An Auditory Model Based Approach for Melody Detection in Polyphonic Musical Recordings

Rui Pedro Paiva[1], Teresa Mendes[1], and Amílcar Cardoso[1]

[1] CISUC – Center for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, University of Coimbra (Polo II), P3030 Coimbra, Portugal
{ruipedro, tmendes, amilicar}@dei.uc.pt

Abstract. We present a method for melody detection in polyphonic musical signals based on a model of the human auditory system. First, a set of pitch candidates is obtained for each frame, based on a cochlear model and periodicity detection using correlograms. Trajectories of the most salient pitches are then constructed. Next, note candidates are obtained by trajectory segmentation (in terms of frequency and pitch salience variations). Too short, low-salience and octave-related notes are then eliminated. Finally, the melody is extracted by selecting the most important notes at each time, based on their pitch salience. We tested our method with excerpts from 12 songs encompassing several genres. In the songs where the solo stands out clearly, most of the melody notes were successfully detected. However, for songs where the melody is not that salient, the algorithm was not very accurate. Nevertheless, the followed approach seems promising.

1 Introduction

As a result of recent technological innovations, there has been a tremendous growth in the Electronic Music Distribution (EMD) industry. Factors like the widespread access to the Internet, bandwidth increasing in domestic accesses or the generalized use of compact audio formats with CD or near CD quality, such as mp3, have given a great contribution to that boom. Presently, it is expected that the number of digital music archives, as well as their dimension, grow significantly in the near future, both in terms of music database size and in number of genres covered.

However, any large music database, or, generically speaking, any multimedia database, is only really useful if users can find what they are looking for in an efficient manner. Today, whether it is the case of a digital music library, the Internet or any music database, search and retrieval is carried out mostly in a textual manner, based on categories such as author, title or genre. This approach leads to a certain number of difficulties, namely in what concerns database search in a transparent and intuitive way. Therefore, in order to overcome the limitations described, research is being conducted in an emergent and promising field called Music Information Retrieval (MIR).

Query-by-humming (QBH) [1, 4, 6] is a particularly intuitive way of searching for a musical piece, since melody humming is a very natural habit of humans. Therefore, several technologies have been developed that aim to permit such function. However, presently, this work is being carried out only in the MIDI realm, which places important usability questions. In fact, usually we look for recorded songs, which can be obtained from CDs or are stored in audio formats such as mp3. Looking for musical pieces in the MIDI format is a much easier problem, since this is a symbolic format where all the notes, as well as their timings, are already available. The main issues are, then, to extract the notes from the hummed query (a well-known monophonic pitch[1] extraction problem,) and to match the query to the melody, usually available in the MIDI file (an information retrieval problem).

Querying "real-world" polyphonic recorded musical pieces requires that some sort of melody representation be extracted beforehand, which creates many more difficulties. Polyphonic musical signals can be converted to symbolic formats either manually or automatically. Manual conversion requires, obviously, a tremendous amount of man-work and specialized skills. On the other hand, analyzing polyphonic musical waveforms is a rather complex task, since we can have many different types of instruments playing at the same time, whose spectra interfere severely with each other. This fact makes it very complicated to separate the different sound sources.

Source separation is a major concern for polyphonic music analysis and automatic music transcription systems, and has no general solution yet. One way to approach this problem is to build computer models that emulate human auditory processing. The human brain processes auditory information in a way called "auditory scene analysis" [3]. As an attempt to replicate human behavior, some work has been carried out aiming to develop computational auditory scene analysis systems. The results obtained are not very accurate yet and are only acceptable for simpler or well-constrained problems. Namely, Ellis [5] tries to analyze a sound waveform by means of competitive theories, where each of them proposes a combination of sounds that might have produced the resulting sound. Sound source models are used as a basis for the proposed method. Bello *et al* [2] and Martin [12] have used computational blackboard systems for simple automatic music transcription. The blackboard system is composed of a global database, where hypotheses are proposed and developed, a scheduler that determines how hypotheses are developed, and knowledge sources, corresponding to experts. Scheirer [14] proposes a model based on perceptual issues, using dynamic clustering of comodulation data. In contrast to the other systems referred, this model is designed for analysis of complex music. Klapuri [10] proposed a method for multi-pitch estimation where the musical signal is analyzed at separate frequency bands. Namely, 18 logarithmic distributed bands from 50 Hz to 6 kHz are used. Then at each band, a fundamental frequency likelihood vector is calculated. Finally, the results from each band are combined to yield global pitch likelihoods. They report results that outperform the average of ten trained musicians. Other models impose constraints in the number of instruments present or the harmonic interaction between them, as referred in [7].

[1] In this paper, we use the term pitch indistinctly of fundamental frequency, though the former is a perceptual variable, whereas the latter is a physical one.

Melody detection can be seen as a sub-problem of polyphonic pitch detection and source separation, where the aim is to detect the main melodic line, regardless of the other sources present. This requires the detection of the dominant notes at each time, not the whole set of notes present. For instance, when we hear a pop song, we have vocals, guitar, bass, percussion and so forth. Yet, in spite of all that information, our brains still can retain the main melodic line.

Only little work has been carried out in the particular problem of melody detection in "real-world" songs. One interesting approach is the one followed by Goto [7]. The author uses a probabilistic model for the detection of melody and bass lines. The sound wave is first band-pass filtered and then a probability density function (*pdf*) is computed for each signal component. The *pdf*s are generated from a weighted-mixture of tone models of all possible fundamental frequencies. The more dominant a model is in the PDF, the more likely the fundamental frequency belongs to that model. The author compared the dominant frequencies detected with hand-labeled marked notes and reports an average rate of 88.4% for the melodic pitch line.

Song *et al* [19] use a different approach, based on the fact that there is no single method that is both accurate and generic. They argue that their method is more pragmatic when the final goal is QBH: instead of trying to extract the melody, they use a mid-level melody representation, which consists of a sequence of audio segments where each segment contains a set of note candidates. Then, they use a variation of dynamic programming for matching the query with the melody mid-level representation.

