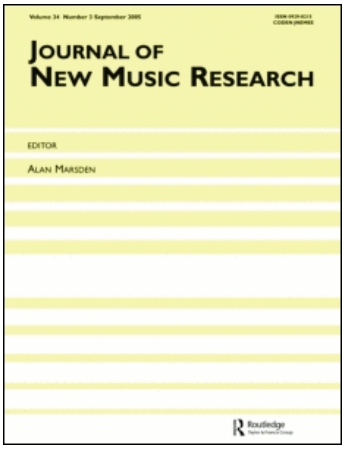


**Journal of New Music Research**
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713817838

# From Pitches to Notes: Creation and Segmentation of Pitch Tracks for Melody Detection in Polyphonic Audio

Rui Pedro Paiva[a]; Teresa Mendes[a]; Amílcar Cardoso[a]
[a] University of Coimbra, Portugal

**To cite this Article** Paiva, Rui Pedro , Mendes, Teresa and Cardoso, Amílcar(2008) 'From Pitches to Notes: Creation and Segmentation of Pitch Tracks for Melody Detection in Polyphonic Audio', Journal of New Music Research, 37: 3, 185 — 205
**To link to this Article: DOI:** 10.1080/09298210802549748
**URL:** http://dx.doi.org/10.1080/09298210802549748

Journal of New Music Research
2008, Vol. 37, No. 3, pp. 185–205

Routledge
Taylor & Francis Group

# From Pitches to Notes: Creation and Segmentation of Pitch Tracks for Melody Detection in Polyphonic Audio

Rui Pedro Paiva, Teresa Mendes, and Amílcar Cardoso

University of Coimbra, Portugal



## Abstract

Despite the importance of the note as the basic representational symbol in Western music notation, the explicit and accurate recognition of musical notes has been a difficult problem in automatic music transcription research. In fact, most approaches disregard the importance of notes as musicological units having dynamic nature.

In this paper we propose a mechanism for quantizing the temporal sequences of the detected fundamental frequencies into note symbols, characterized by precise temporal boundaries and note pitches (namely, MIDI note numbers). The developed method aims to cope with typical dynamics and performing styles such as vibrato, glissando or legato.

## 1. Introduction

Melody detection in polyphonic audio is a research topic of increasing interest. It has a broad range of applications in fields such us Music Information Retrieval (MIR, particularly in query-by-humming in audio databases), automatic melody transcription, performance and expressiveness analysis, extraction of melodic descriptors for music content metadata, plagiarism detection, to name but a few. This is all the more relevant nowadays, as digital music archives are continuously expanding. The current state of affairs places new challenges on music librarians and service providers, regarding the organization of large-scale music databases and the development of meaningful ways of interaction and retrieval.

Several applications of melody detection, namely melody transcription, query-by-melody or motivic analysis, require the explicit identification of musical notes, which allow for the extraction of higher-level features that are musicologically more meaningful than the ones obtained from low-level pitches.[1]

Despite the importance of the note as the basic representational symbol in Western music notation, the explicit and accurate recognition of musical notes is somewhat overlooked in automatic music transcription research. In fact, most approaches disregard the importance of notes as musicological units having dynamic nature.

Therefore, in this paper we propose a mechanism for quantizing the temporal sequences of the detected fundamental frequencies into note symbols, characterized by precise temporal boundaries and note pitches (namely, MIDI note numbers). The developed method aims to cope with typical dynamics and performing styles such as vibrato, glissando or legato.

The accomplished results, despite showing that there is room for improvement, are positive. The main difficulties of the algorithm are found on the segmentation of pitch tracks with extreme vibrato, such as in opera pieces, and on the accurate segmentation of consecutive notes at the same pitch.

[1]For language convenience, we will use the term pitch indistinctly of fundamental frequency (F0) throughout this article, though the former is a perceptual variable, whereas the latter is a physical one. This "abuse" appears in most of the related literature and, for the purposes of the present research work, no ambiguities arise from it.

*Correspondence*: Rui Pedro Paiva, CISUC – Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, University of Coimbra – Pólo II, 3030 Coimbra, Portugal. E-mail: ruipedro@dei.uc.pt

DOI: 10.1080/09298210802549748 

The paper is organized as follows. In this section we introduce the main motivations for explicitly determining musical notes, as well as other work related to the subject. Section 2 offers a brief overview of the main modules of our melody detection approach. The second module, determination of musical notes, is the main topic of this article and is addressed in Section 3. In Section 4, we describe the experimental setup and analyse the obtained results. Finally, in Section 5, we end up with a summary of conclusions and directions for future work.

### 1.1 The note as a basic representational symbol

The note is usually regarded as the fundamental building block of Western music notation. When characterizing a musical note (for example in a written score), features such as *pitch*, *intensity*, *rhythm* (typically representing accents and timing information, e.g. duration, onset and ending time), *performance dynamics* (glissando, legato, vibrato, tremolo, etc.) and sometimes even *timbre* are considered. Hence, in this respect, the goal of any automatic transcription system would be to capture all this information.

While the note is central in Western music notation, it is not evident if the same applies when we talk about perception. In reality, some researchers defend that, instead of notes, humans extract auditory cues that are then grouped into *percepts*, i.e. brain images of the acoustical elements present in a sound. Eric Scheirer argues that "most stages of music perception have nothing to do with notes for most listeners" (Scheirer, 2000, p. 69). In fact, he adds, "the acoustic signal must always be considered the fundamental basis of music perception", since "[it] is a much better starting point than a notation invented to serve an entirely different mode of thought" (Scheirer, 2000, p. 68).

Namely, tonally fused sounds seem to play an important role in music perception (Scheirer, 2000, p. 30). For example, the sounds produced by pipe organs perceptually fuse into one single percept, i.e. the various concurrent sounds are unconsciously perceived as a whole. Thus, trying to explicitly extract the individual musical notes that are enclosed in a tonally fused sonic object seems perceptually unnatural.

Nevertheless, we could also argue that notes are indeed perceived in some situations, for instance while listening to monophonic melodies. In such cases, the average listener easily memorizes them and replicates what he hears, for example by humming or whistling. In addition, he can even try to mimic the timbre of the singer, as well as some of the performance dynamics. In other words, his mental constructs seem to correspond to musical notes, although he may or may not be aware of that.

It is also important to take into consideration that there is some debate on whether or not vibrato, glissando, legato and other performing styles should be represented as quantized notes, mainly in contexts that are bound to introduce some errors, as in vocal melodies. As a matter of fact, in the melody extraction track of the Music Information Retrieval Evaluation eXchange – MIREX'2005 and 2006 – the objective is to identify the sequence of pitches that bear the main melody, i.e. a raw pitch track not represented by flat MIDI notes. On the other hand, our aim is to obtain a set of quantized notes, in the same way as human transcribers do, regardless of the instrument used (with or without significant frequency modulation) or style of the performer (more or less vibrato, legato, etc.).

Regardless of the arguments that can be presented to either support or reject the note as a perceptual construct, the identification of musical notes is essential in music transcription, in order for a symbolic representation to be derived. Furthermore, in other applications such as query-by-humming or melody similarity analysis, it usually needs musical notes rather than pitch tracks. As a result, in our work we consider musical notes as the basic building blocks of music transcription and, therefore, investigate mechanisms to efficient and accurately identify them in musical ensembles.

### 1.2 Related work

The identification of musical notes is somewhat overlooked in the field of automatic music transcription. Regarding the particular melody transcription problem, this is confirmed by the absence of a note-oriented metrics in the audio melody extraction track of the Music Information Retrieval Evaluation eXchange – MIREX'2005 and 2006.

Past work in the field addressed especially the extraction of pitch lines, without explicit determination of notes, or using ad hoc algorithms for the segmentation of pitch tracks into notes (e.g. segment as soon as MIDI note numbers change). This has turn out to be difficult for some signals, particularly for singing (Klapuri, 2004, p. 3). In fact, the presence of glissando, legato, vibrato or tremolo makes it sometimes a challenging task. Yet, amplitude and frequency modulation are important aspects to consider when segmenting notes.

Different kinds of methodologies for note determination, e.g. note segmentation and labelling, are summarized in the following paragraphs.

#### *1.2.1 Note segmentation*

*1.2.1.1 Amplitude-based segmentation.* In monophonic contexts, note segmentation is typically accomplished directly on the temporal signal. In fact, since no simultaneous notes occur, several systems first implement signal segmentation and then assign a pitch to each of the obtained segments, e.g. (Chai, 2001, p. 48). In this

strategy, silence detection is frequently exploited, as this is a good indicator of note beginnings and endings. In algorithmic terms, silences correspond to time regions where the amplitude of the signal (the root mean square energy is generally used) falls below a given threshold. The robustness of these methods is usually improved by employing adaptive thresholds (McNab et al., 1996b; Chai, 2001).

The main limitations of employing only amplitude-based segmentation come from the difficulties in accurately defining amplitude thresholds (particularly in polyphonic contexts, where sources interfere severely with each other). This may give rise to both excessive and missing segmentation points, leading to the unsuccessful separation of notes played legato. Moreover, in a polyphonic context several notes may occur at the same time, with various overlapping patterns. Consequently, note segmentation cannot be performed neither before nor independently of pitch detection and tracking.

*1.2.1.2 Frequency-based segmentation*. Frequency variations are usually better indicators of note boundaries, especially in polyphonic contexts. Here, frame-wise pitch detection is first conducted and then pitch changes between consecutive frames are used to segment notes. To this end, frequency proximity thresholds are normally employed (e.g. McNab et al., 1996b).

However, several of the developed systems do not adequately handle note dynamics. This is frequently the case in transcription systems dedicated to specific instruments such as piano, which do not modulate substantially in pitch (e.g. Hawley, 1993).

In Martins (2001), pitch trajectories are created with recourse to a maximum frequency distance of half a semitone. Nevertheless, smooth frequency transitions between notes might lead to trajectories with more than one note. This was not attended to apparently because most of the used excerpts came from MIDI-synthesized instruments played without note legato.

Keith Martin bases the identification of musical notes on the continuation of pitches across frames and on the detection of onsets. This information is combined and analysed in a blackboard framework (Martin, 1996). The used frequency proximity criteria are not described but, apparently, note hypotheses may contain more than a single note in the case of smooth pitch transitions. The provided examples are not conclusive since tests were implemented with piano sounds only, characterized by having sharp onsets and not modulating significantly in frequency.

The problem of trajectories containing notes of different pitches was addressed in Eggink and Brown (2004). There, the frequency distance is computed based on an average of the past few F0 values. The authors argue that this allows for vibrato while breaking up successive tones even when they are separated by only a small interval. However, even in this situation, it is not guaranteed that individual tracks will contain one single note. Indeed, depending on the defined threshold, smooth frequency transitions between consecutive notes could still be kept in a single track, as we have experimentally confirmed. In this situation, the frequency values in the transition may not differ considerably from the average of the previous values. In other situations, the two notes could be segmented somewhere during the transition, rather than at its beginning. Also, the use of a small interval is not robust to missing pitches in tracks containing vibrato, which could generate abrupt frequency jumps.

In brief, the main drawback of the previous methodologies is that the balance between over and under-segmentation is often difficult: if small frequency intervals are defined, the frequency variations in fast glissando or vibrato zones might be erroneously separated into several notes; on the other hand, if larger intervals are permitted, a single segment may contain more than one note.

