]. With a rate of vibrato ranging from 4 to 7 Hz [4], it is possible for the magnitude of the relative derivative within sung notes containing vibrato to reach higher values than at note changes. Thus, ideally, significant vibrato should be removed before the rd distribution is compiled and before the rd_{lim} is calculated.

Since the robustness of the automatic note detection was of chief importance in this study, removing all vocal files from the analysis was deemed undesirable. The vocal files were retained in the database, and were included when optimal parameters were chosen for the completed note detection algorithm. However, a case study was performed on the subset of files containing monophonic voice excerpts. The parameters of the original algorithm were optimized by the genetic algorithm for just the subset of vocal files, and additionally, an alternative algorithm was developed that included vibrato suppression, and its parameters were also optimized, to see if the results for vocal files could be ameliorated. The alternative algorithm is described as follows.

First, the f_0 curve is divided into a set of preliminary segments by the harmonicity threshold and an initial, fixed relative derivative limit ($rd_{lim,prelim}$), intended to remove poor f_0

[4]Excepting, of course, a change in quality and a potential gap due to the switch from chest voice to head voice, and for male singers, from head voice to *falsetto*.

estimates and noise, but retain vibrato. A back-of-the-envelope calculation estimates the maximum rd value that would be obtained in a vibrato-containing segment to be 5.55. For the detailed calculation, see section A, in annex. With this in mind, the range of possible $\{rd_{lim,prelim}\}$ given to the genadapt algorithm was from 4 to 15.

To suppress vibrato, the method of interpolation and averaging between the local maxima and local minima trajectories in regions of significant vibrato was adopted from Rossignol, et. al. [3]. Here, the M_{value}, or the mean discrepancy between the interpolated local minima and interpolated local maxima trajectories is calculated for each segment. An M_{value} threshold parameter, M_{th} is included. Segments with M_{value}'s above this limit undergo vibrato removal, while those whose M_{value}'s are inferior to the limiting value are left untouched. In Rossignol, et. al. [3], the M_{value} for a vocal audio file containing significant vibrato was 0.087, and 0.032 for a file containing no vibrato. Thus, the set of potential threshold values, $\{M_{th}\}$, passed on to the genadapt parameter choice algorithm ranged from 0.04 to 0.09. For the successful implementation of Rossignol, et. al.'s vibrato suppression algorithm, two other parameters were deemed necessary. Even in vibrato sections, the f_0 curve produced by $F0$ is not perfectly smooth, often containing minute internal variations in a single period. The targeted local maxima and minima are not those of these internal variations, but instead the peaks and valleys of the vibrato pulse. Thus, a minimum extent of a local extrema, in number of samples (N_{pts}), was adopted; consequently, local extrema resulting from a single or small number of misplaced estimates above or below the vibrato curve could be eliminated. Additionally, a minimum temporal distance between adjacent extrema, $dt_{min,ext}$, was also included. After vibrato suppression, the distribution of rd values is compiled, the rd_{lim} is calculated as described in section 3.4.1, and a final segmentation of the f_0 curve is obtained.

3.4.3 Segment addition

The relative derivative limit and the harmonicity threshold serve to break up the original, contiguous f_0 curve into segments for which $harm \geq harm_{th}$ and for which $|rd| \leq rd_{lim}$. Under stringent conditions, the curve will be divided into many more segments than the number of notes in the file. However, the remaining segments will be more likely to represent valid information. To allow for such conditions, an algorithm must be implemented that compiles a set of notes from a potentially larger set of segments. To this end, the average fundamental frequency of each segment is calculated, to form a list of segment pitches, $\{p_i\}$. If p_{i+1} is within a half note range of p_i ($p_{i+1} = p_i \pm$ 25 cents), then the two segments are "added together" to form the same note. If the pitch of the $i + 2^{th}$ segment also falls within a half note range of the original i^{th} segment ($p_{i+2} = p_i \pm$ 25 cents), then it too is "added" to the other two segments to form a single note. This process continues until a segment is found whose pitch, p_j, lies outside the half note range of the original segment pitch, p_i; in this case, a new note is started, with p_j as the new reference pitch. The "addition" of segments amounts to a simple prolongation of the note in the temporal domain, and the duration-weighted average pitch in the frequency domain. The pitch of the compiled note becomes:

$$p_{note} = \frac{\sum_i^{j-1} p_i \cdot d_i}{\sum_i d_i} \tag{15}$$

where d_i is the temporal duration of a given segment. A schematic representation of the "segment addition" process can be viewed in figure 6. As only the onset times of each note are saved in the Ground Truth Database, notes are extended until the start of the following note.

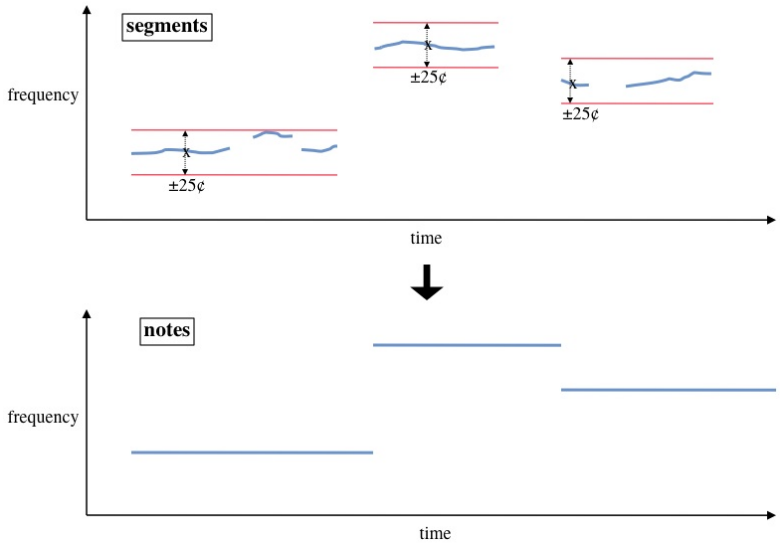

Figure 6: Segment addition.
Segments of a typical f_0 curve are displayed in blue, above. The average pitch of the first segment in a group (x) determines the frequency bounds of the set of segments that will eventually form one note (red lines). Once a segment's average pitch falls outside of these bounds, a new note is created, and a new set of frequency bounds are calculated. The set of notes derived from the segmentation addition algorithm is displayed in blue, below.