In this paper, we describe a multi-stage method for melody detection, based on a model of the human auditory system [16]. In short, the method works as follows. The sound wave is first divided into frames where stationarity can be assumed. For each frame, we get a set of pitch candidates, based on a cochlear model and periodicity detection using correlograms. We determine pitch candidates by finding relevant peaks in a summary correlogram, where peak amplitude gives information regarding its salience. Then, we create pitch trajectories, based on frequency proximity. After trajectory creation, note candidates are obtained by trajectory segmentation and elimination. Short-duration, low-salience and octave-related notes are then eliminated. Finally, the melody is extracted by selecting the most important notes at each time, based on their pitch saliences.

We tested our system on excerpts of 12 songs, encompassing several different genres. The obtained notes were then compared with the correct ones, previously hand-labeled. In the songs where the solo stands out clearly, most of the melody notes were successfully detected. However, for songs where the melody is not that salient, the algorithm was not so accurate. Yet, we could say that the obtained results presented are encouraging.

The following sections describe the work carried out in this paper. Section 2 describes the melody detection method. In Section 3, experimental results are presented and evaluated. Finally, in Section 4, conclusions are drawn and possible directions for future work are pointed out.

2 Melody Detection Method

Our melody detection algorithm is composed of five modules, illustrated in Fig. 1.

The first module, multi-pitch detection (MPD), receives a raw polyphonic musical signal and returns a set of pitch candidates and their respective saliences. Then, pitch trajectories are created based on frequency proximity, in the multi-pitch trajectory construction (MPTC) module.

The resulting trajectories are then segmented, based on frequency and pitch salience variations, leading to an initial set of candidate notes. Since many of the obtained notes are irrelevant, short-duration, low-salience and octave-related notes are eliminated. Finally, the notes comprising the detected melody are extracted by selecting the most salient notes at each time.

For the sake of visualization simplicity, we will illustrate the method with a simple example: a monophonic saxophone riff.

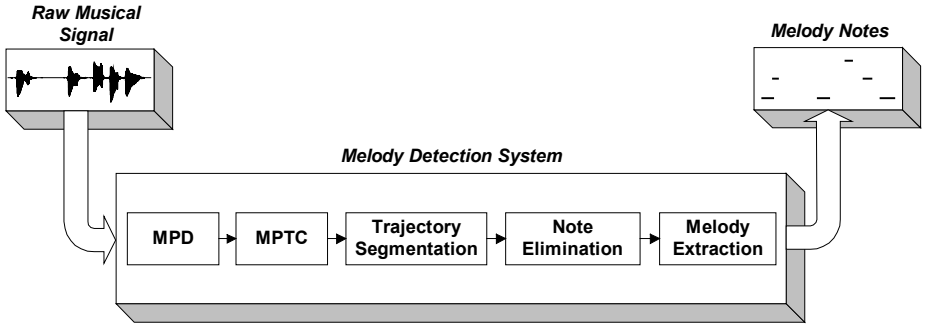

Fig. 1. Melody detection system overview

2.1 Multi-Pitch Detection

In the first stage of the algorithm, the objective is to capture a set of candidate pitches, which constitute the basis of possible future notes. The MPD algorithm receives as input a raw musical signal (monaural, sampling frequency f_s = 22050 Hz, 16 bits quantization) and outputs a set of pitch candidates and respective saliences.

Our goal is to obtain pitch candidates at each time instant. Since we cannot define instantaneous time in a computational model, we have to define some sort of time granularity. Therefore, we select a small enough time window and perform sound wave analysis in a frame-based way. We use a 20 ms frame length, which constitutes a good trade-off between time and frequency resolution: it is small enough for the assumption of signal stationarity and large enough for accurate detection of pitches above 100 Hz. The corresponding number of samples per frame is N = 441. In order to allow for a smooth transition between frames, 50% overlap is employed.

After dividing the musical signal into frames, we perform an auditory model based analysis of each frame, in order to detect the most salient pitches in each. This analysis comprises four stages, diagrammed in Fig. 2.: i) conversion of the sound waveform into auditory nerve responses for each frequency channel, using a model of the cochlea (which creates an image called cochleagram); ii) detection of the main periodicities in each frequency channel using auto-correlation (which produces an image called correlogram); iii) detection of the global periodicities in the sound waveform by calculation of a summary correlogram; and iv) detection of the pitch candidates in the frame by looking for the most salient peaks in the summary correlogram.

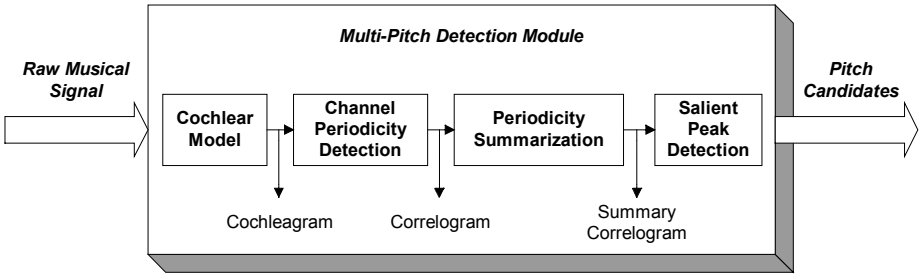

Fig. 2. Multi-pitch detection module

Cochlear Model. In the first stage of the multi-pitch detection system, a model of the cochlea is implemented, which aims to mimic the tasks carried out by the inner ear in the first stages of auditory processing. The cochlea encodes information in the sound wave into a multi-channel representation of auditory nerve firing patterns. The output of the cochlear model is a two-dimensional representation of a sound waveform that permits its visualization as a time-frequency image. In this image, called "cochleagram", each line contains information regarding auditory nerve responses for the corresponding cochlear, or frequency, channel. A good review of the tasks carried out in the cochlea and auditory nerve can be found in [8; 9].

In the present work, we use the cochlear model proposed by Richard Lyon [11] and described and implemented by Malcolm Slaney [16; 17], with some minor adaptations. Below, we give a short description of Lyon's model. For a thorough analysis, we refer the reader to [16].