*1.2.1.3 Probabilistic frameworks for frequency-based segmentation*. Some of the weaknesses described above are tackled under probabilistic frameworks. Namely, Timo Viitaniemi et al. (2003) employ a probabilistic model for converting pitch tracks from monophonic singing excerpts into a discrete musical notation (i.e. a MIDI stream). The used pitch-trajectory model is a Hidden-Markov Model (HMM) whose states correspond to MIDI note numbers, where an acoustic database is utilized to estimate the observation probability distribution. In addition, a musicological model estimates the key signature from the obtained pitch track, which is used to give information on the probability of note occurrence. Finally, inter-state transition probabilities are estimated based on a folk song database and a durational model is used to adjust state self-transition probabilities according to the tempo of the song (known *a priori*). The output of the HMM is the most likely sequence of discrete note numbers, which (ideally) copes with both pitch and performing errors. Note boundaries then directly denote transitions of MIDI numbers. Moreover, note durations are adjusted recurring to tempo information.

Ryynänen and Klapuri (2005) handle note segmentation in the context of a polyphonic transcription system. The overall strategy is very elegant and apparently robust. There, two probabilistic models are used: a note event model, used to represent note candidates, and a musicological model, which controls the transitions between note candidates by using key estimation and computing the likelihoods of note sequences. In the note event model, a three-state HMM is allocated to each MIDI note number in each frame. The states in the model represent the temporal regions of note events,

comprising namely an attack, a sustain and a noise state, and therefore taking into consideration the dynamic properties and peculiarities of musical performances. State observation likelihoods are determined with recourse to features such as the pitch difference between the measured F0 and the nominal pitch of the modelled note, pitch salience and onset strength. The observation likelihood distributions are modelled with a four-component Gaussian Mixture Model (GMM) and the HMM parameters are calculated using the Baum–Welch algorithm. The note and the musicological models then constitute a probabilistic note network, which is used for the transcription of melodies by finding the most probable path through it using a token-passing algorithm. Tokens emitted out of a note model represent note boundaries.

*1.2.1.4 Segmentation of consecutive notes at the same pitch*. In the systems where segmentation is primarily based on frequency variations, consecutive notes with equal pitches are often left unsplit. This occurs both when legato is performed and when a maximum inactivity time (normally referred to as "sleeping time") is allowed in pitch tracking. However, this track inactivity is often necessary in order to handle situations when pitches pass undetected in a few frames, despite the fact that the respective note is sounding.

Approaches that do not permit track inactivity or admit it only during very short intervals usually cause over-segmentation. This seems to be the case of Bello's method [described in Gómez et al. (2006)]. Although not many details are provided, we can presume that the creation of pitch tracks did not allow sufficient frame inactivity, since a profusion of fragments corresponding to the same note often results.

In Eggink and Brown (2004), frame sleeping is consented to and notes are then split when abrupt discontinuities in F0 intensity occur. However, this simple scheme suffers from the same shortcomings associated with amplitude-based note segmentation, namely regarding the accurate definition of thresholds: a satisfactory balance between over and under-segmentation is hard to attain.

This problem is partly solved in Kashino et al. (1995), where terminal point candidates, which correspond to clear minima in the energy contour of each pitch track, are either validated or rejected according to their likelihood and on the detected rhythmic beats. This is much more robust than using only amplitude information but, even so, consecutive notes occurring in between beats may be left unsegmented.

In the note segmentation scheme described in Ryynänen and Klapuri (2005), it is not obvious how this issue is addressed. In fact, the connections between the three states in the models of note events are not strictly left-to-right: the attack state has a left-to-right connection with the sustain state, but this and the noise state might alternate. Thus, when a token is sent to the attack of another note event, a segmentation boundary becomes evident, no matter whether the MIDI note number is the same or not. However, when there is a transition from the noise to the sustain state in a note model, it is not clear if pitch was undetermined for a while or if two consecutive notes at the same pitch were present.

*1.2.1.5 Our approach to note segmentation*. Given the described strengths and weaknesses of amplitude and frequency-based segmentation, our method combines both approaches. Pitch tracks are first constructed, permitting track inactivity in order to cope with undetected, noisy or masked pitches, preventing over segmentation of pitch tracks. Then, frequency-based segmentation is carried out so as to split tracks containing several notes at different pitches. Finally, amplitude-based segmentation is employed, along with explicit onset detection, so as to break apart tracks with consecutive notes at the same pitch.

### *1.2.2 Note labelling*

After segmentation, a note label has to be assigned to each of the identified segments. Typically, pitch detection is executed on short time frames and the average F0 in a segment is quantized to the frequency of the closest equal temperament note (e.g. McNab et al., 1996a; Martins, 2001). This averaging strategy might deal well with frequency modulation, but does not seem appropriate when glissando is present.

In other approaches, the average F0 is computed in the central part of the note, since pitch errors are more likely to occur at the attack and at the decay (Clarisse et al., 2002). In monophonic transcription systems, filtering may be implemented as well to cope with outliers or octave errors (Clarisse et al., 2002). In addition, the median of F0 values may be used rather than the average.

In our method, we convert sequences of pitches to sequences of MIDI values and employ a set of filtering rules that take into consideration glissando, vibrato and other forms of frequency modulation to come up with a candidate MIDI value. Tuning compensation is then applied to the obtained note, as described in the next subsection.

### *1.2.3 Adaptation to instrument and singer's tuning*

Methodologies for note labelling should handle the case where songs are performed off-key, e.g. when the instruments are not tuned to the equal temperament frequencies. This is also frequent in monophonic singing, since only a few people have absolute pitch.

Also, non-professional singers (no matter if they have absolute pitch or not) have a tendency to change their tuning during longer melodies, typically downwards, as referred to in Ryynänen (2004, p. 27).

Some systems attend to this problem, particularly in the transcription of the singing voice or in the adaptation of note labelling to the intonation of the performer (e.g. McNab et al., 1996b; Haus & Pollastri, 2001; Viitaniemi et al., 2003; Ryynänen, 2004). Namely, McNab et al. (1996b) devise a scheme for adjusting note labelling to the own-tuning of individual users. There, a constantly changing offset is employed, which is initially estimated by the difference between the sung tone and the nearest one in the equal temperament scale. Then, the resulting customized musical scale continuously alters the reference tuning, in conformity with the information from the previous note. This is based on the assumption that singing errors tend to accumulate over time. On the other hand, Haus and Pollastri (2001) assume constant sized errors. There, note labelling is achieved by estimating the difference from a reference scale (the equal temperament scale in this case), then conducting scale adjustment and finally applying local refinement rules.

The described approaches make sense in monophonic contexts, where we readily know that all the obtained notes represent the melody. Then, individual singer tuning can be estimated using the set of sung notes. But the same does not apply in polyphonic contexts, where notes from different parts are simultaneously present. In this case, slight departures from the equal temperament scale may occur in singing. This occurs, for example, in a few notes of an excerpt from Eliades Ochoa which we employ (see Table 4, Section 4). However, since many notes are present and source separation is a complex task to accomplish, it is difficult to estimate the tuning of a particular singer (or instrument).

Therefore, we propose a different heuristic for dealing with deviations from the equal temperament scale, which is partly based on the assumptions that off-key instrumental tuning is not significant, and neither are tuning variations in singing, as the employed songs are performed by professional singers in a stable instrumental set-up.

## 2. Melody detection approach: overview

Our melody detection algorithm (Figure 1) comprises three main modules, where a number of rule-based procedures are proposed to attain the specific goals of each unit: (i) *pitch detection*; (ii) *determination of musical notes* (with precise temporal boundaries and pitches); and (iii) *identification of melodic notes*. We follow a multi-stage approach, inspired on principles from perceptual theory and musical practice. Physiological models and perceptual cues of sound organization are incorporated into our method, mimicking the behaviour of the human auditory system to some extent. Moreover, musicological principles are applied, in order to support the identification of the musical notes that convey the main melodic line.

Different parts of the system were described in previous publications (Paiva et al., 2005a,b, 2006) and, thus, only a brief presentation is provided here, for the sake of completeness. Improvements and additional features of the second module (determination of musical notes) are described in more detail.

In the multi-pitch detection stage, the objective is to capture the most salient pitch candidates in each time frame, which constitute the basis of possible future notes. Our pitch detector is based on Slaney and Lyon's (1993) auditory model, using frames of 46.44 ms with a hop size

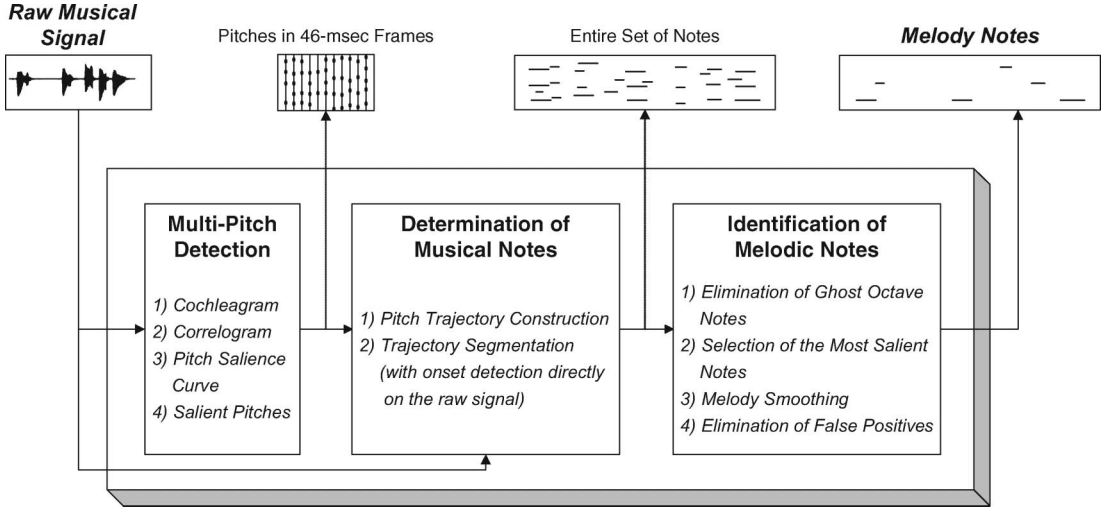


Fig. 1. Melody detection system overview.

of 5.8 ms. For each frame, a cochleagram and a correlogram are computed, after which a pitch salience curve is obtained by summing across all autocorrelation channels. The pitch salience in each frame is approximately equal to the energy of the corresponding fundamental frequency. We follow a strategy that seems sufficient for a melody detection task: instead of looking for all the pitches present in each frame, as happens in general polyphonic pitch detectors, we only capture the ones that most likely carry the main melody. These are assumed to be the most salient pitches, corresponding to the highest peaks in the pitch salience curve. A maximum of five pitch candidates is extracted in each frame. This value provided the best trade-off between pitch detection accuracy and trajectory construction accuracy, in the following stage. Details on the pitch detection algorithm can be found in Paiva et al. (2005a).

Unlike most other melody extraction approaches, we aim to explicitly distinguish individual musical notes, characterized by specific temporal boundaries and MIDI note numbers. In addition, we store their exact frequency sequences and intensity-related values, which might be necessary for the study of performance dynamics, timbre, etc. We start with the construction of pitch trajectories, formed by connecting pitch candidates with similar frequency values in consecutive frames. Since the created tracks may contain more than one note, temporal segmentation must be carried out. This is accomplished in two steps, making use of the pitch and intensity contours of each track, i.e. frequency and salience-based segmentation. This is the main topic of this article and is described in the following sections.

In the last stage, our goal is to identify the final set of notes representing the melody of the song under analysis. Regarding the identification of the notes bearing the melody, we found our strategy on two core assumptions that we designate as the salience principle and the melodic smoothness principle. By the salience principle, we assume that the melodic notes have, in general, a higher intensity in the mixture (although this is not always the case). As for the melodic smoothness principle, we exploit the fact that melodic intervals tend normally to be small. Finally, we aim to eliminate false positives, i.e. erroneous notes present in the obtained melody. This is carried out by removing the notes that correspond to abrupt salience or duration reductions and by implementing note clustering to further discriminate the melody from the accompaniment.