One obvious downside of such a segment addition algorithm is that repeated notes, or consecutive notes with pitches within a half note of each other, cannot be detected. The assumption was made that the increased reliability of stringent harmonicity and relative derivative conditions would be worth the loss in repeated note detectability. Additionally, repeated note detection would require the development of new algorithms such as the search for small dips in harmonicity or the division of syllables in sung files, which are unnecessary if the repeated note problem is ignored. Due to time constraints, these approaches were not elaborated.

3.5 Evaluation

The evaluation routine developed serves to compare the set of notes detected in a single audio file by the automatic pathway to the set of notes stored in the ground truth database.[5] Each note in the ground truth database consists of a pitch and a start time, and thus can be represented by a point in frequency-time space. A detected note, also consisting of a pitch and a start time, is compared to the closest note in the ground truth file. If it is within $\pm 20ms$ on the temporal axis and within ± 25 cents on the frequency axis, the note is given the full score, 1. If the note is within $\pm 100ms$, and within ± 50 cents on the frequency

[5]As the segment addition algorithm (described in section 3.4) is incapable of detecting repeated notes, the repeated notes in the ground truth file were combined in a similar fashion, such that multiple notes within a half-note range of the original note were joined to become just one note.

axis, but outside the aforementioned inner bounds, the note is given a score between 0 and 1, based on the magnitude of its error in both the frequency and temporal dimensions (See figure 7). The bounds were chosen so as to allow for the imperfect detection of notes, while demanding a precision that is perceptually satisfying. In this outer target zone, the final note score, sc, is the average of the temporal score, t_{score}, and the frequency score, c_{score}.

$$t_{score} = \frac{100 - \Delta t}{100 - 20}, \quad c_{score} = \frac{100 - \Delta c}{100 - 50}, \quad sc = \frac{1}{2}(t_{score} + c_{score})$$

where Δt is the difference, in ms, between the detected note's start time and the closest start time in the ground truth file, and Δc is the difference, in cents, between the detected note's pitch and the ground truth pitch. The scores of the "correctly" detected notes (falling within the outer bounds of a note in the ground truth file) are summed to equal the true positive rate, tp. The number of detected notes lying outside of the outer target zone is the false positive rate, fp. The number of notes in the ground truth file that are not detected by the automatic pathway gives the false negative rate, fn. These three measures are then combined to calculate the precision, $prec$, the recall, rec, and finally, the F measure (F) for the sound file.

$$prec = \frac{tp}{tp + fp}, \quad rec = \frac{tp}{tp + fn}, \quad F = \frac{2 \cdot prec \cdot rec}{prec + rec} \tag{16}$$

As neither the imprecise over-detection of notes (allowing for the detection of close to all notes in the ground truth file, but many false notes as well), nor the precise under-detection of notes (nearly all detected notes are true notes but many of the true notes are left undetected) is desirable, the F measure provides an accurate score for the performance of the automatic pathway for a given audio file. The total F score (F_{total}) is then the average F measure for all sound files.

$$F_{total} = \frac{1}{N} \sum_{n} F_n \tag{17}$$

where N is the number of sound files in the database. With the the sound file and ground truth databases compiled, and the window size choice algorithm, the $F0$ software, the note segmentation algorithm and the evaluation routine in place, the final step is to select optimal input parameters for the f_0 estimation and note segmentation chain.

3.6 *Genadapt*, a genetic adaptation algorithm

Genadapt, written by Axel Roebel, is an genetic adaptation code that efficiently calculates optimal or near-optimal parameter sets given a list of possible values for each parameter. Instead of evaluating the performance of every possible combination of input parameters, *Genadapt* uses a process modeled off of natural selection and genetic adaptation to "evolve" towards optimal parameter choice. First, a generation of 80-100 random parameter sets, or *individuals*, is created. The performance of each parameter set is then evaluated, and the best 25 individuals are selected, forming the *parents* of the next generation. Through a series of recombinations and mutations from these parent sets, the next generation of 80-100 individuals is formed. In this manner, the performance of the best individuals slowly increases until the variation in parameters, or *traits* of the population diminishes.

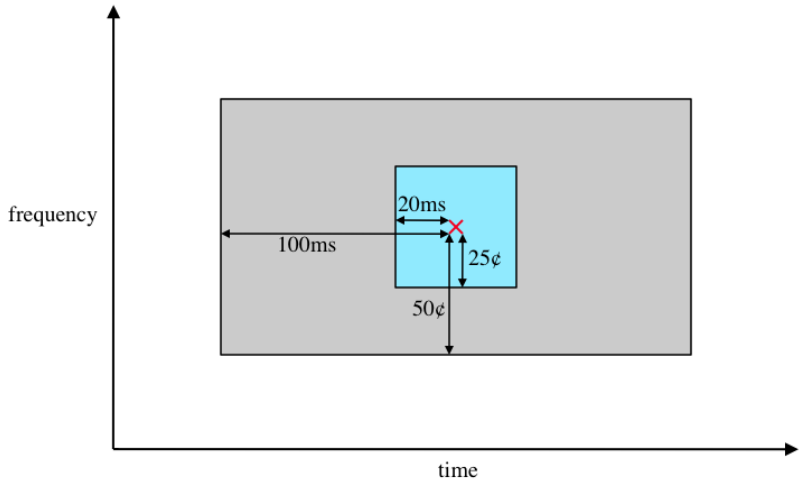

Figure 7: Calculation of the F-measure.
In comparing an automatically detected note to a note in the ground truth database (red x), two target zones are designated. A note found in the inner target zone (blue) within ±25 cents of the true fundamental frequency and within $\pm 20ms$ of the true start time is given a full score of 1. A note found within the outer target zone (grey) but outside of the inner target zone, is given a score on a sliding scale from 0 to 1 based on its distance from the inner target zone in both the frequency and the temporal dimensions. A note found outside of the outer target zone earns a score of 0.

