Real-Time Performance Comparison of Tuning Frequency Estimation Algorithms

Alessio Degani, Marco Dalai, Riccardo Leonardi and Pierangelo Migliorati
DII, Signals and Communication Lab
University of Brescia
Email: {alessio.degani, marco.dalai, riccardo.leonardi, pierangelo.migliorati}@ing.unibs.it

***Abstract*—In this paper, a comparison of the real-time performances of different algorithms for the estimation of the concert pitch (tuning frequency or reference frequency) of music recordings is presented and discussed. The unavailability of ground-truth datasets makes this kind of evaluation on real music recordings less trivial than what it may initially appear. Hence, in this paper we investigate the algorithms best-case performances using simple generated sounds and than we study the estimation reliability and the real-time performances using real world recordings. In particular, we focus on the standard deviation of the estimation for various length of the signal, the reliability of the on-line estimation and the computational complexity.**

Keywords*—*Real-Time Estimation; Tuning Frequency; Concert Pitch

I. Introduction

In several music information retrieval tasks such as melody extraction, key or chord estimation and other pitch based feature extraction processes, a quantization of pitch frequency values to the equal-tempered scale is performed. Usually, the equal-tempered scale is expressed in a equally spaced integer values called *cents* (1) where an equal-tempered semitone interval corresponds to 100 cents. Although the frequency to cents mapping is in many cases supposed to be trivial, the precision of this task relies in the correct estimation of the tuning (reference) frequency f_{ref}. This conversion is in fact computed according the to the equation

$$c = 1200 \cdot \log_2 \left(\frac{f}{f_{ref}} \right), \qquad (1)$$

where f is the frequency value in Hz and c is the corresponding value in cents.

In some cases, a 440 Hz frequency for the reference (A4) pitch is chosen blindly. This assumption is justified by the fact that this tuning frequency is internationally standardized [1]. However, for timbre preference or due to instrumentation issues, the A4 pitch maybe different from 440 Hz. In this case, a wrong assumption or estimation of concert pitch might affect the overall performance of a music information retrieval system [2]. A reference frequency estimation algorithm may be useful also in a real-time audio processing or monitoring. For example, as shown in [11], a real-time estimation can be used to measure the pitch drift of a choral ensemble so that the singers can be able to take countermeasures. An on-line reference frequency estimation can be used as a feedback to automatically control the playing speed of a tape reel player or a turntable. The term real-time estimation means that a system must carry out an estimation of f_{ref} every fixed time interval. This time interval can be short as one analysis frame or may be longer (local analysis window) depending on the application.

The estimation of the tuning frequency usually constitutes a small pre-processing block (or post-processing in some cases) in the work-flow of larger systems and no specific evaluation of this block is available in the literature, to the best of the authors' knowledge. In [2] the authors discuss some musical signal characteristics that can negatively influence the performances of the tuning frequency estimation algorithms, such as, for example, the deviation of the odd harmonics of a complex tone from the equal-tempered scale. Here instead, we study how different algorithms work regardless of the structure of the analysed audio signal.

In this paper, we compare three methods of tuning frequency estimation that are based on the analysis of spectral peaks. In Section II we give a brief introduction of the studied algorithms. In Section III we explain the evaluation and comparison procedure, and in Section IV we introduce the two datasets used. Finally, the results are discussed in the Section V.

II. Tuning Frequency Estimation Methods

In this paper, we perform an evaluation of a specified class of estimation algorithms that shares a common pre-processing step. While other methods are presented in the literature (for example [2]), we focus our attention on those which requires a sequence of spectral peaks as inputs. The spectral peaks can be calculated from the spectrum (windowed FFT) in several ways [3]. For our evaluation purposes, we have used in this paper the peak picking algorithm developed in the context of the Sinusoidal Modelling Synthesis framework (SMS) [4] and presented in [5]. An estimation of each frequency location and peak amplitude is calculated by fitting peaks in the discrete spectrum with a parabola and using the vertex of the parabola as an estimation of the true non quantized peak. The output of this peak picking process is the common starting point shared by all of the three analysed tuning frequency estimation algorithms. All of the analysed audio pieces are mono wave files sampled at 22050 Hz. The FFT is calculated over a window of 8192 samples with an overlap of 75%, that leads to an *hop size* of approximately 93 msec. The peak picking algorithm returns 30 peaks in the range of $50 - 5000$ Hz for each analysis window as suggested in [6]. These 30

peaks are sorted from the highest to the lowest peak magnitude.