## 3. Determination of musical notes

### 3.1 Pitch trajectory construction

In the identification of the notes present in a musical signal, we start by creating a set of pitch trajectories, formed by connecting pitch candidates with similar frequency values in consecutive frames. The idea is to find regions of stable pitches, which indicate the presence of musical notes. In order not to lose information on note dynamics, e.g. glissando, legato, vibrato or tremolo, we took special care to ensure that such behaviours were kept within a single track. The pitch trajectory construction algorithm receives as input a set of pitch candidates, characterized by their frequencies and saliences, and outputs a set of pitch trajectories, which constitute the basis of the future musical notes.

In perceptual terms, such pitch trajectories correspond, to some extent, to the perceptually atomic elements referred to in Bregman (1990, p. 10) In effect, in the earlier stages of sound organization, the human auditory system looks for sonic elements that are stable in frequency and energy over some time interval. In our work, we only resort to frequency information in the development of these atoms. Anyway, energy information could have also been incorporated for the sake of perceptual fidelity. Actually, we have exploited it to disentangle situations of peak competition among different tracks, but frequency information proved sufficient even in such cases.

We follow rather closely Xavier Serra's peak continuation mechanism (Serra, 1989, pp. 61–70, 1997) and so only a brief description is provided here. A few differences are, nevertheless, noteworthy. Other approaches for peak tracking based on Hidden Markov Models or Linear Prediction Coding can be found, e.g. in Satar-Boroujeni and Shafai (2005), Lagrange et al. (2003), and Depalle et al. (1993).

Since we have a limited set of pitch candidates per frame, our implementation becomes lighter. In fact, Serra looks for regions of stable sinusoids in the signal's spectrum, which leads to a trajectory for each found harmonic component. In this way, a high number of trajectories have to be processed, which makes the algorithm a bit heavier, though the basic idea is the same. Moreover, as in our system the number of pitches in each frame is small, these are clearly spaced most of the time, and so the ambiguities in trajectory construction are minimum.

The algorithm is grounded on three main parameters (see Table 1): a maximum frequency difference between consecutive frames (*maxSTDist*), a maximum inactivity time in each track (*maxSleepLen*) and a minimum trajectory duration (*minTrajLen*). Figure 2 illustrates it graphically. There, black squares represent the candidate pitches in the current frame *n*. Black circles connected by thin continuous lines indicate trajectories that have not been finished yet. Dashed lines denote peak continuation through sleeping frames. Black circles connected by bold lines stand for validated trajectories, whereas white circles represent eliminated trajectories. Finally, grey boxes indicate the maximum allowed frequency distance for peak continuation in the corresponding frame.

Table 1. Pitch trajectory construction parameters.

| Parameter Name | Parameter Value |
|---|---|
| *maxSTDist* | 1 semitone |
| *maxSleepLen* | 62.5 ms |
| *minTrajLen* | 125 ms |

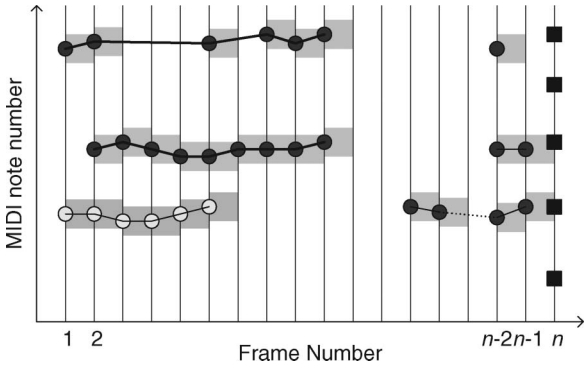


Fig. 2. Pitch trajectory construction algorithm (adapted from Martins, 2001, p. 43).

### 3.1.1 Maximum frequency difference

We defined *maxSTDist* as one semitone since the amount of frequency changing in vibrato, for both the singing voice and musical instruments, is typically around one semitone (Handel, 1989, p. 177). Naturally, this value may vary significantly. For example, the vibrato of lyric singers may reach much broader pitch variations (e.g. three semitones were observed in the female opera excerpt in Table 4, Section 4). As for the frequency of vibrato, typical values are close to 6 Hz (Handel, 1989, p. 177). In practice, separations of almost 2 semitones are permitted, due the fact that continuation uses MIDI numbers.

In this way, the described dynamical features are satisfactorily kept within a common track, instead of being separated into a number of different trajectories, e.g. one trajectory for each note that a glissando may traverse. Hence, a single trajectory may contain more than one note and, therefore, trajectory segmentation based on frequency variations is carried out in the next stage of the melody detection algorithm.

To be more precise, even if a low frequency distance were imposed, some trajectories could contain more than one note, because of smooth transitions between notes, e.g. in legato performances. To cope with this situation, some authors (e.g. Eggink & Brown, 2004) compare the maximum allowed distance to the frequency average of the last few frames. However, as discussed in Section 1.2, it is not assured that individual tracks will contain only one note. Also, this strategy is not robust to missing pitches in tracks with vibrato, which could cause abrupt frequency jumps.

### 3.1.2 Maximum inactivity time

One important aspect to consider in any pitch tracking methodology is that pitches might pass undetected in some frames as a result of noise, masking from other sources or low peak amplitude.

Thus, the second parameter, *maxSleepLen*, specifies the maximum time where a trajectory can be inactive, i.e. when no continuation peaks are found. If this number is exceeded, the trajectory is stopped. For inactive frames, both the frequency and salience values are set to zero. As a result, many sparse trajectories arise (most of them relating to weak notes), which might still be part of the melody.

The maximum inactivity time is set to 62.5 ms. This value was assigned in conformity with the defined minimum note duration (125 msec, see discussion below), being half of it. Although its value may seem too high, it was intentionally selected. Indeed, lower maximum inactivity times usually lead to over-segmentation of an actual note (i.e. a profusion of short trajectories at the same MIDI number). This is due to the fact that, in polyphonic signals, pitch masking occurs more notoriously than in monophonic audio. Therefore, these should be merged later on. Conversely, admitting a longer maximum inactivity time has the drawback that notes played consecutively with only brief pauses be kept within only one track. To this end, trajectory segmentation, now based on salience variations, must be performed.

The reason why we prefer the ''track splitting'' over the ''track merging'' paradigm is that, even with a perfect pitch detector, consecutive notes at the same pitch might be integrated into one single track, e.g. when notes are played legato. The energy level decreases but no silence actually occurs and so track splitting had to be conducted anyway.

### 3.1.3 Minimum trajectory duration

The last parameter, *minTrajLen*, controls the minimum trajectory duration. Here, all finished tracks that are shorter than this threshold, defined as 125 ms, are eliminated. This parameter was set in conformity with the typical note durations in Western music. As Bregman points out, ''Western music tends to have notes that are rarely shorter than 150 ms in duration. Those that form melodic themes fall in the range of 150 to 900 ms. Notes shorter than this tend to stay close to their neighbours in frequency and are used to create a sort of ornamental effect'' (Bregman, 1990, p. 462).

The results of the process for a simple monophonic saxophone riff example are presented in Figure 3. There,

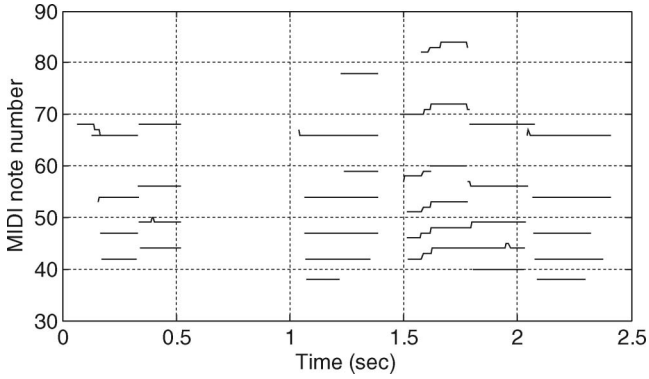


Fig. 3. Results of the pitch trajectory construction algorithm.

we can see that some of the obtained trajectories comprise glissando regions. Also, some of the trajectories include more than one note and should, thus, be segmented.

### 3.2 Frequency-based track segmentation

The trajectories that result from the pitch trajectory construction algorithm may contain more than one note and, therefore, must be divided in time. In frequency-based track segmentation, the goal is to split notes of different pitches that may be present in the same trajectory, coping with glissando, legato, vibrato and other sorts of frequency modulation.

#### *3.2.1 Note segmentation*

The main issue with frequency-based segmentation is to approximate the frequency curve by piecewise-constant functions (PCFs), as a basis for the definition of MIDI notes. However, this is often a complex task, since musical notes, besides containing regions of nearly stable frequency, also comprise regions of transition, where frequency evolves until (pseudo-) stability, e.g. glissando. Additionally, frequency modulation may also occur, where no stable frequency exists. Yet, an average stable fundamental frequency can be determined.

Our problem could thus be characterized as one of finding a set of piecewise-constant/linear functions that best fits to the original frequency curve, under the constraint that it encloses the F0s of musical notes. As unknown variables, we have the number of functions, their respective parameters (slope and bias – null slope if PCFs are used), and start and endpoints.

We have investigated some methodologies for piecewise-linear function approximation. Two main paradigms are defined: ''characteristic points'' and ''minimum error''. Algorithms based on characteristic points do not suit well our needs, e.g. in the case of frequency modulation, and so we constrained the analysis to the minimum error paradigm. This one can be further categorized into two main classes (Pérez & Vidal, 1992). In the first one, an upper bound for the global error is specified and the minimum number of functions that satisfies it, and respective parameters, is computed. This situation poses some difficulties, mostly associated with the definition of the maximum allowed error. In effect, an inadequate definition may lead to a profusion of PCFs in regions of vibrato. In the second (less studied) class, a maximum number of functions is specified, and optimization is conducted with the objective of minimizing the global fitting error. However, these approaches either require that an analytic expression of the curve be known, or need to test different values for the number of functions. Hence, methods in this class do not seem to suit our needs either.

In this way, we propose an approach for the approximation of frequency curves by PCFs, taking advantage of some peculiarities of musical signals.

*3.2.1.1 Filtering of the original frequency curve*. The algorithm starts by filtering the frequency curves of all tracks, in order to fill in missing frequency values that result from the pitch trajectory construction stage. This is carried out by a simple zero-order-hold (ZOH), as in (1). There, $f[k]$ is the frequency value in the current track for its $k$th frame and $f_F[k]$ denotes the filtered curve.

$$\forall_{k\in\{1,2,\cdots,N\}}, f_F[k] = \begin{cases} f[k]; & \text{if } f[k] \neq 0; \\ f_F[k-1]; & \text{if } f[k] = 0. \end{cases} \quad (1)$$

*3.2.1.2 Definition of initial piecewise-constant functions*. Next, the filtered frequency curve is approximated by PCFs through the quantization of each frequency value to the corresponding MIDI note number, as in (2):

$$f_{\text{MIDI}}[k] = \text{round}\left(\frac{\log\left(f[k]/F_{ref}\right)}{\log \sqrt[12]{2}}\right), F_{ref} \approx 8.1758\, Hz, \quad (2)$$

where $f_{\text{MIDI}}[k]$ represents the MIDI note number associated with frequency $f$ in the $k$th frame and $F_{\text{ref}}$ is the reference frequency, which corresponds to MIDI number zero. Therefore, PCFs can be directly defined as sequences of constant MIDI numbers, as in (3).

$$\begin{aligned} &\forall_{i\in\{1,\ldots,nPC\}}, \\ &1: D_i = \{a_i, \ldots, b_i\} = \{k \in \{1,2,\ldots,N\} : f_{\text{MIDI}}[k] = c_i\}, \\ &2: PC_i[k] = c_i, \quad \forall_{k\in D_i}. \end{aligned} \quad (3)$$

There, $PC_i$ represents the $i$th PCF, defined in the domain $D_i$ and characterized by a sequence of constant MIDI numbers equal to $C_i$. Also, the particular case of singleton domains is considered. The total number of PCFs is denoted by $nPC$.