Eventually, the results will stabilize, trait variation will reach a minimum, and the best performing individual is deemed representative of the optimal parameter set.

In our case, the input parameters to the window size choice algorithm, the remaining parameters for f_0 estimation not selected by the window size choice algorithm[6], as well as the input parameters to the note segmentation algorithm were inputted to the genetic adaptation algorithm. A complete description of each parameter, its function, and its source is given in table 1.

4 Results

After 56 generations of the *Genadapt* algorithm, the best individual for the general algorithm had a 76.5% success rate, that is, that on average, 76.5% of the notes in each file were satisfactorily detected. As different instruments typically have different spectra and different note change characteristics, it is quite possible that the overall results would be maladapted to certain instruments types, and better adapted to others. Thus, the results were sorted in terms of instrument type. (See table 2.) Unsurprisingly, vocal files were the most challenging to segment.

[6]The default values of the relative weights of the HAR, MBW, and SCR criteria were used for all tests.

Symbol	Full Name	Description	Source	V/F
sn	signal noise threshold	decibel range considered; spectral peaks more than sn below the max are excluded from analysis	f_0E	V
nl	noise level	ratio of noise versus sinusoidal components in resynthesis	WSC	V
ϵ_{nl}	noise level error	ratio of noise peaks that are misclassified as sinusoidal peaks in noise extraction and resynthesis	WSC	V
N_{wins}	number of window sizes	number of window sizes tested for each soundfile before choosing the optimal one	WSC	V
M_{min}	minimum window size	lower limit in determining N_{wins} windows that are equally spaced in the logarithmic domain (see equation. 9)	WSC	F: 0.005
M_{max}	maximum window size	upper limit in determining N_{wins} windows that are equally spaced in the logarithmic domain (see equation. 9)	WSC	F: 0.160
M_{lim}	window size limit	if minimum window size is below this value, calculate $f_{0,min}$, and then use this window size thereafter	WSC	V
c	window size factor	multiplicative factor in inverse relation between window size and minimum detectable fundamental, f_{min} (see equation 7), used to determine $f_{0,min}$'s	f_0E	V
N_{oct}	number of octaves	determines f_0 frequency range; distance between $f_{0,min}$ and $f_{0,max}$	f_0E	V
FF	frequency factor	determines upper limit of frequencies considered; ratio between $f_{0,max}$ and F_{max}	f_0E	V
p_{HAR}	weight of HAR score	relative importance of HAR versus MBW and SPC criteria	f_0E	F: 0.26
p_{MBW}	weight of MBW score	relative importance of MBW versus HAR and SPC criteria	f_0E	F: 0.14
p_{SCR}	weight of SCR score	relative importance of SCR versus MBW and HAR criteria	f_0E	F: 0.40
$dist$	distance from ref to set $harm_{th}$	distance from the reference value (local min of 2-component GMM distribution or intersection between the two Gaussians) at which to set $harm_{th}$.	NS	V
$harm_{th,min}$	minimum $harm_{th}$	If the adaptive approach sets the harmonicity threshold below this value, $harm_{th}$ defaults to $harm_{th,min}$	NS	V
$harm_{th,max}$	maximum $harm_{th}$	If the adaptive approach sets the harmonicity threshold above this value, $harm_{th}$ defaults to $harm_{th,max}$	NS	V
$N_{std,rd}$	number of standard deviations; rd_{lim}	Number of standard deviations of the rd distribution at which to set the rd limit.	NS	V
$rd_{lim,min}$	minimum rd_{lim}	If the adaptive approach sets rd_{lim} below this value, rd_{lim} defaults to $rd_{lim,min}$	NS	V
$rd_{lim,max}$	maximum rd_{lim}	If the adaptive approach sets rd_{lim} above this value, rd_{lim} defaults to $rd_{lim,max}$	NS	V
msl	minimum segment length	segments shorter than this duration are discarded	NS	V
mnl	minimum note length	after addition of undiscarded segments into notes, notes shorter than this duration are discarded	NS	V
$rd_{lim,prelim}$	preliminary rd_{lim}	alongside harmonicity threshold, determines initial set of segments that will ne scanned for significant vibrato	NS-V	V
N_{pts}	local extrema extent	minimum number of samples to both right and left which are inferior (local max) or superior (local min) to a given sample such that the sample is categorized as a local extrema	NS-V	V
$dt_{min,ext}$	minimum extrema discrepancy	minimum temporal distance between two local extrema.	NS-V	V
M_{value}	minimum vibrato level	minimum mean difference between the interpolation of the set of local minima and the interpolation of the set of local maxima which will invoke vibrato suppression.	NS-V	V

V: variable, **F**: fixed, **WSC**: window size choice, f_0**E**: f_0 estimation, **NS**: note segmentation **NS-V:** note segmentation, voice specific algorithm

Table 1: Complete list of parameters from the developed note detection algorithm inputted to *Genadapt*.