The model implements three main tasks: filtering, detection and compression. First, a cascade of second-order filters models sound propagation down the basilar membrane, which acts as a frequency analyzer. Therefore, each filter corresponds to a cochlear channel that best responds to a particular frequency range. Furthermore, front filters are also implemented, which constitute a simple model of the responses of the outer and middle ears. In the present implementation, using a sampling frequency of 22050 Hz, 96 cochlear filters are used. Fig. 3 depicts every 5^{th} filter response, using a logarithmic frequency axis. This figure was created using Slaney's Auditory Toolbox [17]. As for model parameterization, we use the default parameters

proposed by Slaney, namely a filter Q of 8 and a step factor of 0.25 (which determines the amount of filter overlap).

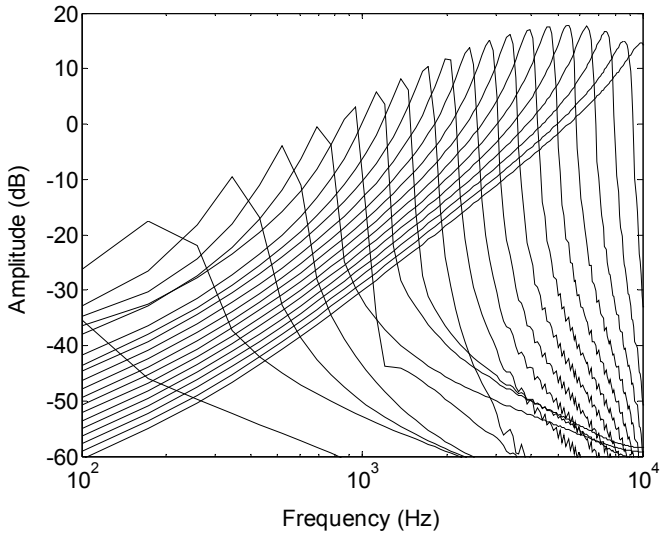

Fig. 3. Frequency response of cochlear filters

After filtering, the movements of the basilar membrane are converted to auditory nerve responses. Since inner hair cells only respond to movement in one direction, an array of half-wave rectifiers is employed to detect the output of each second order filter. This is a simple model of detection that does not account, for instance, for saturation effects.

Finally, four stages of automatic gain control compress the dynamic range of the input into a limited level that the auditory nerve can deal with. The automatic gain control is, in fact, a model of ear's adaptation: the response to a constant stimulus is first large and then, as the auditory system adapts to the stimulus, the response becomes smaller. Regarding parameterization, we use once again the parameters proposed by Slaney [17], namely target values of 0.0032, 0.0016, 0.0008 and 0.0004 and time constants of 640, 160, 40 and 10 ms for the first, second, third and fourth stages of automatic gain control.

Fig. 4 presents a cochleagram for our monophonic saxophone riff example: here, the harmonics of the sound waveform are clearly visible by the horizontal striations. Recall that higher channels correspond to lower frequencies. This picture has a limited time resolution, due to displaying purposes. However, the inner hair cells in the cochlea are extremely sensitive to the time structure of each component of the sound. Thus, a view of the cochleagram for a 20 ms' time slice is presented in Fig. 5b. In this figure, the harmonics are not so clear but a more precise image of auditory nerve firing responses in each channel is obtained.

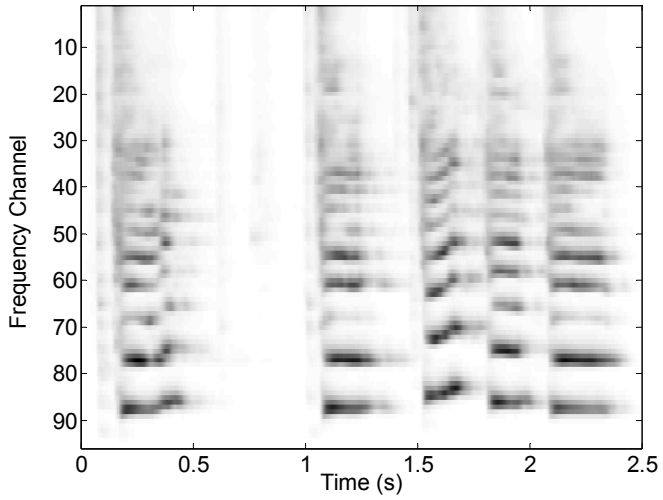

Fig. 4. Cochleagram for a 2.5s' saxophone riff

Channel Periodicity Detection. After computing the auditory nerve firing responses for each frequency channel, the main periodicities in the sound wave are detected. Here, this is accomplished by computation of the auto-correlation function (ACF) in each channel, resulting in a two-dimensional image of the sound signal, where the horizontal axis represents correlation lag and the vertical axis represents frequency. This image is called "correlogram" which means, literally, "picture of correlations" [16]. Each line of the correlogram contains information regarding the salience of the periodicities found for a given frequency channel. Like the cochleagram, the correlogram activity is measured by pixel intensity in the image.

The main objective of the correlogram is to summarize the temporal activity at the output of the cochlea [17]. In fact, many sounds, and particularly musical sounds, are periodic in time, or at least pseudo-periodic. The correlogram is, then, a powerful tool for detecting and visualizing the referred periodicities. As a result, all channels will show peaks at the horizontal positions corresponding to correlation lags that match the periods of repetition present in the signal.

Slaney [16] argues that the correlogram is biologically plausible. In fact, despite the separation of sound into broad cochlear channels, the temporal properties of the original signal are still kept. It is likely that the brain measures periodicities using a neural delay line, a case that is supported by the cross-correlator structures found in the brains of owls and cats. Furthermore, the detection of periodicities is also inspired by the "timing theory" of auditory nerve firing described above.

In terms of computer implementation, here, the periodicities in the cochleagram are obtained by computing the short-time ACF of the neural firing responses in each cochlear channels for a particular time window. As was referred previously, the

sound wave must be divided into frames where stationarity can be assumed. This is equivalent to multiplying the signal by a sliding rectangular window. However, in order to smooth the correlation, a Hamming window is used instead. In order to improve efficiency, the ACF in each window is implemented via de fast Fourier Transform (FFT) algorithm, which is equivalent to performing circular auto-correlation [18].