*3.2.1.3 Filtering of piecewise-constant functions*. However, because of frequency variations resulting from modulation or jitter, as well as frequency errors from the pitch detection stage, fluctuations of MIDI note numbers may occur. Also, glissando transitions are not properly kept within one single function. Consequently, $f_{\mathrm{MIDI}}[k]$ must be filtered so as to allow for a more robust determination of PCFs that may represent actual musical notes. Four stages of filtering are applied with the purpose of coping with common performance styles (vibrato and glissando), as well as jitter, pitch detection errors, intonation problems and so forth. These are reflected by the presence of too short PCFs (i.e. PCFs whose length is below *minNoteLen* = 125 ms, according to the typical minimum note durations previously discussed). Short PCFs are unlikely to constitute actual notes on their own as they usually correspond to transients in glissando or frequency modulation, thus needing to be analysed in the context of other neighbour PCFs.

For this reason, the initial filtering stages recur to the presence of long PCFs (having lengths above *minNoteLen*). Long PCFs satisfy the minimum note duration requirement and so are good indicators of stability regions in actual notes, providing good hints for function merging.

*Oscillation filtering*. In the first filtering stage, sequences of PCFs with alternating values are detected and merged (i.e. sequences of PCFs with MIDI note numbers $c$ and $c+1$, or $c+1$ and $c$). These usually reveal zones of frequency modulation within one note. Such oscillations can be combined in a more robust way in case they are delimited by long PCFs for the reasons appointed above. The general methodology proceeds like this:

1. We start by looking for a long PCF.
2. Next, we search for functions with alternating MIDI numbers until another long PCF is found again.
3. The detected oscillations indicate regions of frequency modulation and, therefore, the respective PCFs are fused as follows:
   (a) If the delimiting functions have the same MIDI number, then the resulting PCF receives this value.
   (b) On the other hand, if the last function has a different MIDI number, it is not obvious which pitch should be assigned. Hence, we sum the durations of the short PCFs in between for each of the two possible MIDI note numbers and select the winner as the most frequent one. In order to account for empty frames in the pitch track under analysis, only non-empty frames are used when counting the occurrences of each MIDI note number.
   (c) The alternating short PCFs are then combined with the corresponding initial long PCF.

This procedure is illustrated in Figure 4, where the thick lines denote long PCFs and thin ones represent short functions.

*Filtering of delimited sequences*. In the second stage, the goal is to combine short PCFs that are delimited by two PCFs with the same note number (again, one of them must be long). This may occur due to pitch jitter from noise, pitch detection errors or tuning issues. Such "enclosed" functions are handled in this fashion:

1. Once again, we start by looking for a long PCF.
2. Then, we search forward for another PCF with the same MIDI number.
3. If the sum of the durations of all the PCFs in between is short, those functions and the delimiting ones are merged.
4. We then repeat from step 2, but now to the left of the long PCF found.

This is exemplified in Figure 5.

*Glissando filtering*. Next, sequences representing glissando are analysed as described below (and illustrated in Figure 6):

1. As before, we first look for a long PCF.
2. After that, we search for a succession of short PCFs with constantly increasing or decreasing MIDI numbers (corresponding to the transition region) and possibly ending with a long PCF.
3. The detected transition region suggests a possible glissando, treated as follows:
   (a) If the final PCF in the sequence is long, the merged PCF maintains its value, based on the evidence that the glissando evolved until the long function.
   (b) Otherwise, if the sequence contains only short PCFs and if the duration of the whole sequence is long enough to form a note, the fused PCF

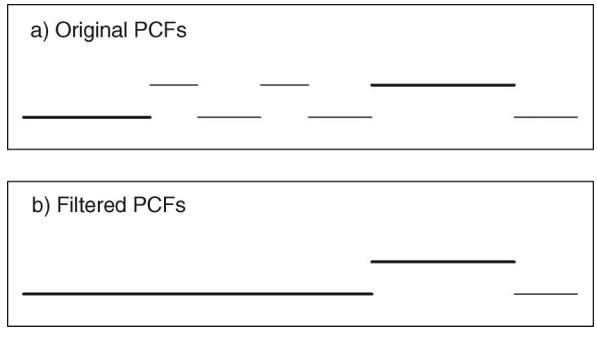


Fig. 4. Oscillation filtering.

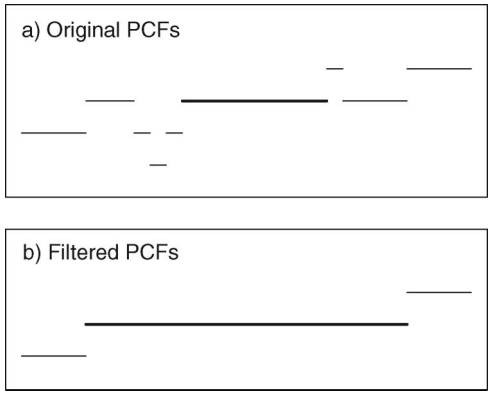


Fig. 5. Filtering of delimited sequences.

receives the value of the most frequent MIDI note number (the last PCF may result from frequency drifting at the ending, and so it does not obtain preference).

*Filtering without the requirement of finding long PCFs*. After making use of long PCFs for filtering, a few short PCFs may still be present, as can be seen in Figure 6. Therefore, two final stages of filtering are applied, much in the same way as filtering of glissando and of delimited sequences was performed, with the difference that no long PCFs need to be found.

In this way, filtering of delimited sequences is first conducted, where we search for a short PCF and then for another PCF after it with equal note number, complying with the procedure described for filtering of delimited sequences. Step 1 is executed differently, since short PCFs are now looked for.

As for glissando filtering, we look for sequences indicating glissando transitions (as in the previous description) starting with short PCFs, and proceeding like this:

1. If the final PCF in the sequence is long, the new PCF keeps its value, as before.
2. Otherwise, if the sequence is long enough to form a note, the new PCF receives the value of the most frequent MIDI note number, also as before.
3. Otherwise, the last MIDI number may correspond to frequency drifting at the decay region. Thus, the sequence of PCFs is merged with the immediately precedent long PCF.

Final short note filtering is illustrated in Figure 7.

*3.2.1.4 Time adjustment*. After filtering, the precise timings of each PCF must be adjusted. Indeed, as a consequence of MIDI quantization, the exact moment where transitions start is often delayed, since the frequencies at the beginning of transitions may be

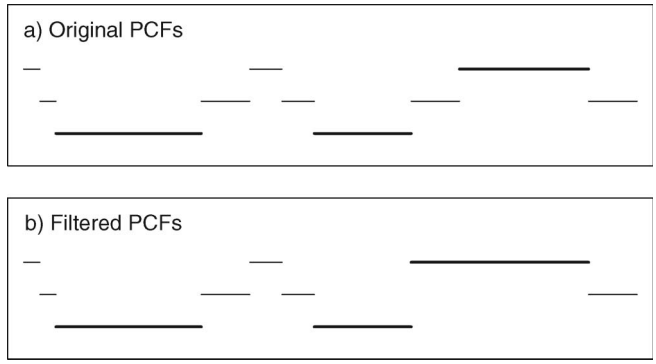


Fig. 6. Glissando filtering.

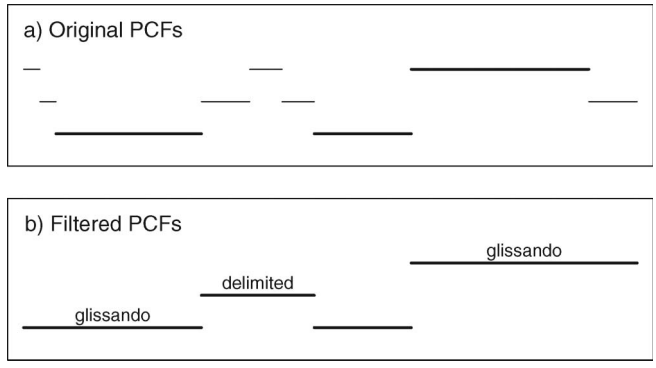


Fig. 7. Final short note filtering.

converted into the previous MIDI number, instead of to the next MIDI number.

Hence, we define the start of the transition as the point of maximum derivative of $f[k]$ after it starts to move towards the following note, i.e. the point of maximum derivative after the last occurrence of the median value.

The median, $md_i$, is calculated only for non-empty frames (non-zero frequency) whose MIDI note numbers maintain their original values after filtering, according to (4). In this way, the median is obtained in a more robust way, since possibly noisy frames and frames corresponding to transient regions are not considered.

$$md_i = \text{median}(f[k]), \quad \forall_{k \in D_i : f_{\text{MIDI}}[k] = c_i \, and \, f[k] \neq 0}. \qquad (4)$$

The discrete derivative is computed using the filtered frequency curve, as in (5):

$$\dot{f}[k] = f_F[k] - f_F[k-1] \qquad (5)$$

### 3.2.2 Note labelling

Once pitch tracks are segmented into regions of different pitch, we have to assign a final MIDI note number to each of the defined PCFs.

Accurate note labelling of singing voice excerpts is usually not trivial because of the enriched dynamics

added by many singers. Moreover, human performances are often unstable (e.g. tuning variations) and affected by errors (e.g. pitch singing errors). These difficulties are not so severe in our circumstances, since we employ recordings of professional singers in stable instrumental set-ups. Therefore, we assume that singing tuning variations are minimum and that the instrumental tuning does not depart significantly from the reference equal temperament scale.

In order to increase the robustness of the assignment procedure, we deal with ambiguous situations where it is not obvious which the correct MIDI number should be. This happens, for instance, when the median frequency is close to the frequency border of two MIDI notes, as in recordings where tuning variations in singing occur (e.g. our Eliades Ochoa's excerpt in Table 4, Section 4) or when instruments are tuned off-key.

*3.2.2.1 Definition of the initial MIDI note number and the allowed frequency range*. Thus, we determine the initial MIDI note number from the median frequency, $md_i$, of each function, according to (2). Then, we calculate the equal temperament frequency (ETF) associated with the obtained MIDI number, by inverting (2). This is carried out with the purpose of checking if the median does not deviate excessively from the reference frequency. Here, we define a maximum distance, *maxCentsDist*, of 30 cents, as in (6).

$$\begin{aligned} iniMIDI_i &= \mathrm{MIDI}(md_i) \\ refF_i &= \mathrm{frequency}(iniMIDI_i) \\ range_i &= \left[refF_i \cdot 2^{-maxCentsDist/1200}; refF_i \cdot 2^{maxCentsDist/1200}\right]. \end{aligned} \tag{6}$$

There, $iniMIDI_i$ represents the candidate MIDI number of the $i$th PCF, $refF_i$ stands for the corresponding ETF, $range_i$ denotes the allowed frequency range and "frequency" is a function for figuring out the ETF from a MIDI note number (i.e. inversion of the "MIDI" function, defined in (2), disregarding the rounding operator).

*3.2.2.2 Determination of the final MIDI note number: tuning compensation*. If the median is in the permitted frequency range of the respective MIDI number, there is evidence that the assigned MIDI number is correct, and so we keep it. It is noteworthy to emphasize that we have intentionally assigned a conservative value to the *maxCentsDist* parameter, as a guarantee that the MIDI values of notes whose medians are in the defined range are correct. This was experimentally confirmed.