Instrument type	F_{avg}
voice	0.640
whistling	0.775
bass synthesizer	0.935
soprano synthesizer	0.823
woodwinds and brass	0.693
plucked strings	0.739
bowed strings	0.838
piano	0.704
multiple	0.681
total	0.765

Table 2: Results by instrument type.

After 46 generations, the best individual for the voice specific algorithm had a 70.0% success rate. When compared to the success of the vocal files in the general algorithm, this represents a 6% improvement. However, since all the other parameters were also catered to just the subset of 37 vocal files, the addition of vibrato suppression cannot account for all of this improvement. When the sound file database was restricted to vocal files, and the parameters of the original algorithm were optimized, the success rate was 66.4%. Thus, inclusion of a preliminary relative derivative limit, followed by vibrato suppression, improves results by a total of 3.6%.

It is interesting to compare the best set of parameters chosen for the general algorithm versus the parameter set chosen for the vocal specific algorithm. (See table 3.) Many parameter settings are quite similar, such as the minimum window size, M_{lim}, the window size factor, c, and the number of octaves included in f_0 estimation, N_{oct}. The chosen values for these parameters are clearly not instrument-specific, and should thus be optimal for all instrument types. Others, such as the minimum segment and minimum note length, are larger for the vocal specific algorithm than in the general case. These higher values reflect the aforementioned fact that minimum vocal note durations are often longer than those of other instruments for which note change is physically easier to control. Additionally, the upper and lower harmonicity threshold limits ($harm_{th,min}$ and $harm_{th,max}$) and the distance parameter ($dist$) are set in such a way that the ideal harmonicity limits for vocal files are usually 10% lower than in the general case. However, as the voice-specific algorithm included two relative derivative limits, one that contributed to preliminary segmentation and the other that segmented after significant vibrato was suppressed, it is possible that some of the noise portions and undesirable f_0 estimates were eliminated by the two relative derivative limits and therefore a higher harmonicity threshold was unnecessary. Though a higher optimal $N_{std,rd}$ for the voice-specific case as opposed to the general case might seem to indicate that a higher relative derivative limit is optimal in vocal files, meaning that steeper jumps are left unsegmented, this is not necessarily the case. Due to the preliminary rd limit, the rd values forming the distribution in the vocal case are smaller in magnitude, on average, than those in the general case. In this manner, $N_{std,rd} = 2$ standard deviations from center is often lower than the $N_{std,rd} = 0.5$ standard deviations in the general case, resulting in a lower rd_{lim} capable of detecting the sliding note changes characteristic of vocal audio excerpts.

The question was posed in section 3.4.1 whether adaptive or fixed parameters were optimal for the note segmentation algorithm. To answer, a tally was made of the number of audio

files whose $harm_{th}$ and rd_{lim} were set adaptively, falling within the strict upper and lower limits, and how many were set by the default limit values. For the general algorithm, the harmonicity threshold was set adaptively for 112 audio files, whereas 22 were set equal to the lower limit, 0.45. Interestingly, the upper limit could have been set almost 10% lower with no effect on results, as for only one audio file was the harmonicity threshold set above 0.70. In the case of the relative derivative limit, for 56 out of the 132 audio files, $harm_{th}$ was set equal to the fixed lower limit, while for 3 files it was set equal to the upper limit. In both cases, the combined approach is clearly advantageous; however, it seems as though a fixed relative derivative limit would work better than a fixed harmonicity threshold.

parameter	**general**	**voice-specific**
sn	36	32
nl	0.5	0.25
ϵ_{nl}	0.16	0.2
N_{ws}	12	11
M_{lim}	0.015	0.014
c	4.8	4.7
N_{oct}	3.5	3.5
FF	2	1.5
$dist$	-0.2	-0.3
$harm_{th,min}$	0.45	0.35
$harm_{th,max}$	0.8	0.7
$N_{std,rd}$	0.5	2
$rd_{lim,min}$	2	1.5
$rd_{lim,max}$	34.5	34.5
msl	0.04	0.058
mnl	0.085	0.11
$rd_{lim,prelim}$	–	14.5
N_{pts}	–	5
$dt_{min,ext}$	–	0.011
M_{value}	–	0.09

Table 3: Comparison of optimum parameters for general versus voice-specific algorithm.

As the total score is highly depended on the evaluation routine, it is challenging to compare the final results to other works. Additionally, the set of audio files used in this study included many files which would be excluded from other studies as, due to existence of percussion or reverberation, they could be classified as polyphonic. Consequently, the final score of 76.5% could be easily improved if these types of files removed from analysis. Additionally, though a sliding F-measure from 0 to 1 based on the proximity of a detected note to a true note was useful for training, one could argue that a binary evaluation routine should be used in calculating the true performance of the alrogithm; the entire frequential range of the true pitch ± 50 cents and the entire temporal range of the true note start time $\pm 100ms$ should be given a full score of 1, whereas detected notes outside this range would receive a score of 0. If these modifications were made, the final average F-measure might be considerably higher than 76.5%. Thus it is hard to say whether 76.5% is "good" or not. In any case, it is possible to describe the sorts of errors made and to determine whether results can be easily improved.

By far the most prevalent source of error derived from imprecise, sliding note changes. Currently, a sliding attack will either be broken up into multiple pitches, be left unsegmented and its pitch will be averaged over the length of the segment, or it will produce a

correctly-pitched but late note that begins when the instrumentalist reaches the final pitch. In all three cases, despite a correct f_0 analysis, the detected note or notes will not be correct in both pitch and start time, and thus, will result in a lower score. Of the 33 sound files containing this type of error (25% of the total), 7 were computer synthesized excerpts containing *glissandos*, 1 was a violin excerpt with a less-obvious but nonetheless sliding note change, and 25 were vocal files with the usual, characteristic sliding note changes. Some of these errors were corrected thanks to vocal-specific parameter choice, allowing for a more stringent rd_{lim}. Others, such as multi-note *glissandos*, are not correctable under the current algorithm, as they are not adequately represented by the discretized 12 half note per octave grid imposed by current musical notation. An example of this second type of slide can be viewed in figure 9, in appendix B.