The first type of tuning frequency estimation method makes use of the pitch histogram (see [7] [8] [6] [9]), and we call it *Hist01*. Here, the width of the histogram bin is 1 cent. The second tested method is the one presented in [10], which is based on circular statistics (*Circ*). The last algorithm [11] uses a Least-Square optimization of the mapping error to the equal-tempered scale. From now on we call it *L-S*

We give now a brief description of these different approaches.

A. Pitch Histogram

The underlying idea of this type of algorithms is to build a histogram of the deviations of each spectral peak from the equal-tempered scale. The deviation in semitones for each peak frequency f_i can be calculated as follows

$$d_i = \frac{c_i - round(c_i)}{100}. \tag{2}$$

From (2), one can see that $d \in [-0.5, 0.5[$ due to rounding operation to the nearest integer. At this point, a histogram $\mathcal{H}$ of all deviations d_i is computed. Each peak deviation is weighted by its peak magnitude r_i to avoid high impact of small (noise) peaks. The overall estimated deviation $\hat{d}$ is the deviation associated to the histogram bin with the maximum value

$$\hat{d} = \arg\max(\mathcal{H}). \tag{3}$$

The reference frequency of the entire music piece can be computed as

$$f_{ref} = 440 \cdot 2^{\frac{\hat{d}}{12}}. \tag{4}$$

A fundamental parameter for this algorithm is the histogram resolution. Many of this kind of algorithms differ only in this parameter. In this paper we discuss only the resolutions of 1 cents.

B. Circular statistics

A different approach for tuning frequency estimation which makes use of circular statistics was presented in [10]. This approach is entirely based on the observation that the deviation d is a periodic measure and not an absolute measure, since it is a "wrapped around" quantity that should be evaluated from the nearest 100 cents grid point. Each cent value is mapped onto a unit circle 100 cents-periodic and represented as an unit modulus vector as follows

$$u = 1 \cdot e^{j\phi}, \tag{5}$$

where

$$\phi = \frac{2\pi}{100} \cdot c. \tag{6}$$

For each peak i with a frequency f_i, we consider the vector $u_i = r_i e^{\phi_i}$ where r_i is the peak amplitude and then we take the mean vector $\hat{u}$ of all circular quantities u_i as follows

$$\Re[\hat{u}] = \frac{\sum_{i=1}^{N} r_i \cos(\phi_i)}{\sum_{i=1}^{N} r_i} \tag{7}$$

$$\Im[\hat{u}] = \frac{\sum_{i=1}^{N} r_i \sin(\phi_i)}{\sum_{i=1}^{N} r_i}. \tag{8}$$

Each peak's ϕ_i is weighted by its peak magnitude r_i to avoid high impact of small (noise) peaks. The overall deviation is then compued from the angle of the resulting vector $\hat{u}$, that is

$$\hat{d} = \frac{1}{2\pi} \arg(\hat{u}). \tag{9}$$

The tuning frequency can be finally estimated using (4).

C. Least-Square Estimation

This algorithm uses a Least-Square optimization approach and is presented in [11]. The aim of this method is to estimate the reference frequency in real-time, in order to visualize the evolution of the tuning frequency of a choir ensemble. In this context, the author of this algorithm has developed an application that runs on a modern *smartphones*.
As the choir changes the tuning frequency during singing, the conductor, with the aid of this application is able to exploit this variation and can take some countermeasures to correct the pitch drift.

In a nutshell, this method, at each analysis frame, calculates an equal-tempered frequency scale values using the previous estimation of f_{ref} and updates this estimation minimizing the average squared error when mapping each peak frequency f_i to the "new" frequency scale.