However, when the median deviates significantly from the reference, it is not clear whether the initial MIDI number is correct or not. In order to clarify this ambiguity, we use a simple heuristic for the determination of the final MIDI number. Basically, if the median is higher than the upper range limit, the final MIDI number may need to be incremented. This is conducted using the following scheme (we describe the analysis carried out using as example the upper range; in any case, we proceed likewise if the median is below the lower range limit, except that in this case the note number might need to be decremented):

1. We first calculate the frequency value in the frontier of the two candidate MIDI numbers, $borderF_i$ which is 50 cents above the reference frequency of the initial MIDI note number, (7):

$$borderF_i = refF_i \cdot 2^{50/1200}. \tag{7}$$

2. Next, we count (i) the number of frames, *numH*, for which the frequency is above the frontier, i.e. the number of frequency values corresponding to the incremented MIDI number and (ii) the number of frames, *numL*, where the frequency is below the median. Then:
   (a) If $numH > numL$, we conclude that the final MIDI number should be changed to the incremented value.
   (b) Otherwise, it is left unchanged.

The parameters used in this algorithm are presented in Table 2.

An example of obtained results is depicted in Figure 8 for a pitch track from Eliades Ochoa's "Chan Chan" and the female opera excerpt presented in Table 4, Section 4. There, dots denote the F0 sequence under analysis, grey lines are the reference segmentations, dashed lines denote the results attained prior to time correction and final note labelling and solid lines stand for the final achieved results. It can be seen that the segmentation methodology works quite well in these examples, despite some minor timing errors that may have even derived from annotation inaccuracies. The results for the sketched opera track, where strong vibrato is present, are particularly satisfactory.

### 3.3 Salience-based track segmentation

As for salience-based track segmentation, the objective is to separate consecutive notes at the same pitch, which the pitch trajectory construction algorithm may have interpreted as forming one single note.

Ideally, we would conduct note onset detection directly on the audio signal in order to locate the beginnings of the musical notes present. However, robust onset detection is a demanding task, even for monophonic recordings. For example, most methodologies that rely on variations of the amplitude envelope behave satisfactorily for sounds with sharp attacks, e.g. percussion or plucked guitar strings, but show some difficulties

Table 2. Parameters for frequency-based track segmentation.

| Parameter Name | Parameter Value |
|---|---|
| *minNoteLen* | 125 ms |
| *maxCentsDist* | 30 |

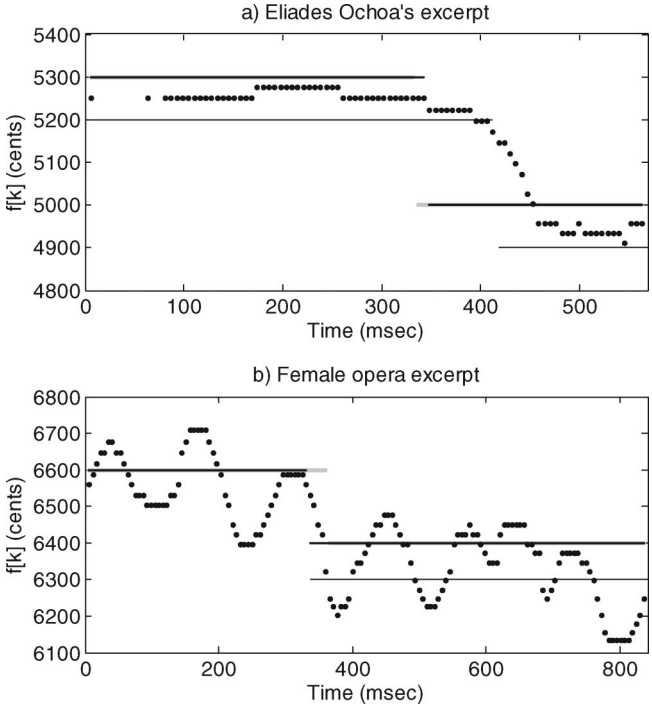


Fig. 8. Results of the frequency-based track segmentation algorithm.

when notes are played with little amplitude inflection, for example, in glissando or in intentionally smooth attacks, e.g. bowed violin strings. The problem is all the more acute in polyphonic mixtures, where energy bursts from the attacks of different notes overlap in time.

On the other hand, the pitch salience curve of each track holds some interesting properties that could also be exploited. First, the salience value depends on the evidence of pitch, which is correlated (though not exactly equal) to the energy of the F0 under consideration. Hence, the envelope of the salience sequence is similar to an amplitude envelope: it grows at note attack, has then a steadier region and decreases at the decay. For this reason, notes could be segmented by detecting clear minima in the pitch salience sequence. The drawback of this approach is the accurate identification of actual note segmentation points, as this curve is often affected by noise due to inactive frames, harmonic collisions, as well as performing styles such as tremolo.

In order to conduct a more robust segmentation of consecutive notes at the same pitch, we combine onset detection with the analysis of the pitch salience curve with the objective of matching detected onsets to salience minima in the curve.

### 3.3.1 Candidate segmentation points

In a first attempt towards salience-based segmentation, we developed a prominent valley detection method for determining salient minima corresponding to candidate segmentation points.

*3.3.1.1 Filtering of the original salience curve*. As in the frequency-based segmentation stage, we start by filtering the salience sequence with a ZOH, due to missing values.

*3.3.1.2 Looking for clear salience minima*. After ZOH filtering, we iteratively look for all clear local minima of the filtered salience sequence, $s_F[K]$, i.e. sufficiently prominent minima (as defined below). This is carried out in this manner:

1. First, all local minima and maxima are found, coping with plateaus. In these situations, the indexes of the corresponding minima/maxima are assigned to the middle of the plateau.
2. Then, only deep enough minima are selected. This is accomplished in the following recursive procedure:
   (a) The global minimum of $s_F[K]$ is found.
   (b) Next, the set of all local maxima is divided into two subsets, one to the left and another one to the right of the global minimum.
   (c) The global maximum for each subset is then determined.
3. The global minimum is selected as a clear minima if its prominence, i.e. the difference between its amplitude and that of both the left and right global maxima, is above the defined minimum peak-valley distance, *minPvd*.
4. Finally, the set of all local minima is also divided into two new intervals, to the left and to the right of the global minimum.
5. The described operations are then recursively repeated for each of the new subsets until all deep minima and respective prominences are found.

One difficulty of the proposed scheme is its lack of robustness. In effect, the best value for *minPvd* was found to vary from track to track, along different song excerpts. Indeed, a unique value for that parameter leads to both missing and extra segmentation points. Also, it is sometimes difficult to distinguish between note endings and amplitude modulation in some performances.

Therefore, this method was combined with note onset detection directly on the audio signal. The obtained onsets are then matched to the candidate segmentation points that resulted from the detection of prominent valleys.

In this way, *minPvd* should receive a low, conservative, value so that missing segmentation points are unlikely. In addition, this parameter ought to be adaptive in order to cope with differences in salience ranges across different notes. Hence, *minPvd* was set to 10% of the maximum amplitude range of the salience sequence under consideration (whose values belong to the [0;100], after the normalization conducted in the pitch detection stage). Due to the defined low value, extra false segmentation points occur, which are eliminated later on via onset matching. Moreover, as a consequence of employing a low threshold, the encountered minima are not that "clear" any longer.

Figure 9 illustrates the algorithm for detection of candidate segmentation points. There, the pitch salience sequence of a trajectory from Claudio Roditi's performance of "Rua Dona Margarida" is depicted, where "o" represent correct segmentation candidates and "*" denote extra segmentation points.

### 3.3.2 Onset detection

The objective of onset detectors is to accurately locate the beginnings of musical notes in acoustic signals. Humans perceive onsets as a consequence of noticeable changes in the intensity, pitch or timbre of a sound (Klapuri, 1999).

A substantial amount of research related to onset detection has been recently conducted. This constitutes a research topic of its own and for this reason a detailed study of the subject is out of the scope of our work. Conversely, we used the pragmatic strategy of basing our efforts on state-of-the-art techniques. For an overview of the area (see, e.g. Bello et al., 2005; Klapuri, 1999).

The algorithm implemented in our system is largely based on Klapuri (1999), with some adaptations. His approach shows some similarities with parts of Eric Scheirer's mechanism for tempo and beat analysis (Scheirer, 1998), from which we have also borrowed a few elements.

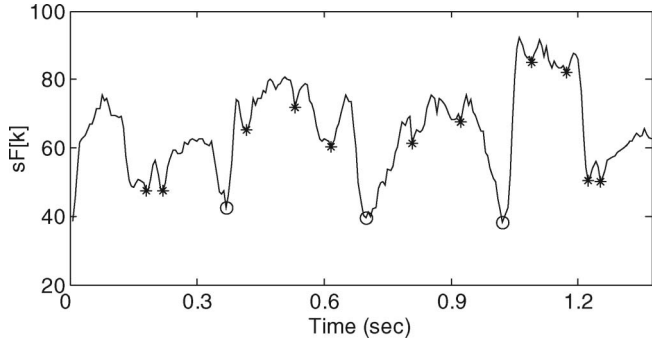


Fig. 9. Results of salience-based track segmentation: initial candidate points.

*3.3.2.1 Band-pass filtering*. The central idea is to perform onset detection in a band-wise manner. A bank of nearly critical band filters is chosen, covering the frequencies from 44 Hz to the Nyquist frequency. For a sampling frequency $f_s = 22050$Hz, 20 filters result, where the first one is low-pass, the last one is high-pass and the remaining are band-pass.

Elliptic filters are employed, in order to ensure a maximally sharp cutoff in the transition band, as proposed in Scheirer (1998). Since it is important to maintain the temporal properties of the signal in each band, we imposed zero-phase as a requirement. Thus, bi-directional filtering (Smith, 1997, p. 330) is carried out as in (8):

$$S_B^i(z) = S(z) * H^i(z) * H^i(z^{-1}). \tag{8}$$

Here, $S_B^i(z)$ represents the filtered output at band $i$, $H^i(z)$ denotes the filter discrete transfer function at the same band and $S(z)$ represents the original signal, all in the $Z$-domain. As a consequence of bi-directional filtering, and according to the desired final transfer function, we specified the following filter parameters: third-order filters, with 1.5 dB ripple in the pass-band and 20 dB of rejection in the stop-band. The design parameters are (roughly) doubled because of bi-directional filtering, e.g. sixth-order filters result, in conformity with the definitions in Scheirer (1998). As for the cutoff frequencies, the three lowest filters are one-octave wide BPFs, whereas the remaining are third-octave wide BPFs, with no band overlapping.

*3.3.2.2 Determination of energy variations in each band.* After filtering, the objective is to compute the energy variations in each band, as a basis for the detection of onset components. To this end, the amplitude envelope in each band is first extracted via rectify-and-smooth. This is accomplished like this:

1. The output of each band is first decimated to 200 Hz, so as to make calculations easier.
2. Then, the decimated outputs of each band are full-wave rectified and smoothed with a 100 ms half-Hanning window, as in (9):

$$\begin{aligned} s_H^i[k] &= \left|s_B^i[k]\right| * w[k - (W/2)]; \\ w[n] &= 0.5 - 0.5\cos\left(\tfrac{2\pi(n-1)}{W-1}\right);\ n = 1, 2, \ldots; W/2. \end{aligned} \tag{9}$$

Here, $w[n]$ denotes the half-Hanning window and $s_{\mathrm{H}}^i[K]$ represents the smoothed output at band $i$. This window approximates reasonably well the energy integration performed by the human auditory system (Scheirer, 1998). Again, we guarantee zero output delay by shifting the window.

3. Next, information about energy variations in each band is achieved by computing the first-order derivatives, $s^i_H[K]$, of the amplitude envelopes, $s^i_H[K]$.
4. Finally, the derivative curve in each band is half-wave rectified, since we are only interested in the points of positive energy variations.