The second most-prevalent source of error was over-segmentation; 15 files, or 11% suffered from this type of error. A higher-than-desirable $harm_{th}$ or a lower-than-optimal rd_{lim} resulted in many more segments than notes. Although the f_0 estimation correctly detected a note, the application of $harm_{th}$ and rd_{lim} resulted in a segment that was shorter than the minimum segment length, and thus the segment was cast out and the note was left undetected. However, in noisy files or those containing background percussion, some of the correct segments cast out by the msl were indeed shorter than some of the segments due to percussion or noise (See figure 10, in appendix B). Thus the chosen mnl and the values for parameters involved in setting the harmonicity threshold and relative derivative limit represent a compromise between the detection of correct notes and the elimination of noise and percussion. Additionally, as the previous type of error calls for a more stringent rd_{lim} and over-segmentation calls for a more lenient rd_{lim}, it seems that the current parameter settings also attempt to compromise between the detection of gradual note changes and over-segmentation. In any case, the inclusion of a segment addition algorithm means that over-segmentation in the case of consecutive segments within a half-note range of each other are added up to form one note. As such, perhaps the true negative consequences of over-segmentation are not taken into account. If the segment addition algorithm were removed from the program, perhaps over-segmentation errors would become less prevalent. Moreover, it may be advantageous to vary the relative derivative threshold and the minimum segment and note lengths in a given sound file in relation to the harmonicity at each point in the file. Thus, one could allow more abrupt pitch changes and shorter notes and segments when harmonicity values were high, but segment fast pitch changes and discard short segments when harmonicity values were low. This would facilitate the elimination of noise without hindering the detection of short notes, and would allow portions of vibrato to remain contiguous segments which could then be easily averaged to find the target pitch. However, due to time constraints, these ideas were not explored.

The third type of error, occurring in 9 files, was the existence of incorrect notes resulting from poor f_0 estimates. In two of these files, the erroneous f_0s were subharmonics of the true notes; thus, a smaller window size and its corresponding higher minimum detectable fundamental frequency could have disqualified the subharmonic as a viable f_0 estimate and thus the true pitch would most likely be detected. For an example of this type of error, see figure 11, in appendix B. However, in the other 7, the window size could not be decreased without loss of the detection of some of the lowest notes in the file. Though incorrect f_0 estimates can sometimes be eliminated if they correspond with low harmonicity values or a fast variation in the f_0 curve, not all the errors of this type can be avoided in this fashion.

Thus, either other parameters of the $F0$ algorithm must be varied (such as the relative weights of the 3 scoring criteria), or $F0$ itself must be modified if all errors of this origin are to be avoided.

Finally, the window size choice algorithm was the source of error in 2 audio files, and a different window size choice could have helped improve results for 2 more. Viewed inversely, the window size choice algorithm chose an adequate window size allowing for the detection of all notes 98.4% of the time, and chose an ideal window size, minimizing the potential for subharmonic errors 97.0% of the time. This is certainly an achievement. However, for perfect results, the, most-likely noise-based causes of non-ideal window size choice would have to be taken into account in an improved version of the window size choice algorithm.

As for the voice-specific algorithm, it is clear that vibrato suppression and the application of two relative derivative limits do improve results. However, the minimum vibrato level, M_{value}, was set at the top of the range given to *Genadapt*, which indicates that the parameter was optimized when the fewest possible number of segments underwent vibrato suppression. If the upper limit of this range had been higher, perhaps results would have been even better without much vibrato suppression at all. This indicates that the vibrato suppression algorithm, as is, is suboptimal. The use of a minimum local extrema extent (N_{pts}), and a minimum extrema discrepancy ($dt_{min,ext}$), to deal with internal variation within vibrato segments in the f_0 curve is a rather imprecise way to achieve the goal of locating only the relevant local minima and maxima pertaining to the vibrato period. A better method might be to use a smoothing filter before calculating the local extrema.

5 Conclusion

The final product of this project is a note detection algorithm that takes a monophonic, uncompressed audio file as input and outputs either a midi file or a set of pitches and start times. Highlights of this program are its highly successful window size choice program, and its ability to work reasonably well for a large variety of instruments and to some extent, even in cases with background percussion, noise, and reverberation. Drawbacks of the program include its inability to detect repeated notes, and its weaker performance for voice than for other instruments. As vocal audio files were the most challenging to segment, an alternative algorithm was developed which included a method for vibrato suppression. Though this alternative algorithm did lead to marginal improvements in note detection, the vibrato suppression algorithm must be improved before automatic note detection in vocal excerpts can attain the same effectiveness as for other instrument types. The improved vibrato suppression algorithm should then be incorporated into the general note detection algorithm such that it represents a truly robust algorithm, performing adequately for all instruments. Alternatively, one could attempt to correlate the relative derivative threshold and minimum segment and note lengths with the harmonicity values at a given point in an audio file, which might facilitate vibrato suppression, noise elimination and detection of short notes. Additionally, the specific case of repeated notes should be studied; the segment addition process should be replaced by searches for repeated note indicators such as small dips in harmonicity and transitions between syllables. Finally, a highly reliable automatic tempo detection program must be incorporated, so that the set of detected notes can be translated into a musical score.

Acknowledgements

I'd like to thank IRCAM, the members of the Analysis/Synthesis group, and Axel Roebel, in particular, for his help and guidance throughout this project.