At first, a deviation in integer semitone index is calculated at each peak frequency f_i as

$$s_i = round\left[12 \cdot \log_2\left(\frac{f_i}{f_{ref}}\right)\right], \tag{10}$$

then the new reference frequency is estimated in a Least-Square sense using

$$f_{ref}^{new} = \frac{\sum_i (f_i \cdot 2^{\frac{s_i}{12}})}{\sum_i 2^{\frac{s_i}{6}}}. \tag{11}$$

This method can estimate a f_{ref} that goes out of the bounds $f_{ref} \pm 50$ cents. For this reason, the authors of the *L-S* algorithm takes some countermeasures in order to prevent this issue. First, a *Reset* button is present in the real-time application. Second, the current estimated f_{ref} is stored a circular buffer of B estimations, and the new estimated f_{ref} is taken as the median of this buffer. In our experiments, we choose $B = 20$, and if the estimation exceeds the limit of $f_{ref} \pm 50$ cents, we force $f_{ref}^{new} = 440$ Hz simulating the *reset* button. Further details of this method can be found in [11].

III. Evaluation Strategy

In the literature, some authors use synthesized tones (for example sinusoids or sawtooth) in order to make an evaluation of the precision [11]. That can be useful to test if a given algorithm is sufficiently precise in an ideal case. However, this kind of signals do not give a fair representation of the real world recordings and not all of the characteristics of the tested algorithm can be exploited. Even symbolic music data such as *MIDI* files or *MusicXML* sheets synthesized by a sequencer, do not provide a valid ground-truth. This is because the sound banks used by a sequencer (called *SoundFonts*) may exhibit an unpredictable detuning. However, synthesized symbolic music provides a good starting point because we can make the assumption that all of the synthesized songs shares the same "detuning behaviour".

One dataset of synthesized symbolic music (MS2012) is considered and discussed in Sec. IV. Furthermore, a test using a generated sawtooth sweep is performed in order to test the precision of the algorithm for the best-case scenario (no noise, only one pitch with known frequency). The real-time test are made using some real world recording.

A. Estimation reliability

In this test we study the reliability of the estimations of the algorithms.

First, a test with a simple synthetic sound is performed in order to verify the best-case accuracy of the algorithms. The synthetic sound is a sawtooth sweep defined as

$$s(t) = \frac{1}{H}\sum_{h=1}^{H}\frac{1}{h}\sin[2\pi h(f_0 t + \alpha t^2)]. \tag{12}$$

For a 5 sec. signal sampled at 44100 Hz and a base frequency $f_0 = 440$ Hz, we set $\alpha = 5$ and the number of harmonics to $H = 45$ in order to obtain an aliasing-free test signal. An estimation of f_{ref} is calculated for each analysis frame.

Second, using the dataset described in Sec. IV, we study the estimation using a varying quantity of analysed signal. This evaluations points out the ability of the methods to give a consistent estimation using few data. The real-time constraints forces the tuning frequency estimation algorithms to give a value of f_{ref} using a little portion of data (for example an audio buffer). If an algorithm gives a consistent estimation using few data it means that it can be used for a reliable real-time estimation. The reliability of the reference tuning estimation is analysed under the hypothesis that for a given song, the f_{ref} remains constant (global f_{ref}) over the entire music piece. Reasonably, a *trusted* algorithm must carry out the same f_{ref} estimation regardless of which part of the song is examining. In more detail, we analyse the standard deviation of the estimation for all the songs in the dataset, making $N = 50$ estimation for each music piece using $p\%$ of the song picked randomly. Let $f_{ref_s}^{n,p}$ be the estimated reference frequency using the $p\%$ of the frames of the song s at the extraction $n \in [1 \dots N]$. The standard deviation σ_p is calculated using

$$\sigma_p = \sqrt{\frac{\sum_{s=1}^{S}\sum_{n=1}^{N}\left(f_{ref_s}^{n,p} - f_{ref_s}^{global}\right)^2}{SN}}. \tag{13}$$

The test of σ_p is performed using both $k = 30$ and $k = 5$ peaks per frame.