*3.3.2.3 Integration of the information in all bands and onset detection*. The collected information is then integrated across all bands and then onsets are detected in this fashion:

1. First, we linearly sum the calculated derivatives and look for noticeable maxima in the summed derivative. This is executed much in the same way as in the previous procedure for detection of clear minima, except that now we search for peaks instead of valleys.
2. The summed derivative curve is then normalized to the [0;1] interval.
3. After that, we select initial onset candidates by finding all peaks whose saliences are above $minPeakSal = 0.05$; this value is intentionally low so that no true peaks are lost, which was confirmed experimentally.
4. We then delete components that are closer than $minOnsetDiff = 50$ ms to a more intense component, since some peak neighbourhoods may be very dense (Klapuri, 1999).
5. Finally, clear onsets, i.e. onsets with amplitudes above $clearOnsetMag = 0.4$, are selected; this value provided the best trade-off between over and under detection.

*3.3.3 Validation of candidate segmentation points*

After onset detection, our goal is to validate the previously obtained candidate segmentation points. This strategy has some similarities with Kashino's beat and terminal point integration (Kashino et al., 1995). The main difference is that, there, beats are used in the definition of global processing scopes, rather than note onsets. However, if only the main beats are used, consecutive notes at the same pitch coming out during the interval between two beats may pass undivided. Also, beats occurring in the middle of a note may induce inadvertent segmentations.

*3.3.3.1 Onset matching*. Hence, onset matching is carried out for all candidate segmentation points. Namely, if a candidate valley is close to an identified onset, i.e. they are separated by less than $maxValleyOnsetDiff = 20$ ms, that valley is defined as an actual segmentation point.

Making only use of note onsets leads to a few false negatives, due to undetected onsets. Ideally, our method would catch such absent points if perfect onset detection were achieved. As this is not the case, "clear" segmentation points without matching onsets are erroneously left out. Thus, these candidates are selected no matter if no corresponding onsets are found.

Hence, we accept all clear minima in the salience sequence as valid segmentation points, i.e. valleys whose prominences are above $clearValleyProm = 35$ (recall the normalization to the [0;100] interval, in the pitch detection stage). The attained results are presented in Figure 10 for an excerpt from Claudio Roditi's "Rua Dona Margarida". Gray horizontal lines represent annotated notes, whereas black lines stand for the extracted notes. The small grey vertical lines denote the desired segmentations and the black vertical ones represent the obtained segmentation points.

*3.3.3.2 Note segmentation and time correction*. Next, the collected segmentation points are used to further split the notes that resulted from the frequency-based segmentation stage. Here, the determined points are adjusted to the locations of the detected onsets.

Moreover, the starting times of the notes are also adjusted when onsets are found close to the original note beginnings (with a maximum *maxValleyOnsetDiff* distance, and occurring before the start of the note). This copes to some extent with the problem of noisy attack transients, where the pitch may oscillate significantly. In such cases, the track receives no F0s from the note attack region, causing late note beginnings.

The set of parameters employed for salience-based segmentation is presented in Table 3.

## 4. Experimental results, analysis and conclusions

### 4.1 Ground-truth data

One difficulty regarding the evaluation of MIR systems comes from the absence of sizeable and meaningful standard test collections and benchmark problems. This was partly solved through the creation of a set of evaluation databases for the ISMIR 2004 Melody Extraction Contest (MEC-04) and for MIREX'2005.

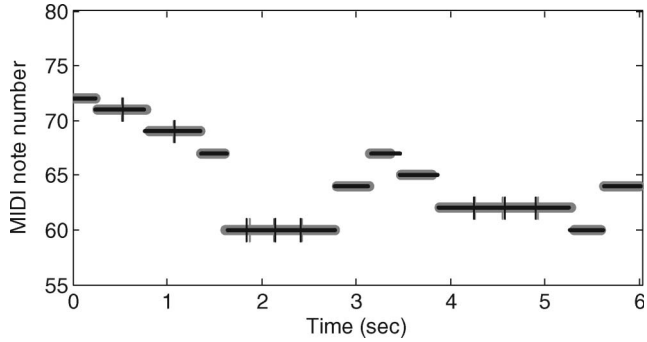


Fig. 10. Results of the salience-based track segmentation algorithm.

In our test-bed, we collected excerpts of about 6 s from 11 songs (the topmost ones in Table 4), enclosing several different categories. In the selected excerpts, the notes bearing the main melody were manually annotated with recourse to visual spectrogram analysis, where note boundaries can be identified to some extent. Naturally, this is not trivial in polyphonic mixtures. Hence, repetitive and localized listening of the song excerpts was executed in parallel, to support and improve the temporal accuracy of the observed note boundaries.

As for the MIREX'2004 database, we adopted the defined training set. Namely, 2 items of synthesized singing voice plus background music (daisy2/3), 2 items of saxophone melodic phrases with accompaniment (jazz2/3), 2 items consisting of a MIDI synthesized polyphonic sound with a predominant voice (midi1/2), 2 items of opera singing, one with a male and another with a female soloist, plus orchestration, and 2 items of sung pop music plus accompaniment (pop1/4) (MIREX, 2004; Gómez et al., 2006). The selected excerpts, each of around 20 s, were (semi-)automatically annotated. As multi-tracking recordings were available, state-of-the-art monophonic pitch detection was carried out on the channel carrying the identified main melody. Note segmentation was then manually applied on the results. Ten additional similar excerpts were used just for testing purposes (not used for training).

We tuned the proposed algorithms with both the training set of the MEC-04 database and a small database we had previously created (hereafter termed PDB, for personal database – see Table 4). We then evaluated the algorithm in the MEC-04 test set.

Each of these databases was designed taking into consideration its diversity and musical content. Namely, all of them were polyphonic and multi-instrumental (except for the MIDI excerpts, which were polyphonic but mono-instrumental) with samples containing notes with real dynamics (glissando, legato, vibrato, tremolo

Table 3. Parameters for salience-based track segmentation (line 1: detection of candidate segmentation points; lines 2 to 11: onset detection; line 12: onset matching; line 13: selection of clear candidates.

| Parameter Name | Parameter Value |
|---|---|
| *minPvd* | 10% of amplitude range |
| *filterbank frequency range* | [44, Nyquist frequency] Hz |
| *filter types* | elliptic |
| *filter order* | 3rd (6th in practice) |
| *band-pass ripple* | 1.5 dB (≈3 dB in practice) |
| *stop-band attenuation* | 20 dB (≈40 dB in practice) |
| *decimation frequency* | 200 Hz |
| *half-Hanning duration* | 100 ms |
| *minPeakSal* | 0.05 |
| *minOnsetDiff* | 50 ms |
| *clearOnsetMag* | 0.4 |
| *maxValleyOnsetDiff* | 20 ms |
| *clearValleyProm* | 35 |

Table 4. Description of used song excerpts. The first 11 rows are from our test-bed and rows 12–21 are from MEC-04's training set.

| ID | Song Title | Category | Solo Type |
|---|---|---|---|
| 1 | Pachelbel's Canon | Classical (MIDI synth.) | Instrumental |
| 2 | Handel's Hallelujah | Choral | Vocal |
| 3 | Enya – Only Time | New Age | Vocal |
| 4 | Dido – Thank You | Pop | Vocal |
| 5 | Ricky Martin – Private Emotion | Pop | Vocal |
| 6 | Avril Lavigne – Complicated | Pop/Rock | Vocal |
| 7 | Claudio Roditi – Rua Dona Margarida | Jazz/Easy | Instrumental |
| 8 | Mambo Kings – Bella Maria de Mi Alma | Bolero | Instrumental |
| 9 | Compay Segundo – Chan Chan | Son | Vocal |
| 10 | Juan Luis Guerra – Palomita Blanca | Bachata | Vocal |
| 11 | Battlefield Band – Snow on the Hills | Scottish Folk | Instrumental |
| 12 | daisy2 | Synthesized singing voice | Vocal |
| 13 | daisy3 | Synthesized singing voice | Vocal |
| 14 | jazz2 | Saxophone phrases | Instrumental |
| 15 | jazz3 | Saxophone phrases | Instrumental |
| 16 | midi1 | MIDI synthesized | Instrumental |
| 17 | midi2 | MIDI synthesized | Instrumental |
| 18 | opera_fem2 | Opera singing | Vocal |
| 19 | opera_male3 | Opera singing | Vocal |
| 20 | pop1 | Pop singing | Vocal |
| 21 | pop4 | Pop singing | Vocal |

and other sorts of frequency and amplitude modulation), as well as consecutive notes at the same pitch, were selected. The range of polyphonic complexity varied from low (e.g. ''Rua Dona Margarida'', with only two pitched voices plus soft percussion) to high (e.g. Handel's ''Hallelujah'', with four choral voices plus orchestra), with most samples in a medium range of nearly 3–5 voices plus percussion. In terms of speed, again the range was broad: some excerpts were slow and soft (e.g. ''Bella Maria de Mi Alma''), others were very fast (e.g. opera_male3) and most were in between (e.g. ''Snow on the Hills''). In some excerpts, although the MIDI samples contained almost no vibrato or glissando, most of the others had it to more (e.g. opera and pop samples in the MEC-04 database, ''Chan Chan'' and ''Palomita Blanca'' in the PDB set), or less extent (e.g. ''Thank You'', ''Complicated''). Moreover, other features such as reverberation were present in samples like Enya's.



### 4.2 Pitch trajectory construction

As expected, pitch tracks were generally very well constructed, since we have a limited number of pitch candidates in each frame (at most five), which minimizes ambiguities in the peak continuation algorithm. A few exceptions occurred in excerpts where close pitches in each frame were found (pitch differences of around one semitone).

In order to evaluate the pitch trajectory construction algorithm in quantitative terms, we first computed the accuracy of the pitch detector, used in the first stage of our melody detection system: 81% average accuracy in the training set. This number was calculated based on the difference between each annotated pitch and its closest extracted pitch among the five extracted, in each frame. Details on the exact metrics used for pitch accuracy computation are described in Gómez et al. (2006).

After the creation of pitch tracks, most of the originally detected pitch candidates were still kept: an average loss of only 1.1%. In fact, with the elimination of tracks shorter than 125 ms, a few actual pitches were lost. The advantage is that 74% of the initial trajectories were eliminated, nearly all of them corresponding to ghost tracks. A higher percentage of deleted tracks could have been attained at the expense of lower pitch accuracy (e.g. with $minTrajLen = 200$ ms, 82.2% of the trajectories are removed, but the pitch accuracy decreases to 77.8%). On the other hand, decreasing the minimum track length slightly improves pitch accuracy at the cost of maintaining a high number of fake tracks (e.g. with $minTrajLen = 100$ ms, the pitch accuracy increases to 80.2% but only 68.3% of the trajectories are removed). We tested a number of values in the range [80;200 ms] and confirmed that the selected value, 125 ms, provided the best trade-off between deletion rate and pitch accuracy. In any case, the variation of this parameter did not cause a dramatic impact in the overall results.

On the other hand, the correct choice of the *maxSleepLen* parameter proved more influential to the pitch track construction process. In order to study its effect, we carried out different trials in the [31.25;125] ms range. Comparing to the selected value, 62.5 ms, lower maximum inactivity times originated over-segmentation of the tracks, i.e. the occurrence of too many tracks corresponding to one actual note (e.g. with 31.25 ms, the number of inadvertent separations of notes across several tracks increased 28.1%). On the other hand, higher values led to the inclusion of too many notes in one single track, ''ignoring'' pauses (e.g. with 125 ms, the number of missing segmentations increased 26.3%). An additional negative effect was the continuation of actual notes with ghost notes or harmonics from other notes.