It is common to normalize the ACF so that its value at zero lag is equal to one, in order to reduce is dynamic range. However, this procedure eliminates any indication of the relative power in different cochlear channels. Therefore, the correlations are partially normalized by the square root of the power [17]. In this way, its dynamic range becomes comparable to the one of the cochleagram, keeping the relative powers between channels [16]. An example of a 20 ms' correlogram frame for our saxophone riff is presented in Fig. 5b. This picture shows the utility of correlograms for the analysis of periodic signals: there are clear vertical lines at particular autocorrelation lags, indicating instants when a large number of cochlear channels fire at the same period. This in turn is a clear indication of the pitch periods present in the signal.

Periodicity Summarization. As we referred above, the vertical lines across several cochlear channels show evidence of pitch. Therefore, a summary correlogram (SC) is built by summing the ACFs across all channels at each time lag. This measures the likelihood that a periodicity corresponding to a particular time lag is present in the sound waveform.

In order to enhance the peaks in the SC, we extract its envelope. This is carried out, typically, by half-wave rectifying and low-pass filtering the SC. We also perform squaring, so that the peaks are emphasized. Since the correlograms in the previous stage are all non-negative, the SC will also be non-negative and so does not need to be rectified. Therefore, only squaring and filtering are carried out. Moreover, unlike Slaney who normalizes the summary correlogram in each frame (dividing it by the value at zero lag) [17], we use its exact values, since they are useful for trajectory segmentation, as will be explained in Section 2.3.

An example of a summary correlogram is presented in Fig. 5d, where the determined pitch candidates are marked, as will be described below.

Salient Peak Detection. The final stage of the multi-pitch detection module consists of finding a set of pitch candidates based on the most salient peaks in the summary correlogram. To accomplish this task, we first look for all peaks in the SC, excluding the one at zero lag, and obtain their respective saliences, i.e., their amplitudes. Then, we eliminate all peaks that are not salient enough. To accomplish this task, we find the highest peak salience, *maxPeakSal*, and determine the minimum allowed peak salience, *minPeakSal*, using the minimum salience ratio parameter, *minSalRatio*

The detection of the main periodicities for our example is illustrated in Fig. 5d, where the most salient peaks, i.e., pitch candidates, are marked. The frequencies for the pitch candidates are then obtained by inverting the periods corresponding to the found peaks. Finally, the pitch saliences in all frames are normalized to the [0; 100] interval, for comparison in the following stages of the melody detection system.

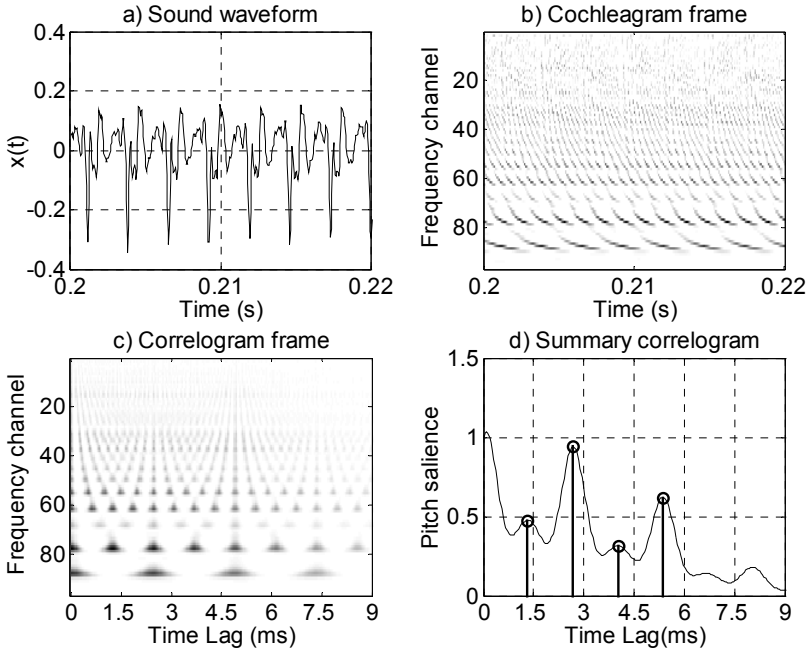

Fig. 5. Illustration of the four stages of the MPD algorithm

The four stages of the MPD algorithm are illustrated in Fig. 5: panel a) presents a 20 ms frame of our saxophone riff; panels b) and c) depict the corresponding cochleagram and correlogram images; and panel d) shows the summary correlogram, where the candidate pitch periods are marked. The summary correlogram was normalized for this graphic since it is easier to get a picture of pitch likelihoods, though our implementation does not perform normalization, as was referred before.

At this point, the motivation for extracting multiple pitches when we are only interested in the melodic line deserves a better explanation. In fact, extracting a single pitch would be both easier and more intuitive. However, since we are performing pitch detection in a polyphonic context, it often happens that the pitch corresponding to the melody is not the most salient one in each frame. Therefore, selecting several pitch candidates at this stage allows for the detection of lower-salience melody notes, which might not be captured if only a single pitch was extracted. We performed some experiments, to be reported in a future publication, which confirmed our assumption. The issue of note salience in a mixture of simultaneous notes is then dealt with in the following stages of the melody detection system.

The methodology for multi-pitch detection is summarized in Algorithm 1. Parameter definition is presented in Table 1. The parameters for the cochleagram are not presented, since we used the default values defined by Slaney [17], as referred above.

Algorithm 1. Multi-pitch detection

1. Compute the cochleagram for each time frame
   1.1. Apply Lyon's cochlear model
2. Compute the correlogram for each time frame
   2.1. Multiply each line of the cochleagram frame by a Hamming window
   2.2. Determine the ACF function for each channel via FFT
   2.3. Normalize the ACF
3. Compute the summary correlogram for the each time frame
   3.1. Sum the ACF across all channels
   3.2. Enhance peaks: squaring + low-pass filtering
4. Detect salient peaks in the summary correlogram
   4.1. Determine minimum allowed peak value (salience)
      - *maxPeakSal* ← maximum peak value
      - *minPeakSal* ← *maxPeakSal* × *minSalRatio*
   4.2. Eliminate pitches with low salience
      4.2.1. If peak salience < *minPeakSal*, eliminate peak
   4.3. Convert pitch periods to frequencies
5. Normalize pitch saliences in all frames to the [0; 100] interval
6. Return pitch frequencies and saliences for all frames.

