Segmentation of Pitch Tracks for Melody Detection in Polyphonic Audio

R. P. Paiva, T. Mendes, and A. Cardoso

CISUC – Centre for Informatics and Systems of the University of Coimbra
Department of Informatics Engineering, University of Coimbra (Polo II), P3030, Coimbra, Portugal
phone: + (351) 239 790 000, fax: + (351) 239 701 266, email: {ruipedro, tmendes, amilcar}@.dei.uc.pt
web: www.dei.uc.pt

ABSTRACT

We propose a method for segmentation of pitch tracks for melody detection in polyphonic musical signals. This is an important issue for melody-based music information retrieval, as well as melody transcription. Past work in the field addressed especially the issue of extracting melodic pitch lines, without explicit definition of notes. Thus, in this work, we propose a two-stage segmentation of pitch tracks for the determination of musical notes. In the first stage, frequency-based segmentation is conducted with recourse to frequency variations in pitch tracks. In the second phase, salience-based segmentation is performed in order to split consecutive notes with equal value, using pitch salience minima and note onsets.

1. INTRODUCTION

Query-by-humming (QBH) is a particularly intuitive way of searching for a musical piece, since melody humming is a natural habit of humans. This is an important research topic in an emergent and promising field called Music Information Retrieval (MIR). Several techniques have been proposed in order to attain that goal, e.g., [1]. However, presently, this work is being carried out only in the MIDI domain, which places important usability questions. Querying "real-world" polyphonic recorded musical pieces requires a melody representation of the songs, which is a complex task since there can be many types of instruments playing at the same time, whose spectra interfere severely with each other.

Only little work has been conducted in the problem of melody detection in polyphonic audio, [2, 3, 5, 6]. Additionally, most of the work is only concerned with the extraction of melodic pitch lines. In our approach, we propose a multi-stage strategy for melody detection, with explicit note determination. An overview of the system is given in Section 2. Segmentation of pitch tracks, the main topic of this paper, is the subject of Section 3. In Section 4, evaluation results are discussed. We finish with some conclusions.

2. MELODY DETECTION METHOD: OVERVIEW

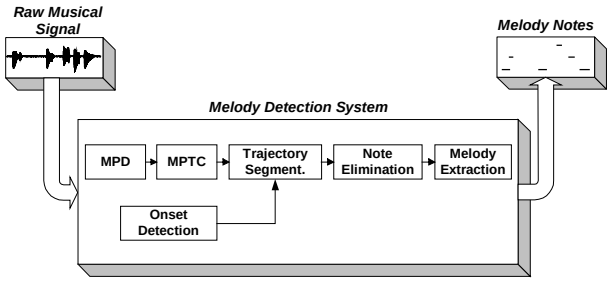

Fig. 1. Overview of the melody detection system.

Our melody detection algorithm comprises five modules, as illustrated in Fig. 1. The general strategy was described previously, e.g., [6], and, thus, only a brief presentation is provided here.

In the Multi-Pitch Detection (MPD) stage, the objective is to capture a set of pitch candidates, which constitute the basis of possible future notes. We use an auditory model for pitch detection.

Multi-Pitch Trajectory Construction (MPTC), in the second stage, aims to create a set of pitch tracks, formed by connecting consecutive pitch candidates with similar frequency values. The general idea is to find regions of stable pitches, which indicate the presence of musical notes.

The trajectories that result from the MPTC stage may contain more than one note and, therefore, must be segmented. This is the main topic of this paper and is described in the following section.

In the fourth stage, irrelevant note candidates are eliminated, based on their saliences, durations and on the analysis of harmonic relations. We make use of perceptual rules of sound organization, namely "harmonicity" and "common fate", where common frequency and amplitude modulation are exploited.

In the last stage, our goal is to obtain a final set of notes comprising the melody of the song under analysis. We base our strategy on two assumptions that we designate as the "salience principle" and the "melodic smoothness principle". In the salience principle, we take advantage of the fact that the main melodic line often stands out in the mixture. In the smoothness principle, we use the fact that small frequency intervals favour melodic coherence, since smaller steps in pitch hang together better.

3. SEGMENTATION OF PITCH TRACKS

As referred above, the trajectories obtained in the MPTC stage may contain more than one note and, so, must be segmented in order to explicitly obtain musical notes. Both frequency and pitch salience segmentation are carried out.

3.1 Frequency Segmentation

In frequency segmentation, the goal is to split notes with different values that may be present in the same trajectory, taking into consideration the presence of glissandos and frequency modulation.