As for the maximum semitone deviation, the outcome of decreasing the selected value, 1 semitone, was the inadvertent separation of notes with vibrato and glissando. On the other hand, increasing it up to 2 semitones also caused the continuation of actual notes with ghost notes or harmonics from other notes. Moreover, the complexity of pitch trajectory construction increased, as the number of peak continuation candidates in each frame also increased.

### 4.3 Frequency-based segmentation

Detailed results for frequency-based segmentation are presented in Table 5. The first column represents the number of pitch tracks that are necessary to segment after pitch trajectory construction. The second column denotes the percentage of missed segmentations and the third one presents the average time error for the obtained segmentation points in comparison to the annotated ones. The fourth column shows the total number of extracted notes for each song excerpt, from which the percentage of false positives and semitone errors are computed (last two columns). Results are summarized in the bottom lines, where the values in ''#'' columns are summed, whereas all the others are averaged.

In the comparison of extracted and annotated segmentation points, we defined a maximum time deviation of *max* {62.5 ms,[2] 10% of pitch track length}. This is the reason why timing errors in the order of 90 ms were present in a few samples.

The timing accuracy was generally good (an average error of 28 ms), with most deviations under 20 ms. These may have even resulted from annotation inaccuracies. A few slightly higher deviations occurred in tracks with transition zones with many empty frames. However, two particularly high time errors appeared in one track from Dido (ID 4) and another one from the jazz2 excerpt

[2]Half the minimum note length.

Table 5. Results for frequency-based track segmentation.

| ID | # Tracks to Segm. | % False Negatives | Time Error | # Extracted | % False Positives | % Semitone Errors |
|---|---|---|---|---|---|---|
| 1 | 3 | 0.0 | 9.2 | 16 | 0 | 0 |
| 2 | 1 | 0.0 | 16.5 | 13 | 0 | 0 |
| 3 | 0 | (—) | (—) | 11 | 0 | 0 |
| 4 | 2 | 0.0 | 85.8 | 16 | 0 | 0 |
| 5 | 1 | 0.0 | 15.1 | 10 | 0 | 10 |
| 6 | 3 | 0.0 | 10.6 | 14 | 0 | 0 |
| 7 | 2 | 0.0 | 33.5 | 19 | 0 | 0 |
| 8 | 3 | 0.0 | 32.6 | 12 | 0 | 0 |
| 9 | 1 | 0.0 | 24.8 | 10 | 0 | 0 |
| 10 | 0 | (—) | (—) | 11 | 0 | 0 |
| 11 | 4 | 25.0 | 31.3 | 26 | 0 | 0 |
| 12 | 2 | 0.0 | 18.5 | 23 | 0 | 0 |
| 13 | 0 | (—) | 0.0 | 11 | 0 | 0 |
| 14 | 2 | 0.0 | 94.2 | 21 | 0 | 4.8 |
| 15 | 0 | (—) | (—) | 22 | 0 | 0 |
| 16 | 3 | 0.0 | 19.4 | 38 | 0 | 0 |
| 17 | 0 | (—) | (—) | 22 | 0 | 0 |
| 18 | 8 | 37.5 | 17.3 | 37 | 0 | 10.8 |
| 19 | 10 | 80.0 | 12.2 | 52 | 0 | 9.6 |
| 20 | 4 | 50.0 | 5.6 | 34 | 5.88 | 11.8 |
| 21 | 1 | 0.0 | 44.1 | 28 | 0 | 0 |
| Sum/Avg PDB | 20 | 5% | 28.7 ms | 158 | 0.0% | 0.6% |
| Sum/Avg M04 | 30 | 43.3% | 27.2 ms | 288 | 0.7% | 4.9% |
| Sum/Avg | 50 | 28.0% | 28.0 ms | 446 | 0.45% | 3.4% |



(ID 14). There, an interesting situation occurred: the pitch track contained a solo note, which was followed by an accompaniment note and again a note solo. The accompaniment note was the one responsible for the inclusion of two notes in the same pitch track. Then, the most notorious minima in the composite salience sequence did not correspond to the actual note boundaries. This situation gives a good illustration of some of the difficulties entailed in the accurate construction of pitch tracks in polyphonic contexts.

As for the detection of segmentation points, very good results were achieved in the PDB database (5% false negatives). Nevertheless, the average performance in the M04 database was poor, where 43.3% of the required segmentations passed undetected. This was mostly due to the opera excerpts. In reality, the frequency-based segmentation strategy had some trouble with samples with extreme vibrato. In these, the first stage of MIDI quantization led to a succession of several short initial PCFs. Since the method relies on the detection of long PCFs, the resulting ambiguities were not properly disentangled. Moreover, many empty frames existed in those tracks, placing additional difficulties on the algorithm. This was particularly problematic in the male excerpt, where the SACF was particularly noisy. However, if the opera samples are excluded from analysis, an average of 16.7% turns out, which is more acceptable. Indeed, in all the other excerpts most of glissando and frequency-modulated notes were correctly dealt with.

In terms of false positives, only two notes were inadvertently segmented, namely in the pop1 (20) excerpt. In effect, the implemented approach was somewhat more prone to false negatives than to false positives.

Regarding semitone errors, most of them occurred in the opera excerpts. We are especially satisfied with the fact that in 446 extracted notes involving complex dynamics and tuning issues, only 15 semitone errors (3.4%) occurred.

To sum up the analysis on frequency segmentation accuracy, we have also calculated average precision and recall figures. The average recall (i.e. the percentage of annotated segmentation points correctly identified) was 72% and the average precision (i.e. the percentage of identified segmentation points that corresponded to actual segmentation points) was 94.7%.

Most of the encountered difficulties came from opera tracks with extreme vibrato. In those cases, the number of false negatives and semitone errors was clearly above the average. In any case, in excerpts with moderate vibrato, results were quite satisfactory.

As expected, assigning an adequate value to the *minNoteLen* parameter proved decisive, as the accuracy of the algorithm may vary considerably with it. If the minimum note duration is low, the number of missing segmentation points reduces whereas the number of false positives increases. For instance, with *minNoteLen* = 62.5 ms, the average recall boosted to 100%, but the average precision dropped to 52.6%. On the other hand, when it is too high, the number of true negatives increases and, surprisingly, the number of false positives also increases. For example, with *minNoteLen* = 250 ms, the average recall and precision have both dropped to 40 and 52.6% respectively. In fact, since the segmentation algorithm relies on the presence of long notes, these become less likely and so short notes guide the process. As a consequence, notes with frequency modulation are inadvertently segmented into notes at different pitches, increasing, in turn, the number of semitone errors. We have attempted several values in the range [62.5;250] ms. We experimented with several values in the range [62.5,250] and verified that the selected value, 125 ms, provided the best trade-off between precision, recall and semitone error rate.

As for the *maxCentsDist* parameter, its variation had a much lower impact on the overall results. We tested values in range [15;45] cents and observed a maximum of 26 semitone errors and a minimum of 15 (with the selected value, 30 cents).

### 4.4 Salience-based segmentation

Regarding salience-based segmentation, results are presented in Table 6.

As expected, this is quite complex in polyphonic contexts, since onset detection is not trivial in this case. Furthermore, harmonic collisions between different sources corrupt the energy levels of F0 candidates, in which the algorithm bases the identification of segmentation candidates. Also, the distinction between amplitude modulation and note boundaries is not always clear. In addition, reverberation effects such as the ones found in Enya's excerpt add extra difficulties to the problem.

Nevertheless, 25% of false negatives and 17.3% of false positives can be considered acceptable for this stage of research. Further complexities were found in Hallelujah (ID 2), Enya (3), Juan Luis Guerra (10), Battlefield Band (11) and male opera (19), where the number of false negatives was clearly above the average. These generally correspond to samples with higher polyphonic complexity, reverberation or tremolo (opera). In other complicated excerpts, such as Eliades Ochoa's, the algorithm was rather successful.

In some other excerpts, false negatives did not occur significantly but false positives appeared instead, e.g. in Ricky Martin (5) and midi2 (17). In reality, the balance between under and over-segmentation proved difficult to achieve.

With respect to timings, some deviations of around 40–50 ms occurred, which are higher than desired, though not excessively. This happened partly because the exact locations of valleys in the pitch salience sequences are disturbed by harmonic collisions with sonic components from other sources. Slight annotation errors also affected it.

To sum up the salience-based segmentation phase, as the number of identified segmentation points was high in comparison to the actual segmentation points, the average precision was low: 41.2%. With respect to recall, an average of 75.0% was accomplished.

We tried different values for the main parameters of the algorithm. Regarding the *clearOnsetMag* parameter, its variation did not have much impact on the overall results. Several values were tested in the range [0.2;0.8], obtaining average recall and precision in the [70.8%;80.6%] and [38.7%;41.3%] intervals, respectively. Particularly, with *clearOnsetMag* = 0.2, the average recall improved to 80.6% and the average precision reduced to 38.7%. However, looking closely at the results, we observed only 4 additional true segmentation points found, against 25 more false positives, the reason why we preferred to define *clearOnsetMag* = 0.4.

The *maxValleyOnsetDiff* parameter also caused low variance in the results. Varying it in the range [10;50] ms led to average recall and precision in the [69.4%;79.2%] and [40.3%;41.2%] intervals, respectively. Namely, with *maxValleyOnsetDiff* = 50 ms, the average recall improved to 79.2% and the average precision reduced to 40.7%. However, as a consequence of the higher allowed time deviation, the average time error increased. Thus, the best trade-off between precision, recall and timing accuracy was obtained with *maxValleyOnsetDiff* = 20 ms.

Unlike the previous parameters, *clearValleyProm* had great impact on the results of the salience-based segmentation algorithm, particularly in the recall metric. In fact, varying this parameter in the range [17.5;70] led to average recall and precision in the [62.5%;97.2%] and [40.5%;41.2%] intervals, respectively. Namely, with *clearValleyProm* = 17.5, the average recall boosted to 97.2% and the average precision reduced to 38.5%. As before, these figures must be analysed with caution, as we are dealing with small numbers. In fact, almost all segmentation points were detected (72, against 54 with the selected value) at the expense of a large increase in the number of false positives (112 against 77). Thus, *clearValleyProm* = 35 offered a better trade-off between over and under-segmentation.

### 4.5 Test set

In order to assess the generality of our approach and the particular parameter tuning, we evaluated it with the test

Table 6. Results for salience-based track segmentation.

| ID | # Tracks to Segm. | % False Negatives | Time Error | # Extracted | % False Positives |
|---|---|---|---|---|---|
| 1 | 0 | (—) | (—) | 16 | 0.0 |
| 2 | 7 | 57.1 | 29.0 | 13 | 7.7 |
| 3 | 5 | 60.0 | 46.1 | 11 | 18.2 |
| 4 | 2 | 0.0 | 25.8 | 16 | 6.3 |
| 5 | 1 | 0.0 | 11.1 | 10 | 60.0 |
| 6 | 3 | 0.0 | 26.3 | 14 | 7.1 |
| 7 | 8 | 0.0 | 17.0 | 19 | 0.0 |
| 8 | 1 | 0.0 | 3.2 | 12 | 8.3 |
| 9 | 3 | 0.0 | 20.9 | 10 | 0.0 |
| 10 | 4 | 50.0 | 9.2 | 11 | 9.1 |
| 11 | 5 | 40.0 | 27.6 | 26 | 0.0 |
| 12 | 1 | 0.0 | 56.4 | 23 | 13.0 |
| 13 | 0 | (—) | (—) | 11 | 27.3 |
| 14 | 3 | 33.3 | 12.8 | 21 | 9.5 |
| 15 | 3 | 0.0 | 3.1 | 22 | 9.1 |
| 16 | 2 | 0.0 | 41.7 | 38 | 5.3 |
| 17 | 6 | 0.0 | 35.7 | 22 | 45.5 |
| 18 | 6 | 33.3 | 52.8 | 37 | 29.7 |
| 19 | 6 | 50.0 | 48.3 | 52 | 38.5 |
| 20 | 1 | 0.0 | 10.1 | 34 | 14.7 |
| 21 | 5 | 20.0 | 43.3 | 28 | 21.4 |
| Sum/Avg PDB | 39 | 28.2% | 22.3 ms | 158 | 8.2% |
| Sum/Avg M04 | 33 | 21.2% | 35.7 ms | 288 | 22.2% |
| Sum/Avg | 72 | 25.0% | 28.8 ms | 446 | 17.3% |



set used in the MIREX'2004 evaluation, which consisted of 10 extra samples of the same kind as its training set.