Table 1. MPD parameters

Parameter Name	Parameter Value
frame length	20 ms
frame overlap	50%
minSalRatio	0.2

Unlike automatic music transcription systems, this algorithm does not deal with the well known and complex "octave problem". In fact, at this stage it is not important to analyze if a given pitch candidate corresponds to a real note or appears as a ghost note, whose fundamental frequency is a harmonic of some real note, a few octaves above. Some of the ghost notes will be eliminated already at this stage based on the pitch salience threshold, whereas others will be eliminated in the following stages of the melody detection algorithm.

2.2 Trajectory Construction

The second stage of the melody detection algorithm aims at creating a set of pitch trajectories, formed by connecting consecutive pitch candidates with similar frequencies. The idea is to find regions of stable pitches, which indicate the presence of musical notes. The MPTC algorithm receives as input a set of pitch candidates, charac-

terized by their frequencies and saliences, and outputs a set of pitch trajectories, which constitute the basis of the final melody notes.

We follow rather closely Serra's peak continuation algorithm [15]. However, since we have a limited set of pitch candidates per frame, our algorithm is much lighter. In fact, Serra looks for regions of stable sinusoids in the signal's spectrum, which leads to a trajectory for each harmonic component found. Therefore, a high number of trajectories have to be processed, which makes the algorithm much heavier, though the basic idea is the same. Another difference is that we first quantize frequencies to the closest MIDI note. We found that peak continuation based on MIDI note numbers allows for a more robust trajectory build up. One reason for this seems to come from the fact that the location of peaks oscillates somewhat due to interference from other sources in the sound mixture. Furthermore, the representation of notes using MIDI numbers simplifies an eventual representation of the sound waveform in MIDI format (e.g., for generation of a MIDI file).

This algorithm is based on the definition of a maximum frequency deviation (in semi-tones in our case) for continuing trajectories. We define a value of one semi-tone, motivated by the fact that some songs comprise glissando and vibrato regions, as well as by the frequency oscillations that may result from interference of other sources. Therefore, in this way, all these phenomena are kept within a common track, instead of being separated into a number of different trajectories, e.g., one trajectory for each note that one glissando may traverse. The drawback of allowing a larger frequency deviation is that a single trajectory can contain more than one note. This is the reason why we perform trajectory segmentation, in the next stage of the melody detection algorithm.

Also, we specify a maximum number of frames where a trajectory can be inactive, i.e., when no continuation peaks are found. If this number is exceeded, the trajectory is stopped. Here, we define a maximum of five inactive frames.

Finally, any trajectory must be longer than a minimum trajectory length. Therefore, all finished trajectories that are shorter then this threshold, are eliminated. The minimum trajectory length in our implementation is nine frames.

We present a detailed description of the implemented algorithm in [13].

The result of the MPTC algoritjm is illustrated in Fig. 6, for our saxophone riff example. There, we can see that some of the obtained trajectories comprise glissando regions. Also, some of the trajectories include more than one note and should, therefore, be segmented.

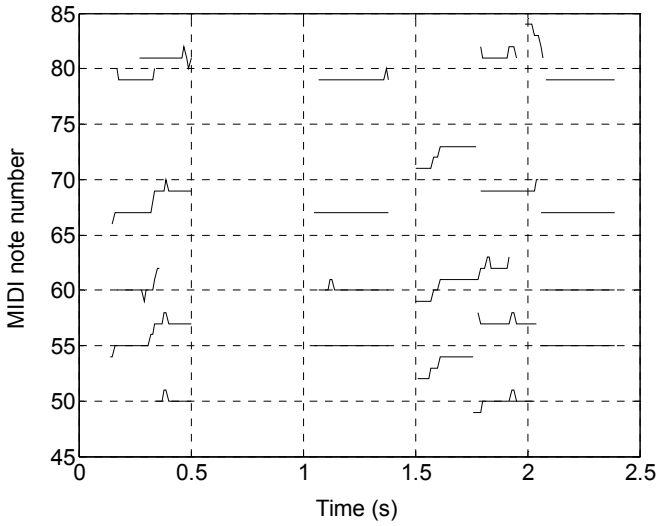

Fig. 6. Illustration of the MPTC algorithm

2.3 Trajectory Segmentation

As we mentioned previously, the trajectories that result from the MPTC algorithm may contain more than one note and, therefore, must be segmented. This is the task of the third stage of the melody detection method. The trajectory segmentation algorithm receives as input a set of trajectories of pitch candidates and outputs a set of segmented trajectories, i.e., note candidates.

Two types of segmentation have to be conducted. The most intuitive one is frequency segmentation, where the goal is to separate all the different frequency notes that are present in the same trajectory. The other one, salience segmentation, aims at separating consecutive notes that have the same fundamental frequencies, which the MPTC algorithm may have interpreted as forming only one note. This requires segmentation based on salience minima, which mark the limits of each note. Here, it is important to the say that the salience value depends both on the evidence of pitch for that particular frequency and on the intensity of the frequency component. Consequently, a salience curve consists of a growing region, corresponding to note onset, a more or less stable zone, corresponding to the steady part of the note, and a decreasing region, corresponding to note offset. Thus, notes can be segmented by detecting clear minima in the salience curve.

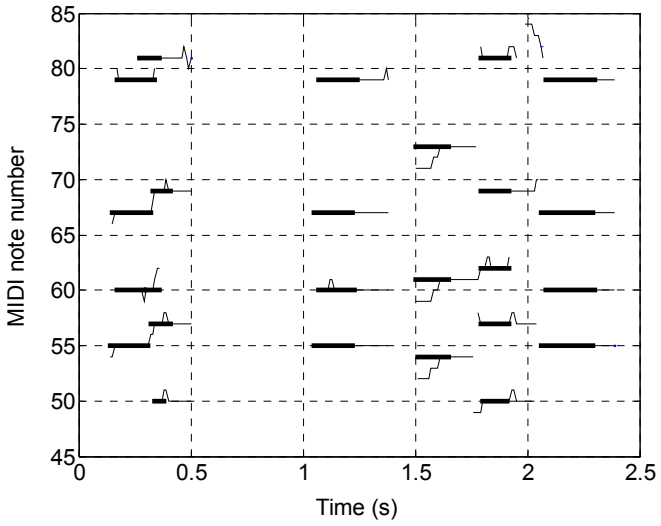

Fig. 7. Illustration of the trajectory segmentation algorithm

As for frequency segmentation, the main idea is to find sufficiently long sequences of the same note number. Only then trajectories are segmented. When note transitions are found but the current note sequence is not long enough, i.e., larger than nine frames (as defined in the MPTC algorithm), the trajectory is not segmented, since it may correspond to the start of a glissando region. Furthermore, when we find short sequences delimited by the same note number, e.g., {70, 71, 71, 71, 71, 70}, these are interpreted as possible modulation regions, and so no segmentation takes place.