References

[1] Arturo Camacho. *SWIPE: A Sawtooth waveform inspired pitch estimator for speech and music*. PhD thesis, University of Florida, 2007.

[2] Matthias Krauledat. Fundamental frequency estimation. Master's thesis, Westfälische Wilhelmsuniversität Münster.

[3] S Rossignol, X Rodet, J Soumagne, Collette J.-L., and P. Depalle. Automatic characterization of musical signals: feature extraction and temporal segmentation. *Journal of New Music Reseach*, 28(4):281–295, 2006.

[4] Yohan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato. *STL-QPSR*, 35(2-3):045–068, 1994.

[5] Chunghsin Yeh, Axel Roebel, and Xavier Rodet. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech and Language Processing*, 18-6(2):1116–1126, 2010.

[6] M Zivanovic, Axel Roebel, and Xavier Rodet. Adaptive threshold determination for spectral peak classification. *Computer Music Journal*, 32:57–67, 2008.

Appendices

A Back-of-the-envelope calculation of the maximum relative derivative resulting from vibrato

The f_0 curve in a vibrato-containing segment is modeled as a simple sine function:

$$f_0(t) = Asin(2\pi f_v t)$$

where f_v is the vibrato rate, and A is the vibrato amplitude or extent (See figure 8). It's derivative, $\frac{df_0}{dt}$, is then:

$$\frac{df_0}{dt} = 2A\pi f_v cos(2\pi f_v t)$$

which has a maximum when $cos(2\pi f_v t) = 1$, with amplitude

$$max(\frac{df_0}{dt}) = 2A\pi f_v$$

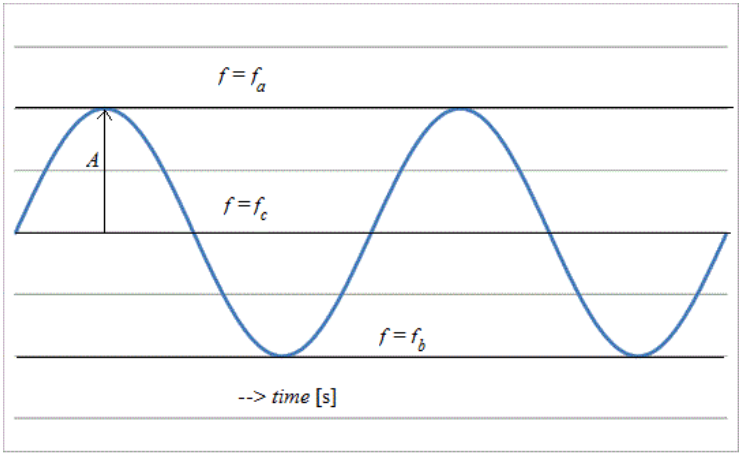

Figure 8: Idealized vibrato segment.

The maximum rd value is:

$$max(rd) = max(\frac{f_0[j+1] - f_0[j]}{dt \cdot (f_0[j] \cdot f_0[j+1])^{1/2}})$$

$$= max(\frac{f_0[j+1] - f_0[j]}{dt}) \cdot \frac{1}{min((f_0[j] \cdot f_0[j+1])^{1/2})}$$

$$\approx max(\frac{df_0}{dt}) \cdot \frac{1}{min((f_0[j] \cdot f_0[j+1])^{1/2})}$$

$$\approx 2A\pi f_v \cdot \frac{1}{min((f_0[j] \cdot f_0[j+1])^{1/2})}$$

Taking a maximal vibrato extent of ± 125 cents (slightly larger than the value cited in [4]), the maximum fundamental, f_a and the minimum fundamental, f_b, can be calculated in relation to the average pitch, f_c (see equation 2):

$$\frac{f_a}{f_c} = 2^{125/1200}, \quad \frac{f_b}{f_c} = 2^{-125/1200}$$

Rewriting the maximum amplitude in terms of these frequencies, A can be replaced by $f_a - f_c$. Additionally, since no point in the f_0 curve falls below f_b, $f_0[j] \cdot f_0[j+1]$ can be no smaller than f_b^2 Finally, a maximum frequency of $f_v = 10Hz$ is used, well above the usual 7 Hz upper limit for vocal vibrato. Substituting, the maximum rd value is can now be approximated:

$$max(rd) \approx \frac{2\pi(10)(f_a - f_c)}{f_b} \approx 2\pi(10)(\frac{f_a/f_c}{f_b/f_c} - \frac{f_c}{f_b}) \approx 5.55$$

B Examples of different error types.

Guide to understanding:

Each example is presented as a series of 5 graphs. The first graph is the f_0 estimation curve of the given audio file as it is outputted from $F0$. The second graph displays harmonicity

with respect to time; the red line represents the harmonicity threshold. The third graph is the relative derivative of the remaining portions after the application of the harmonicity threshold; the relative derivative limit is displayed in red. As the magnitude of the relative derivative is limited to below this value, both the positive and negative of the rd_{lim} are depicted. In the 4th graph, the remaining segments are displayed in blue; each note in the ground truth file and its corresponding half-note target range are depicted in black. The final notes detected by the automatic pathway are displayed in the final graph, in colors ranging from blue to red based on how closely they match the ground truth notes, again shown in black.