TABLE I: Minimum, Maximum, Average and Total length of the songs in the datasets in HH:MM:SS format

Dataset	min	max	average	total
MS2012	00:00:08	00:25:34	00:02:46	14:06:16

B. Local tuning estimation

In the case which the hypothesis of $f_{ref} = const$ for a given song are not satisfied (for example in the choral or a-cappella exhibitions where a pitch drift is present, or in the tape or vinyl recording with a non-constant motor rotation of the playing gear), we need another kind of test. For this purpose we simulate a real-time reference frequency estimation where a value of f_{ref} is carried out each 4 or 2 seconds during the reproduction of the musical piece. In more detail, we give a f_{ref} estimation every local analysis window that groups $L_{wnd} = 80$ or $L_{wnd} = 40$ analysis frames. Each local window has a 50% of overlap with the adjacent windows. With this test we can observe the variation of the tuning frequency along the time axis of a musical piece.

C. Computation time

Since the test script is written in Matlab code, we use the *Matlab Profiler* to calculate the execution time of the algorithms. The computational complexity of the various implementations of the algorithms may be different using other programming languages (for example C/C++, ...), especially for the *Hist* algorithm. For this reason, our measured execution time may vary from an implementation to another. However, since the *Circ* and *L-S* methods consist only in a algebraic calculations (see (9) and (11)), a further optimization in the computational complexity are not required.

IV. Data Set

In our evaluation process we use a synthesized symbolic music dataset (MS2012) for testing the estimation reliability and computation time. Three real world recordings are considered for investigate the local tuning estimation behaviour. Table I shows some statistics about the length of the musical pieces in the MS2012 dataset.

The following sections illustrates some useful details of the dataset.

A. MuseScore Symbolic Music Dataset (MS2012)

This dataset in a collection of 306 songs in *MIDI* and *MusicXML* format. All the songs in this dataset are distributed under *free-to-share* Creative Commons CC0 license and are

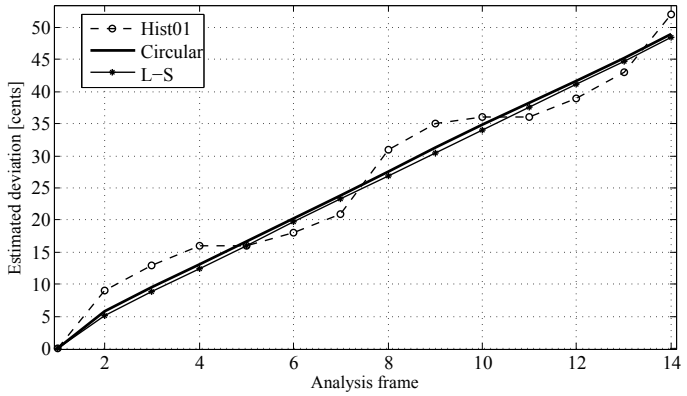

Fig. 1: Frame by frame f_{ref} estimation of a sawtooth sweep signal.

kindly provided by **MuseScore**[1]. The complete dataset with some user generated metadata is freely available at [13]. For the evaluation test we have to synthesize symbolic data to a 22050 Hz, 16 bit mono PCM wave file. For this task we have used an open source software synthesizer named **FluidSynth**[2] with the soundfont named *FluidR3*.
In Table II the most used musical instruments (using the GM standard nomenclature) in the dataset are reported. The values represent the percentages of songs in which an instrument is used in one or more track of the song.

TABLE II: Instruments (GM names)

Instrument	%
Acoustic Grand Piano	26%
String Ensemble 1	19%
Choir Aahs	16%
Flute	15%
Clarinet	14%
French Horn	13%
Trumpet	12%
Trombone	11%
Tuba	10%
Violin	10%
Alto Sax	10%
Acoustic Guitar nylon	9%

V. Results

A. Estimation reliability

As shown in Fig. 1, all the algorithms follow the pitch drift of the sawtooth sweep signal. Although the *Circ* and *L-S* shows a very close behaviour, a little "delay" on the estimation with *L-S* can be seen in this test due to the memory of the system that takes the previous estimation as a starting point for the actual estimation. We can also notice the effect of the quantization in the histogram for the *Hist01* algorithm.