After frequency segmentation, the obtained candidate notes must be analyzed so as to check whether they should be further divided. In fact, there may be consecutive distinct notes at the same fundamental frequency that, erroneously, form a unique long note. In this situation, those notes must be divided. In order to accomplish this task, salience segmentation takes place. The main idea is to find clear salience minima that suggest the presence of more than one note. This is implemented using a recursive procedure.

Finally, after all notes are segmented, their onset and offset times are adjusted. For each note, we get its maximum salience value and then define the onset as the first frame were the salience rises above 20% of the maximum salience found. The procedure is the same for the offsets, i.e., the note offset corresponds to the first frame where the salience rises above 20% of the maximum salience, starting from the end.

A detailed description of the implemented algorithm is presented in [13].

Fig. 7 illustrates trajectory segmentation, using the initial trajectories from the MPTC algorithm (Fig. 6). The obtained notes are depicted with thick lines. We can see that glissando and modulation regions are properly dealt with (check notes starting approximately at time 1.5s). Furthermore, some trajectories are truncating as a consequence of the assumption for offset detection.

2.4 Note Elimination

The objective of the fourth stage of the melody detection algorithm is to delete some of the note candidates, based on their saliences, durations and on the analysis of octave relations. The note elimination algorithm receives as input a set of note candidates and outputs a reduced set of notes, relevant for melody extraction.

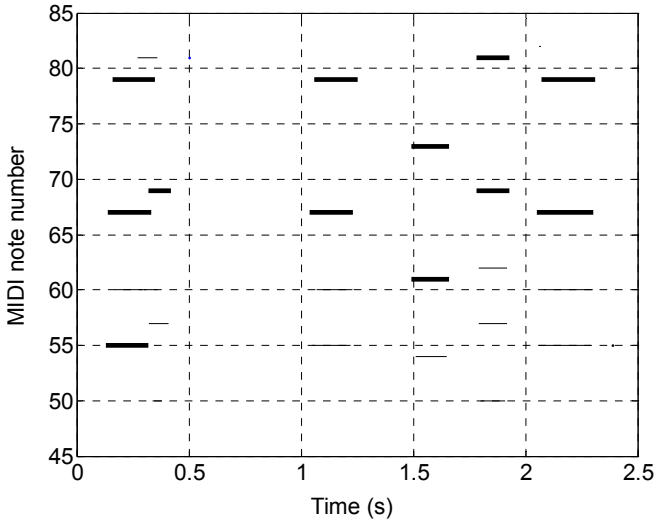

Fig. 8. Illustration of the note elimination algorithm

First, low-salience notes are deleted. A note is low-salience if its average salience is below a value of 20 and if the number of frames whose salience is above that threshold is not enough, i.e., less than five frames. Next, all the notes that are too short, i.e., whose duration is below the minimum of nine frames, defined in the MPTC algorithm, are also deleted. Finally, we look for octave relations between all notes, based on the fact that some of the obtained pitch candidates are sub-harmonics of real pitches in the sound wave. If two notes have approximately the same onset and offset times and are harmonically related, it is possible that the lower one is just a sub-harmonic of the higher one. Therefore, we compare their respective saliences in order to take a decision: if the salience of the lower note is less than 60% of the salience of the higher note, the lower one is eliminated. We describe this algorithm in greater detail in [13].

Fig. 8 illustrates note elimination, based on the note candidates of Fig. 7. The obtained notes are depicted with thick lines. It can be seen that many of the note candidates are eliminated at this point.

2.5 Melody Extraction

In the final stage of the present melody detection system, our goal is to obtain a final set of notes comprising the melody of the song under analysis. The melody extraction algorithm receives as input the set of notes returned by the note elimination algorithm and outputs the final melody notes.

This stage of the proposed system, being probably the most important one, is also the most difficult one to carry out. In fact, many aspects of auditory organization influence the perception of melody by humans, for instance in terms of the role played by the pitch, timbre and intensity content of the sound signal. In our approach, we do not attack the problem of source separation, as would normally be the case. Instead, we base our strategy on the assumption that the main melodic line often stands out in the mixture.

This algorithm starts by analyzing intersections between notes. The beginning and end of intersection regions is used to segment the sound signal, as illustrated in Fig. 9, where s_i stands for the i-th obtained segment.

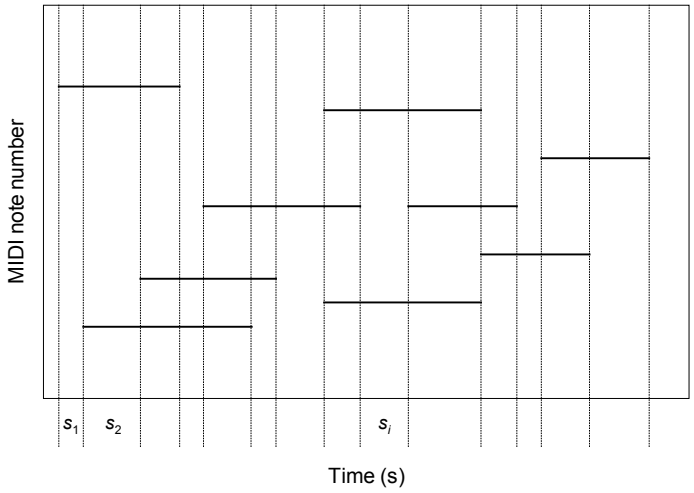

Fig. 9. Segmentation based on note intersection

Then, for each segment, we determine the three most salient notes, based on the average pitch salience of each note in each segment. Notes below MIDI note number 50 (143.83 Hz) are excluded. This procedure is motivated by the fact that the notes comprising the melody are, usually, in a middle frequency range.