The main issue with frequency segmentation is to approximate a frequency curve by piecewise-constant functions (PCFs), as a basis for track segmentation. However, this is not trivial, since musical notes, besides containing regions of approximately stable frequency, also contain regions of transition, where frequency evolves until (pseudo-)stability, e.g., glissando. Moreover, frequency modulation can occur, and so 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 PCFs that best approximate a frequency curve. As unknown variables we have the number of functions, their respective parameters (only bias, for constant functions), and start and end points.

We have investigated some approaches for piecewise-constant or linear function approximation. However, the algorithms that minimize the global approximation error either require an analytic expression of the curve or need to test different values for the number of functions, e.g., [7]. In this way, we propose an algorithm for approximation of frequency curves from musical notes by PCFs, taking advantage of some peculiarities of musical signals.

The algorithm starts by filtering the frequency curves of all tracks, in order to fill in missing values, as a result of the MPTC algorithm. This is carried out by a simple zero-order-hold (ZOH), as in (1). There, $f[k]$ is the frequency value of the k^{th} frame in the current track, in a total of N frames, and $f_F[k]$ denotes the filtered curve.

$$f_F[k]=\begin{cases} f[k], & if\ f[k]\neq 0 \\ f_F[k-1], & if\ f[k]=0 \end{cases},\ \forall_{k\in\{1,\cdots,N\}} \tag{1}$$

After that, the filtered frequency curve is approximated by PCFs through the quantisation of each frequency value to the corresponding MIDI note number, according to (2). There, $f_{MIDI}[k]$ represents the MIDI value corresponding to frequency $f_F[k]$ and F_{ref} is the frequency of MIDI note number zero.

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

Then, PCFs can be directly defined as sequences of constant MIDI values. Generally, it comes (3), where, PC_i represents the i^{th} PCF, defined in the domain D_i and characterized by a sequence of constant MIDI values equal to c_i. Also, the special case of singleton domains is allowed. The total number of PCFs is denoted by *nPC*.

$\forall_{i\in\{1,\cdots,nPC\}}$,

$$1:\ D_i=\{a_i,\cdots,b_i\}=\left\{\begin{array}{l} k\in\{1,\cdots,N\}: f_{MIDI}[k]=c_i \ \text{ and} \\ f_{MIDI}[k]=f_{MIDI}[k\pm 1] \end{array}\right\} \tag{3}$$

$$2:\ PC_i[k]=c_i\quad,\forall_{k\in D_i}$$

However, due to frequency variations resulting from modulation, as well as frequency errors from the MPD stage, fluctuations of MIDI values may be present. Also, glissandos should be kept within one single function. Therefore, $f_{MIDI}[k]$ must be filtered, so as to allow for a more robust determination of PCFs that may represent musical notes. Three stages of filtering are applied in order to appropriately deal with too short note candidates. Hence, PCFs whose length (i.e., the number of elements in the domain) is below the *minNLen* parameter (22 frames ≈ 120ms) are dealt with.

In the first filtering stage, the general idea is that short sequences delimited by long sequences with the same note number, e.g., {70,…,70,71,71,70,70,71,70…,70}, are interpreted as possible frequency modulation regions. So, they should be filtered, keeping the value of the delimiting sequence (70 in this example). Formally, it comes (4), where $L(PC_i)$ denotes the length of the i^{th} PCF. Thus, the support of some PCFs is enlarged, while others are eliminated.

$\forall_{i\in\{1,2,\cdots,nPC:L(PC_i)\geq minNoteLen\}}$,

$$\text{if}\ \left(\begin{array}{l} \exists_{m\in\{i+2,\cdots,nPC\}}: c_m=c_i \ \text{ and} \\ \sum_{j=i+1}^{m-1} L(PC_j)<minNLen \end{array}\right)\Rightarrow \tag{4}$$

$$1:\ D_i=\{a_i,\cdots,b_m\}$$

$$2:\ \forall_{j\in\{i+1,\cdots,nPC\}}, PC_j=PC_{j+m-i},\ nPC^{new}=nPC-m+i$$

However, some short PCFs are still kept. Therefore, a similar filtering is applied, much in the same way as in (4), with the difference that no long sequences need to be found.

At this moment, the only short PCFs present correspond to glissandos. Hence, PCFs forming note transitions are merged. Formally, it comes (5). There, *LN* contains LN_1, i.e., the index of the first long PCF in a transition, and m denotes the index where the glissando stops. Thus, m corresponds either to LN_1 or to the last PCF, if no long notes are found. In the former, the resulting PCF receives the value of the long PCF, since its length gives strong evidence that the transition finishes there. In the latter, the final MIDI values may have resulted from frequency drifting in the offset and, so, the resulting PCF receives the value of the longest PCF in the transition.