Regarding frequency segmentation, the average recall was 84.2 (72% in the training set) and the average precision was 76.2% (94.7% in the training set). Most of the tracks resulting from the pitch trajectory construction algorithm were correctly segmented, with an average timing error of 37.5 ms (28 ms in the training set). However, we encountered some difficulties in one of the opera samples, with a number of false positives noticeably above the average, which caused the observed drop in the average precision. This was a direct consequence of the extreme vibrato observed.

As for salience segmentation, we faced the same difficulties described in the training set. Again, a high number of false positives occurred, most visibly in the opera excerpts, resulting in an average precision of 30.8% (41.2% in the training set). In terms of recall, an average value of 77.4% was achieved (72% in the training set), with an average timing error of 30.1 ms (28.8 ms in the training set).

Concerning semitone errors, in 238 extracted notes, only 11 were in error, i.e. 4.6% (3.4% in the training set). Again, most of them occurred in the opera excerpts.

To sum up, despite the variation observed in some of the metrics (either increasing or decreasing), our approach showed reasonable generality, segmenting tracks with moderate to high frequency modulation with very good accuracy, demonstrating sometimes more difficulties in tracks with extreme vibrato and causing the appearance of some false positives in the segmentation of consecutive notes at the same pitch.

Additionally, in the MIREX'2004 evaluation, except for our algorithm only Bello's carried out note segmentation. An edit distance score was calculated, in which our method behaved generally better as a direct consequence of its overall higher pitch detection accuracy (MIREX, 2004; Gómez et al., 2006). Nonetheless, even in several cases where Bello's algorithm had a better pitch detection performance, our approach still yielded a better edit distance score. This is a result of Bello's mechanism for note determination. Although not many details on the pursued strategy are provided (Gómez et al., 2006), we observed that over-segmentation frequently occurred in his method, which probably resulted from too low maximum trajectory sleeping. Hence, a profusion of fragments corresponding to the same note arises. These repetitions are punished as wrong insertions by the edit distance metric, increasing the computed distance.

## 5. Conclusions and future work

During the course of this work, a few obstacles were encountered, which are worth mentioning and pointing out possible solutions. Namely, a difficulty in pitch trajectory construction comes from the fact that some notes might be included in other's pitch tracks because of frequency continuation. This is a complex problem, since timbre is hard to "measure" and, thus, sources are not recognized before peak continuation. In any case, when this situation causes sudden intensity differences, salience-based segmentation can handle it to some extent.

Likewise, notes may also be continued by harmonics of other notes. Again, salience-based segmentation attenuates this difficulty.

Also, the question of closely spaced F0 candidates gave rise to some difficulties in the pitch trajectory construction process, namely in the male opera excerpt. However, as we mentioned previously, in our system the number of pitches in each frame is small and these are clearly spaced most of the time. Hence, the ambiguities in trajectory construction are minimum.

Regarding track inactivity, when F0s are missing the overall energy level in the note's frequency range should be analysed. For the sake of accuracy, a track should only be allowed to sleep if the energy level in its frequency range was sufficient to presume masking. Otherwise, no perceptual restoration would have occurred, being more likely that the note had actually stopped.

As for frequency-based segmentation, the main difficulties of the followed methodology resulted from its dependency on the *minNoteLen* parameter. Indeed, pitch tracks with extreme vibrato were sometimes hard to accurately segment. This is a more fundamental issue, which would probably require a different PCF identification approach. Namely, instead of quantizing the F0 values to MIDI note numbers, the original frequency curve should be directly analysed. However, algorithms for piecewise function approximation seemed inadequate to our problem, besides being rather parameter-dependent.

Finally, the results achieved for salience-based segmentation are encouraging but the balance between over and under-segmentation needs additional attention. Namely, more robust polyphonic onset detectors are required. Also, the onsets of accompaniment notes may mislead the procedure for validation of segmentation candidates. A possible way of improving the method would be to avoid validations by onset candidates that match the beginnings of other detected notes. Moreover, the pitch salience sequence is corrupted by harmonic collisions, especially in excerpts with higher polyphonic complexity. This could be attenuated, e.g. by an iterative estimation and cancellation scheme for pitch detection.

As regards the discrimination between note boundaries and amplitude modulation, this should be further worked out. In fact, Albert Bregman suggests that abrupt rises in intensity represent new notes (Bregman, 1990, p. 71). Therefore, if amplitude modulation is slow, the succession of pitches is simply heard as a single note. However, when amplitude modulation is fast, the higher rates of intensity growth promote the perception of several consecutive notes. In this way, the slope of salience variation until steady-state is reached should be measured and used to validate segmentation candidates.

## Acknowledgements

This work was partially supported by the Portuguese Ministry of Science and Technology, under the program PRAXIS XXI.

## References

Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. & Sandler, M. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, *13*(5), 1035–1047.

Bregman, A.S. (1990). *Auditory scene analysis: the perceptual organization of sound*. Cambridge, MA: MIT Press.

Chai, W. (2001). Melody retrieval on the web. MSc thesis, School of Architecture and Planning, Massachusetts Institute of Technology, USA.

Clarisse, L.P., Martens, J.P., Lesaffre, M., De Baets, B., De Meyer, H. & Leman, M. (2002). An auditory model based transcriber of singing sequences. In *Proceedings of the International Conference on Music Information Retrieval – ISMIR'2002*, Paris, France.

Depalle, P., Garcia, G. & Rodet, X. (1993). Analysis of sound for additive synthesis: tracking of partials using hidden Markov models. In *Proceedings of the International Computer Music Conference – ICMC'93*, Tokyo, Japan.

Eggink, J. & Brown, G.J. (2004). Extracting melody lines from complex audio. In *Proceedings of the International Conference on Music Information Retrieval – ISMIR'2004*, Barcelona, Spain.

Gómez, E., Streich, S., Ong, B., Paiva, R.P., Tappert, S., Batke, J.-M., Poliner, G., Ellis,, D. & Bello, J.P. (2006). A quantitative comparison of different approaches for melody extraction from polyphonic audio recordings. Technical Report, Music Technology Group, Pompeu Fabra University, Spain.

Handel, S. (1989). *Listening – an introduction to the perception of auditory events*. Cambridge, MA: MIT Press.

Haus, G. & Pollastri, E. (2001). An audio front end for query-by-humming systems. In *Proceedings of the International Symposium on Music Information Retrieval – ISMIR'2001*, Bloomington, USA.

Hawley, M. (1993). Structure out of sound. PhD thesis, Media Laboratory, Massachusetts Institute of Technology, USA.

Kashino, K., Nakadai, K., Kinoshita, T. & Tanaka, H. (1995). Organization of hierarchical perceptual sounds: music scene analysis with autonomous processing modules and a quantitative information integration mechanism. In *Proceedings of the International Joint Conference on Artificial Intelligence – IJCAI'95*, Montréal, Canada, pp. 158–164.

Klapuri, A.P. (1999). Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing – ICASSP'99*, Phoenix, USA.

Klapuri, A.P. (2004). Signal processing methods for the automatic transcription of music. PhD thesis, Tampere University of Technology, Finland.

Lagrange, M., Marchand, S., Raspaud, M. & Rault, J.-B. (2003). Enhanced partial tracking using linear prediction. In *Proceedings of the International Conference on Digital Audio Effects – DAFx'03*, London, UK.

Martin, K.D. (1996). Automatic transcription of simple polyphonic music: robust front end processing. In *Proceedings of the 3rd Joint Meeting of the Acoustical Societies of America and Japan*, Honolulu, USA.

Martins, L.G. (2001). PCM to MIDI transposition. MSc thesis, Department of Electrical and Computer Engineering, University of Porto, Portugal.

McNab, R.J., Smith, L.A. & Witten, I.H. (1996a). Signal processing for melody transcription. In *Proceedings of the Australasian Computer Science Conference – ACSC'96*, Melbourne, Australia.

McNab, R.J., Smith, L.A., Witten, I.H., Henderson, C.L. & Cunningham, S.J. (1996b). Towards the digital music library: tune retrieval from acoustic input. In *Proceedings of the ACM International Conference on Digital Libraries – DL'96*, Bethesda, USA.

MIREX (2004). ISMIR'2004 Audio Description Contest (or 1st Music Information Retrieval Evaluation Exchange – MIREX'2004). *International Conference on Music Information Retrieval – ISMIR'2004*. URL: http://ismir2004.ismir.net/ISMIR_Contest.html, available by July 31, 2006.

Paiva, R.P., Mendes, T. & Cardoso, A. (2005a). An auditory model based approach for melody detection in polyphonic musical recordings. In U.K. Wiil (Ed.), *Computer music modeling and retrieval – CMMR 2004*. Esbjerg, Denmark, *Lecture Notes in Computer Science, volume 3310* (pp. 21–40). Berlin/New York: Springer Verlag.

Paiva, R.P., Mendes, T. & Cardoso, A. (2005b). On the definition of musical notes from pitch tracks for melody detection in polyphonic recordings. In *Proceedings of the International Conference on Digital Audio Effects – DAFx'05*, Madrid, Spain.

Paiva, R.P., Mendes, T. & Cardoso, A. (2006). Melody detection in polyphonic musical signals: exploiting perceptual rules, note salience and melodic smoothness. *Computer Music Journal*, *30*(4), 80–98.

Pérez, J.-C. & Vidal, E. (1992). An algorithm for the optimum piecewise linear approximation of digitized curves. In *Proceedings of the International Conference on Pattern Recognition – ICPR'92*, The Hague, The Netherlands, pp. 167–170.

Ryynänen, M.P. (2004). Probabilistic modelling of note events in the transcription of monophonic melodies. MSc thesis, Tampere University of Technology, Finland.

Ryynänen, M.P. & Klapuri, A.P. (2005). Polyphonic music transcription using note event modeling. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics – WASPAA'2005*, New York, USA.

Satar-Boroujeni, H. & Shafai, B. (2005). A robust algorithm for partial tracking of music signals. In *Proceedings of the International Conference on Digital Audio Effects – DAFx'05*, Madrid, Spain.

Scheirer, E.D. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, *103*(1), 588–601.

Scheirer, E.D. (2000). Music-listening systems. PhD thesis, School of Architecture and Planning, Massachusetts Institute of Technology, USA.

Serra, X. (1989). A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition. PhD thesis, Department of Music, Stanford University, USA.

Serra, X. (1997). Musical sound modeling with sinusoids plus noise. In C. Roads, S. Pope, A. Picialli & G. De Poli (Eds.), *Musical signal processing*. Lisse, The Netherlands: Swets & Zeitlinger Publishers.

Slaney, M. & Lyon, R.F. (1993). On the importance of time – a temporal representation of sound. In M. Cooke, S. Beet & M. Crawford (Eds.), *Visual representations of speech signals*. New York: Wiley.

Smith, S.W. (1997). *The scientist and engineer's guide to digital signal processing*. USA: California Technical Publishing.

Viitaniemi, T., Klapuri,, A. & Eronen, A. (2003). A probabilistic model for the transcription of single-voice melodies. In *Proceedings of the Finnish Signal Processing Symposium – FINSIG'2003*, Tampere, Finland.