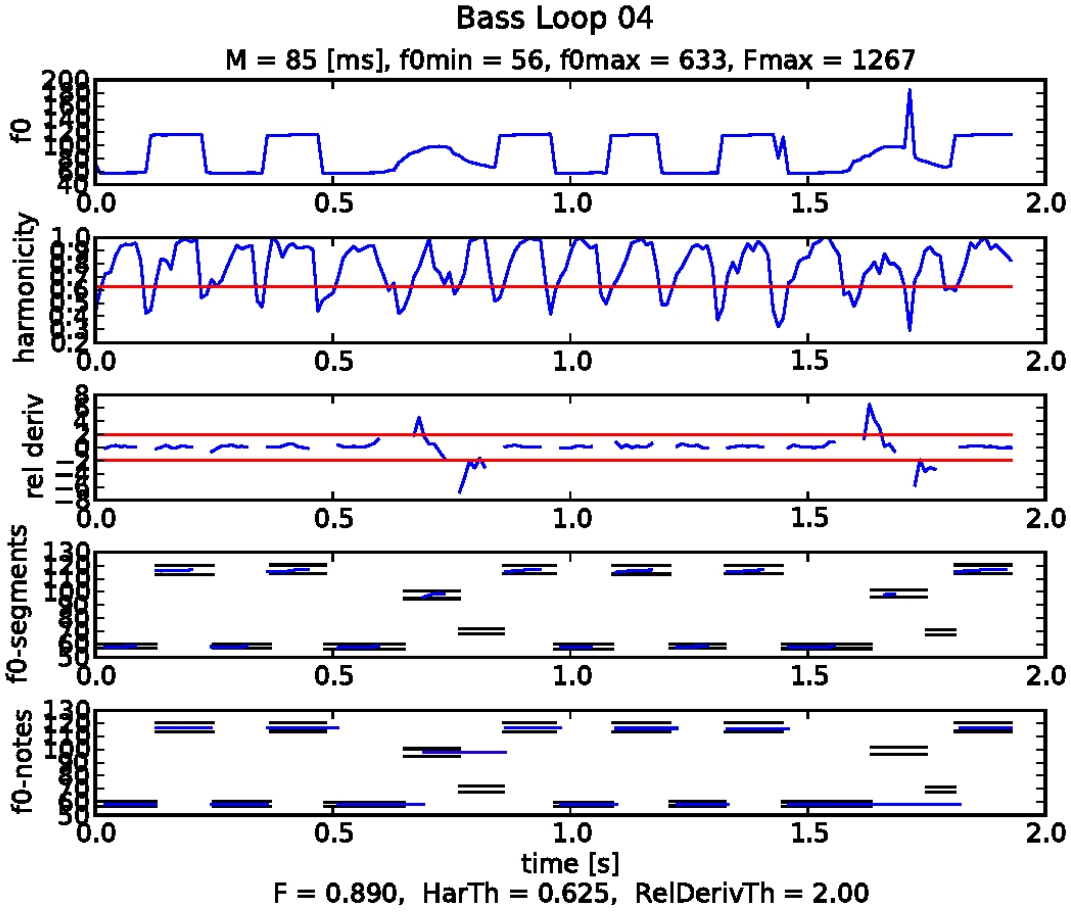

Figure 9: Example of an uncorrectable note slide.
In the top graph, the f_0 estimation curve of a computer synthesized bass loop is displayed. Note that at 0.65 s and again at 1.6 seconds, the f_0 arcs, depicting an upwards and downwards *glissando*. The harmonicity values corresponding to the *glissandos* are relatively high. However, the relative derivative (3rd graph) in these arcs (and especially in the second arc which is punctuated by an erroneous estimate around 1.75 s), is high enough to be segmented. As the notes in the ground truth file cannot correctly depict a *glissando*, final notes detected by the automatic pathway (final graph) during these instances are incorrect as well.