Next, we eliminate all the notes that are not dominant, i.e., that are not in the three most salient notes for more than 2/3 of their total number of frames or do not have the highest salience for more than nine frames. Finally, we do not allow any simultaneous notes. Therefore, we truncate notes that end after the next note starts (or vice-versa, depending on their respective saliences in the common segments), eliminate notes included in larger duration notes and, for notes with approximate onsets and offsets, keep only the most salient one.

Our melody extraction algorithm is described in detail in [13].

Fig. 10 illustrates melody extraction, based on the example in Fig. 8. The final melody notes are depicted with thick lines.

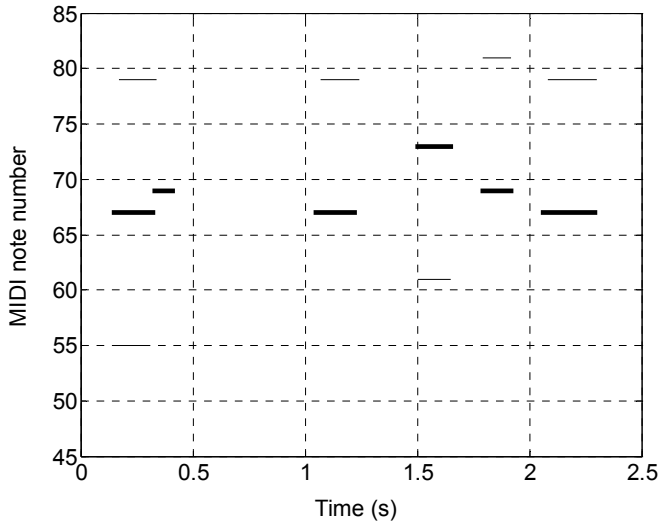

Fig. 10. Illustration of the melody extraction algorithm

3 Experimental Results

One difficulty regarding the evaluation of MIR systems results from the absence of standard test collections and benchmark problems. Therefore, we created our own test database, having care regarding its diversity and musical content. We collected excerpts of about 6 seconds from 12 songs, encompassing several different genres. The selected songs contain a solo (either vocal or instrumental) and accompaniment parts (guitar, bass, percussion, other vocals, etc.).

The obtained results are summarized in Table 2. There, "V" stands for vocals and "I" stands for instrumental.

Fig. 11 shows an example of the results of the melody detection system for an excerpt of the song "Thank You", by Dido.

Table 2. Results of the melody detection system

Song Title	Genre	Solo Type	#Total Notes	#Correct Notes
Pachelbel's Kanon	Classical	I	16	8 (50%)
Handel's Hallelujah	Choral	V	15	n. r.
Enya – Only Time	Neo-Classical	V	11	10 (90,9%)
Dido – Thank You	Pop	V	16	13 (81.25%)
Ricky Martin – Private Emotion	Pop	V	10	6 (60%)
Avril Lavigne – Complicated	Pop/Rock	V	14	9 (64.3%)
Rua Dona Margarida	Jazz	I	19	18 (94.7%)
Mambo Kings – Bella Maria de Mi Alma	Bolero	I	12	8 (75%)
Compay Segundo – Chan Chan	Latin	V	10	n. r.
Juan Luis Guerra – Palomita Blanca	Rumba	V	10	8 (80%)
Battlefield Band – Snow on the Hills	Scottish Folk	I	26	20 (76,9%)
Saxophone riff	(monophonic)	I	6	6 (100%)

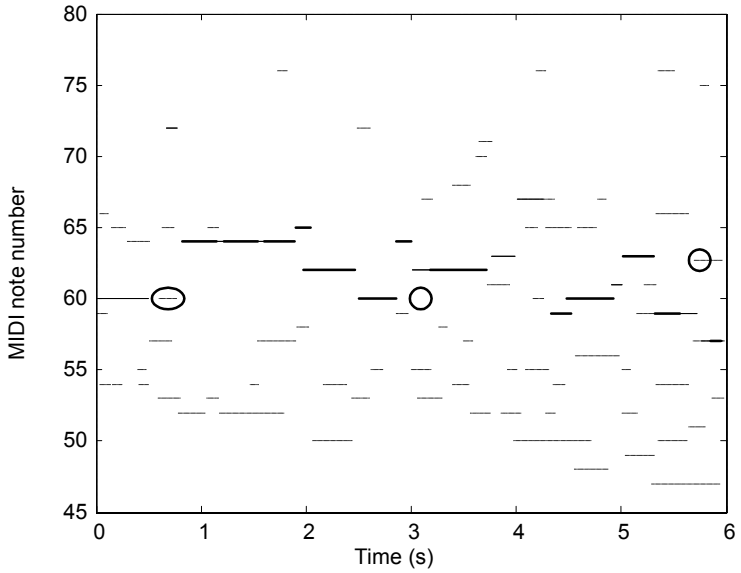

Fig. 11. Detected melody for "Dido - Thank You" excerpt

In this example in Fig. 11, we can see that the correct notes (thick lines) match the obtained melody notes (thin continuous lines) in most of the cases. The undetected notes are marked with circles. As can be seen, two of the three missing notes were

present in the notes obtained after elimination (dotted lines). One of the missing notes, approximately at time 5.8s, corresponds to erroneous trajectory segmentation. There are also other minor segmentation errors. The detected melody notes were compared with the correct notes, previously hand-labeled. In the absence of the melody line, the system detected the dominant accompaniment part, since sound sources are not discriminated. This can be seen in Fig. 11, by the thin continuous lines. This is consistent with the way humans seem to memorize melodies: a mix of solo regions with accompaniment regions, in the absence of a solo. However, we decided to ignore the notes where the accompaniment part dominates, in the same way as Goto does [7]. In order to extract only the melody, we would need a means of separating notes according to their sources. The most intuitive, but complex, way to accomplish this task would be to use timbre models. Other possibilities would be to separate notes according to their frequency ranges, note intensity levels (since the intensity of a solo varies usually in a smooth way) or duration of notes (e.g., it is not likely that a short duration note in the middle of two long notes belongs to the same source as them).