The estimation reliability of each algorithm is measured using the standard deviation σ_p as defined in (13). For each song we calculate σ_p over $N = 50$ estimation using $p\%$ of

[1]http://www.musescore.com
[2]http://www.fluidsynth.org/

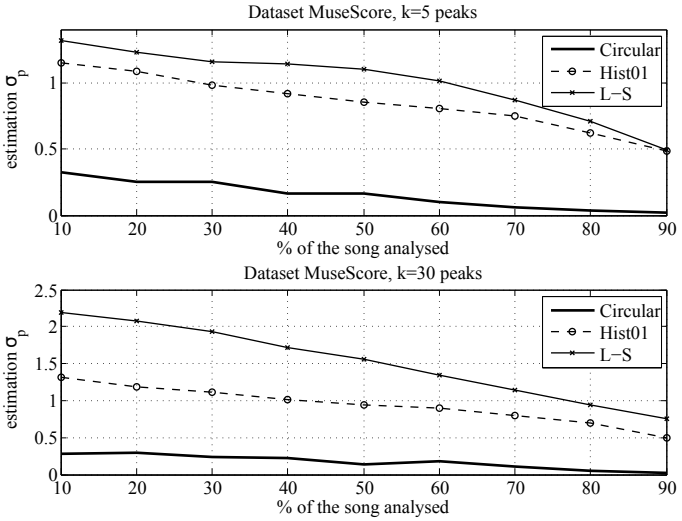

Fig. 2: Reliability of the algorithms on the MS2012 dataset.

the song's analysis frame. In Fig. 2 the results of this test with $k = 5$ and 30 peaks per frame are reported.

As we can see in Fig. 2, all of the algorithms have a small σ_p but only the *Circ* algorithms gives a reliable estimation with the MS2012 dataset with small amount of data. Since the minimum audible pitch difference is $3 - 4$ cents [14] and in the neighbourhood of 440 Hz an interval of ± 4 cents means a difference of about ± 1 Hz, we can see that only the *Circ* algorithm guarantees (on average) an inaudible estimation error even using a very little portion of of the song.

Another fact is that the estimation using only $k = 5$ peaks is more reliable with respect to the estimation using $k = 30$ peaks, especially for the *L-S* algorithm. Using few peaks, whereas they are ranked using their magnitude, the first considered peaks are the most salient spectral peaks. The *L-S* algorithm is the only one that does not take into account the peaks magnitude, hence it treats all peaks with the same weight. In this way, also the small magnitude noise peaks have the same impact of the true spectral peaks. For that reason, the *L-S* method works better when only the first prominent peaks are considered in the estimation task.

However, the result of *L-S* can be misleading. Although the deviation remains small, the Fig. 2 show that the standard deviation of the estimation of the *L-S* is bigger than the other algorithms. This fact suggest that the *Circ* is far more reliable than *L-S* and *Hist01*. As we will see in the following Section, this consideration is true for the *Hist01* but not for the *L-S*. In fact, the *L-S* algorithm is not well suited for a global f_{ref} estimation because the only link with the previous frames estimations is the f_{ref} estimated at the time instant immediately preceding the actual estimation. For that reason, only the last portion of the data contributes effectively to the estimation of the f_{ref}. As an example, if a song with a constant f_{ref} exhibit a detuning in the last seconds of the songs, the global f_{ref} estimation using *L-S* can be dramatically wrong. In the next Section we will show that the *L-S* is well suited to the real-time f_{ref} estimation.