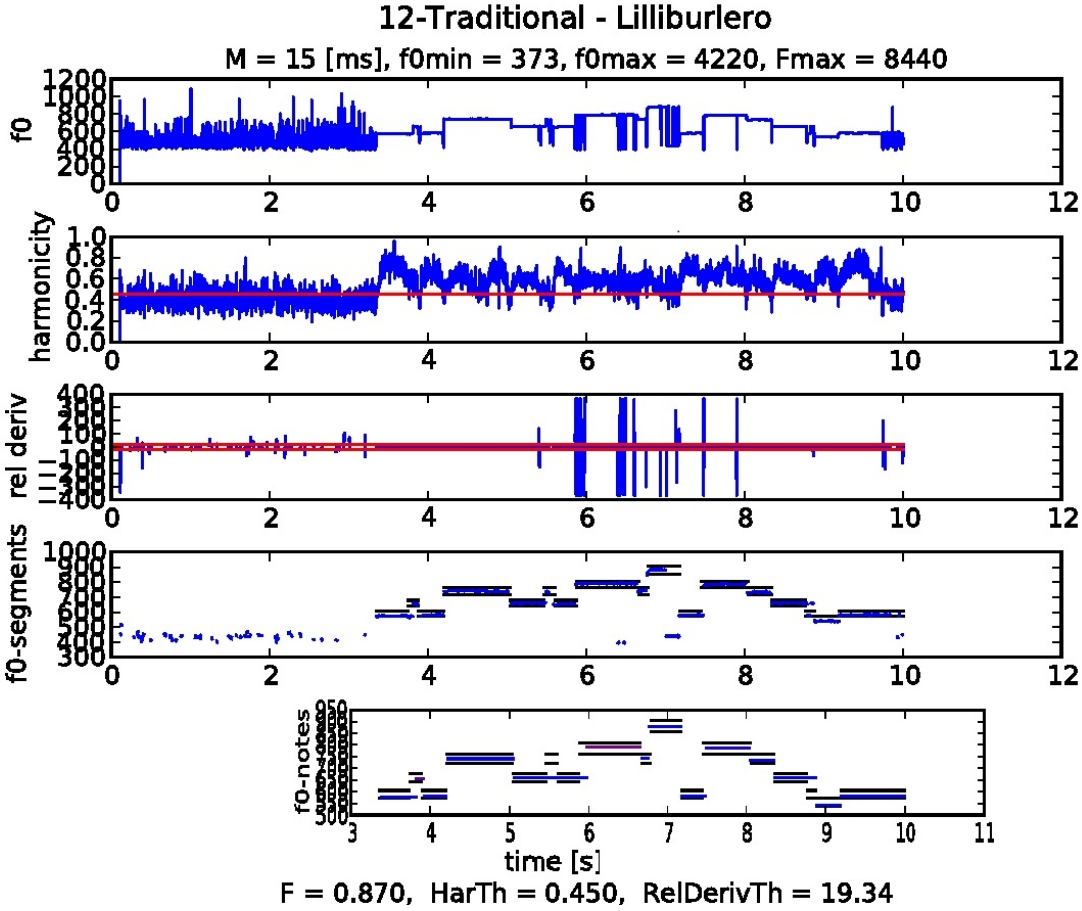

Figure 10: Example of trade-off between over-segmentation and noise elimination. The excerpt "Lilliburlero" begins with 3.3 seconds of timpani solo. At 3.3 seconds, the trumpet enters with the melody, but the timpani continues to play throughout the whole file. Note the noisy look to the f_0 curve in the first 3.3 seconds, and the corresponding lower harmonicity values. In the 4th graph, many short segments are found in the first 3.3 s of the file and also below the trumpet notes in the 6th, 7th and 10th second of the piece. These segments are a result of the timpani and not the trumpet melody; thanks to the minimum segment length parameter, they are all effectively eliminated. However, the 6th and 15th trumpet notes are detected in the segment depiction, but not in the final note version, because these segments were also shorter than the minimum note length. Loss of these two notes is the price to pay to be able to eliminate background percussion.

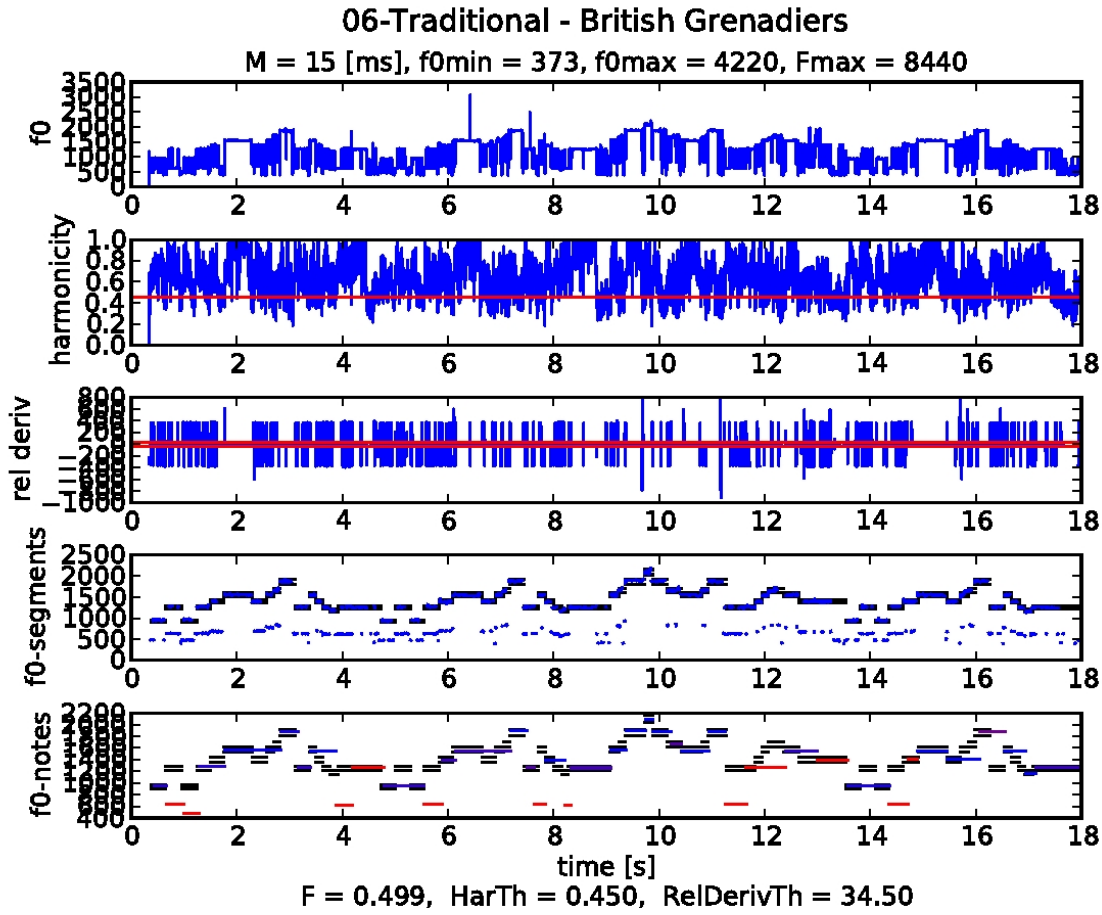

Figure 11: Example of a subharmonic errors that could be eliminated with a better window size choice.

In the top graph, the f_0 estimation curve of a flute excerpt is displayed. It appears very noisy, as many of the estimates alternate quickly between a higher and lower note. In the 4th graph, the segmented depiction of the curve, evidence of subharmonic f_0 estimation becomes clearer, as a set of incorrect segments seems to follow the true notes and correct segments, roughly one octave below. The final result is most definitely a sub par representation of the original file. Though this type of error can be attributed to poor f_0 estimation, if a smaller window size was chosen, these subharmonics would lie below the minimum detectable f_0 and thus could be eliminated.