In our test cases, we observed that some of the notes were erroneously segmented (too much or too little segmentation) and others were shorter than the original ones. This resulted from noise in both the frequency and salience sequences, as well as frequency deviations in the MPTC algorithm, which lead to excessive trajectory segmentation. The noise in the salience sequences results often from interference from other sources in the sound mixture, namely percussive instruments. One possible way to deal with this issue would be to smooth the frequency and salience sequences before segmentation. Another possibility would be to filter out percussive sounds from the mixture, which seems to be a challenging task. We also observed a few semi-tone deviations (a small number of them). These errors resulted from the previous one and so should diminish after we deal with the problems coming from segmentation. We decided to ignore these small errors since our goal is to check whether a note is present or not, no matter in how many sub-notes the algorithm divides it. These errors can be reduced as was referred.

We can see that the algorithm could not find any reasonable melody in some excerpts ("#Correct Notes = n. r.: not reasonable"). However, in the cases where the melody stands clearly out of the background and percussion is not too intense, good results were achieved, which matches Goto's results [7]. In two of examples, we the system achieved an accuracy close to 100% (only one missing note). Furthermore, the results obtained for pop/rock and rumba songs surprised us positively, since they have strong percussion (Juan Luis Guerra), as well as intense guitars (Ricky Martion) with distortion (Avril Lavigne).

It is also worthwhile to say that, in the tested examples, many of the missing notes were still present after the note elimination stage. This suggests that a more robust melody extraction module could lead to better results.

We also tested our system with a simple monophonic saxophone riff, as referred throughout this paper. In this example, the results were very good in terms of detection of glissandos, vibratos and note onsets and offsets. Consequently, we hope our system could be used as a robust monophonic pitch detection tool.

4 Conclusions

We have presented a system for melody detection in polyphonic musical signals. This is a main issue for MIR applications, such as QBH "real-world" music databases. The work conducted in this field is presently restricted to the MIDI realm, and so we guess we make an interesting contribution to the area, though our results were not satisfactory enough for real applications. However, the achieved results are encouraging, since we have not exploited the full potential of our approach yet. Furthermore, to our knowledge, only Goto [7] addresses the issue of melody detection in polyphonic music, but without trying to explicitly extract notes. Also, our system is reasonably simple and light, except for the multi-pitch detection module, due to cochlear modeling and auto-correlation computation.

Regarding future work, we plan to further work out some of the described limitations, namely devising a more robust algorithm for melody extraction. Additionally, we plan to apply some sort of pre-processing in order to filter out percussive sound components, which seems a to be a very demanding task. Therefore, we plan to evaluate the feasibility of Independent Component Analysis for source separation. The main idea would be to separate the solo and accompaniment parts (namely, percussive ones) and then detect the melody in the solo part using our proposed approach.

5 Acknowledgements

This work was partially supported by the Portuguese Ministry of Science and Technology (MCT), under the program PRAXIS XXI. We also would like to thank Malcolm Slaney for making available the source code of his Auditory Toolbox.

References

1. Bainbridge, D., Nevill-Manning, C., Witten, I., Smith, L., McNab, R.: Towards a Digital Library of Popular Music. ACM International Conference on Digital Libraries (1999) 161-169
2. Bello, J. P., Monti, G., Sandler, M.: Techniques for Automatic Music Transcription. First International Symposium on Music Information Retrieval (2000)
3. Bregman, A. S.: Auditory Scene Analysis: the Perceptual Organization of Sound. MIT Press (1990)
4. Chai, W.: Melody Retrieval on the Web. MSc Thesis, Massachusetts Institute of Technology (2001)
5. Ellis, D.: Prediction-Driven Computational Auditory Scene Analysis. PhD Thesis, Massachusetts Institute of Technology (1996)
6. Ghias, A., Logan, J., Chamberlin D., Smith, B. C.: Query by Humming: Musical Information Retrieval in an Audio Database. ACM Multimedia Conference (1995)

7. Goto, M.: A Predominant-F0 Estimation Method for CD Recordings: MAP Estimation Using EM Algorithm for Adaptive Tone Models. IEEE International Conference on Acoustics, Speech and Signal Processing (2001)
8. Handel, S.: Listening – An Introduction to the Perception of Auditory Events. MIT Press (1991)
9. Hartmann, W. M.: Signals, Sound and Sensation. AIP Press (1997)
10. Klapuri, A.: Multipitch Estimation and Sound Separation by the Spectral Smoothness Principle. IEEE International Conference on Acoustics, Speech and Signal Processing (1982)
11. Lyon, R. F.: A Computational Model of Filtering, Detection and Compression in the Cochlea. IEEE International Conference on Acoustics, Speech and Signal Processing (2001) 1282-1285
12. Martin, K. D.: Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing. 3rd Joint Meeting of the Acoustical Societies of America and Japan (1996)
13. Paiva, R. P., Mendes, T., Cardoso, A.: A Methodology for Detection of Melody in Polyphonic Musical Signals. 116th Audio Engineering Convention (2004), to be presented
14. Scheirer, E. D.: Music-Listening Systems. PhD Thesis, Massachusetts Institute of Technology (2000).
15. Serra, X.: Musical Sound Modeling with Sinusoids Plus Noise. In: Roads, C., Pope, S., Picialli, A., De Poli, G. (eds.): Musical Signal Processing (1997)
16. Slaney, M., Lyon, R. F..: On the Importance of Time – A Temporal Representation of Sound. . In: Cooke, M., Beet, S., Crawford, M. (eds.): Visual Representations of Speech Signals (1993)
17. Slaney, M.: Auditory Toolbox: A Matlab Toolbox for Auditory Modeling Work (version 2). Technical Report, Interval Research Corporation (1998)
18. Smith, S.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing (1997)
19. Song, J., Bae, S. Y., Yoon, K.: Mid-Level Music Melody Representation of Polyphonic Audio for Query-by-Humming System. International Symposium on Music Information Retrieval (2002)