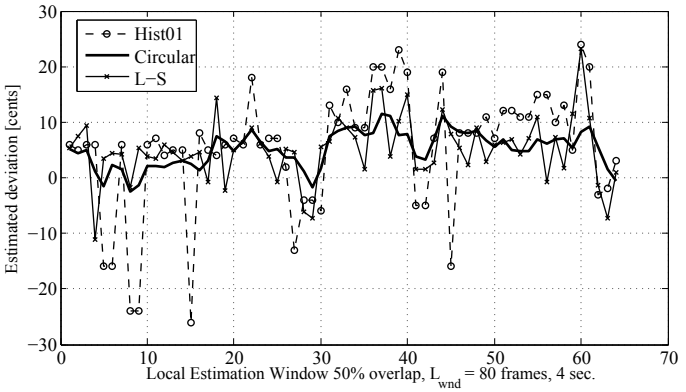

Fig. 3: Local tuning estimation of the song "Let It Be" played by The Beatles

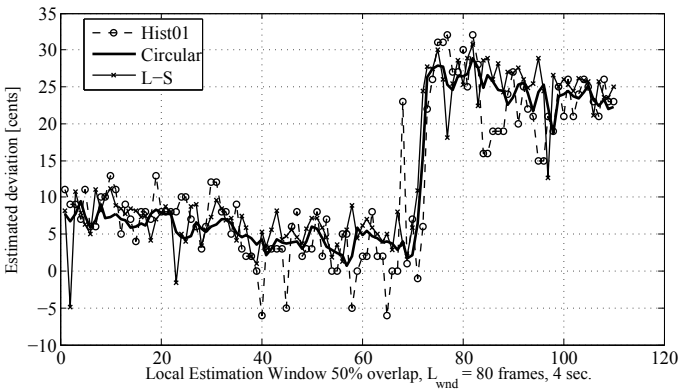

Fig. 4: Local tuning estimation of "Variations 16-20" in J.S. Bach, Goldberg Variations, BWV 988, played by Wanda Landowska, Paris (1933), CD version of 78 rpm recording

B. Local tuning estimation results

For the local tuning estimation performances evaluation, each algorithm is "forced" to give a f_{ref} estimation every L_{wnd} analysis frame simulating a real-time behaviour. For the firsts two testing songs, we use a local analysis window of length $L_{wnd} = 80$ frames. This means that an estimation is carried out every 4 seconds. In this test we measure the deviation in cents between the estimate tuning frequency and a reference of 440 Hz.

In Figure 3 and 4 the local estimation performances of the algorithms is reported. In order to get a more stable reference frequency estimation, we use $k = 5$ peaks per frame. In more detail, in Figure 3 we show the local estimation results for a constant reference tuning song *Let It Be* performed by *The Beatles* that we assume to have a constant f_{ref}, while in Figure 4 we show how the algorithms behaves when a pitch drift occurs during the reproduction. This particular recording comes from a remastered CD version of an old (1933) 78 rpm vinyl recording of classical music.
In both cases, the *Circ* method gives a less "shaky" estimation with respect to the others.

The *L-S* algorithm seems to have a more stable estimation compared with the *Hist01*. In this case, the *L-S* method

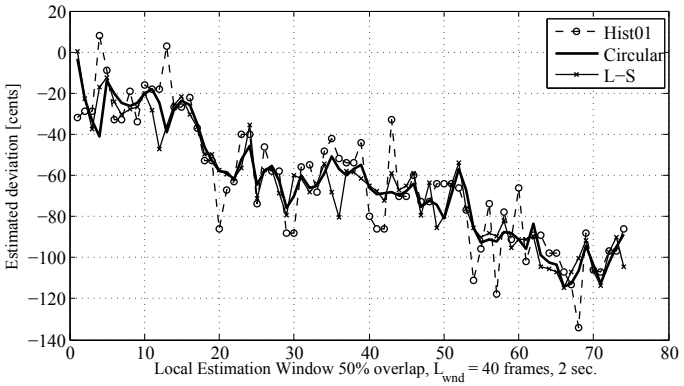

Fig. 5: Local tuning estimation of Choir performance used in [11]

TABLE III: Average execution time per song

Algorithm	Average Time (s)
Circular	0.19
Hist01	2.63
L-S	1.15

can exploit better his "recursive" attitude since the previous estimation is the starting point for the next f_{ref} evaluation in order to make a consistent estimation.

In Figure 5, we show the results of the estimation with the same choir ensembles recording used in the evaluation test in [11] and setting $L_{wnd} = 40$ frames (2 seconds) in order to use roughly the same window length used for the tests in [11]. The performance suffers of a gradual pitch falling that exceeds one semitone from the starting tuning frequency. In that case, the *Circ* and *L-S* algorithms perform quite well and shows a similar behaviour. For this reason, this two algorithms are more suitable for the real-time frequency estimation than the *Hist01*.

C. Execution Time

The last evaluation test considers the computation time required for each algorithm. This test was performed using the Matlab Profiler utility on the f_{ref}^{global} estimation of all the 3060 songs in the MS2012 dataset with $k = 30$ peaks per frame. Only a relative evaluation is given here since the algorithms optimization in terms of computation efficiency is not the goal of this paper. We think that a further improvement on the absolute execution time may be possible, for example, using a C/C++ implementation of the methods. The average execution time in seconds are reported in Table III. The test are made on a 64 bit GNU/Linux equipped laptop, with 4 GB of RAM and an Intel i5 M 430 CPU at 2.27 GHz without parallel Matlab optimization.

As we can see in Table III, all of the algorithms are fast enough to run also in real-time. The *L-S* has been designed to be real-time even in a computationally limited environment such as modern *smartphones*. In our test, however, the best performing algorithm in terms of computational cost is the *Circ*.

VI. Conclusions

The lack of scientific ground-truth makes the evaluation of the precision of the tuning frequency estimation algorithms a non trivial task. However, under certain hypothesis as shown in Sec. III, we were able to study the behaviour of the various tested algorithms. As mentioned in the Sec. V-C, from the point of view of the computational complexity, all of the studied algorithms can satisfy the real-time constraint. Nevertheless, as seen in Sec. V-A, only *Circ* algorithm gives a reliable estimation using few data. Furthermore, the presence of a median filter on a data buffer and the dependency of the result from the previous estimation, introduces a sort of delay in the *L-S* estimation.

The number of peaks per frame k is not a fundamental parameter for the *Circ* and *Hist01* methods, while the *L-S* algorithm can benefit from using few peaks. However, in the computational cost optimization point of view, using few peaks means less computation time. In our tests we demonstrate that $k = 5$ peaks per frame are sufficient for all of estimation task presented here. Moreover, our tests show that the *Circ* algorithm outperforms the other ones in terms of reliability. For the real-time adaptability to reference frequency changes the *Circ* and the *L-S* shows comparable results and both are well suited for this task.

Furthermore, our tests show that *L-S* gives worst results for the global f_{ref} estimation compared against the other two algorithms.

References

[1] ISO, "Acoustics - standard tuning frequency (standard musica pitch)," *ISO*, vol. 16, no. 1975, 1975.

[2] A. Lerch, "On the requirement of automatic tuning frequency estimation," *Proc. of the 7th Int. Conf. on Music Information Retrieval (ISMIR)*, pp. 212–215, 2006.

[3] K. Dressler, "Sinusoidal extraction using an efficient implementation of a multi-resolution fft," *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx)*, 2006.

[4] A. L. X. Amatriain, J. Bonada and X. Serra, *in Udo Zölzer DAFX-Digital Audio Effects*. John Wiley & Sons, 2002, ch. Spectral processing, pp. 373–438.

[5] X. Serra, "Musical sound modeling with sinusoids plus noise," in *Musical Signal Processing*, A. P. C. Roads, S. Pope and G. D. Poli, Eds. Swets & Zeitlinger Publishers, 1997, ch. Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122.

[6] E. Gómez, "Tonal description of music audio signals," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[7] M. Ryynänen, "Probabilistic modelling of note events in the transcription of monophonic melodies," Master's thesis, Tampere University of Technology, 2004.

[8] C. Harte and M. Sandler, "Automatic chord identifcation using a quantised chromagram," *Audio Engineering Society Convention 118*, vol. 5, 2005.

[9] S. G. Yongwei Zhu, Mohan S. Kankanhalli, "Music key detection for musical audio," *Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)*, pp. 30–37, 2005.

[10] K. Dressler and S. Streich, "Tuning frequency estimation using circular statistics," *Proc. of the 8th Int. Conf. on Music Information Retrieval, (ISMIR 2007)*, pp. 357–360, 2007.

[11] J. B. Volker Gnann, Markus Kitza and M. Spiertz, "Least-squares local tuning frequency estimation for choir music," *131st AES Convention*, 2011.

[12] D. P. W. Ellis, "The 'covers80' cover song data set," *URL: http://labrosa.ee.columbia.edu/ projects/coversongs/covers80/*, 2007.

[13] A. Degani, "The 'ms2012' symbolic music dataset," *URL: http://www.ing.unibs.it/alessio.degani/?p=ms2012*, 2012.

[14] E. Zwicker, *Psychoacoustics: Facts and Models*. Springer, 2006.