$\forall_{i\in\{1,\cdots,nPC:L(PC_i)<minNLen\}}$,

$$1:\ LN=\left\{\begin{array}{l} k\in\{i+1,\cdots,nPC\}: L(PC_k)\geq minNLen \ \text{ and} \\ \forall_{j\in\{i+1,\cdots,k-1\}}, L(PC_j)<minNLen \ \text{ and} \\ \text{sgn}(c_{j+1}-c_j)=\text{sgn}(c_j-c_{j-1}) \end{array}\right\}$$

$$2:\ m=\begin{cases} LN_1, & LN\neq\varnothing \\ nPC, & LN=\varnothing \end{cases} \tag{5}$$

$$3:\ D_i=\{a_i,\cdots,b_m\}$$

$$4:\ c_i=\begin{cases} c_m, & L(PC_m)\geq minNLen \\ c_j: j=\arg\max_{j=i}^{m}\left(L(PC_j)\right), & otherwise \end{cases}$$

$$5:\ \forall_{j\in\{i+1,\cdots,nPC\}}, PC_j=PC_{j+m-i},\ nPC^{new}=nPC-m+i$$

Finally, the precise timings for each PCF must be adjusted. In fact, as a result of MIDI quantisation, there is a delay in the moment where transitions start, since the frequencies at the beginning of a transition may be converted to the MIDI value of the first note, instead of the next one. In this way, we define the start of a transition as the point of maximum derivative of $f[k]$ after it starts to move towards the next note, i.e., the point of maximum derivative after the last occurrence of the median value of $f[k]$ in D_i, md_i. It then comes (6), where $\dot{f}[k]$ represents the derivative of $f[k]$, $last_i$ stands for the last index of the i^{th} PCF where md_i occurs, and dir_i denotes the direction of frequency movement from PCF i to i+1.

$\forall_{i\in\{1,\cdots,nPC-1\}}$,

$$1:\ md_i=\text{median}\left(f[k]\right)\quad,\forall_{k\in D_i: f_{MIDI}[k]=c_i \text{ and } f[k]\neq 0}$$

$$2:\ last_i=\left\{k\in D_i: f[k]=md_i \text{ and } \forall_{m>k,m\in D_i}, f[m]\neq f[k]\right\}$$

$$3:\ dir_i=\text{sgn}(c_{i+1}-c_i) \tag{6}$$

$$4:\ b_i=\left\{k\in D_i: k=\arg\max_{k=last_i}^{b_i}\left(\dot{f}[k]\right) \ \text{and}\ \text{sgn}\left(\dot{f}[k]\right)=dir_i\right\}$$

$$5:\ a_{i+1}=b_i+1$$

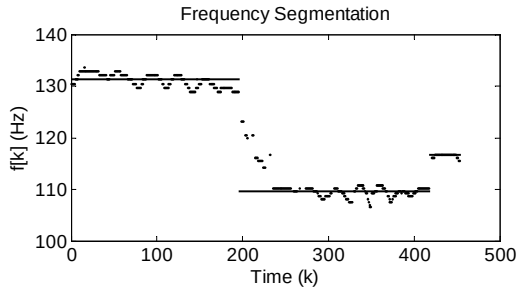

Fig. 2. Illustration of frequency segmentation.

The algorithm for frequency segmentation is illustrated in Fig. 2, for a pitch track from The Mambo King's "Bella Maria de Mi Alma". There, the constant lines represent the obtained PCFs.

3.2 Pitch Salience Segmentation

As for pitch salience segmentation, our goal is to separate consecutive notes with the same value, which the MPTC algorithm may have interpreted as forming one single note. This requires segmentation based on pitch salience minima, which mark the limits of each note. In fact, the salience value depends on the evidence of pitch for that particular frequency, which is lower at the onsets and offsets. Consequently, the envelope of the salience curve is similar to an amplitude envelope: it grows at the note onset, has then a more steady region and decreases at the offset. In this way, notes can be segmented by detecting clear minima in the pitch salience curve.

As in the frequency segmentation stage, the algorithm starts by filtering the salience curve with a ZOH, due to missing values. Additionally, as the salience curve may be somewhat noisy, we add a low-pass filtering stage in order to smooth it. A zero-phase Blackman-sinc filter [9] is used, as in (7), where $s[k]$ is the k^{th} pitch salience value in the current track, '*' denotes discrete convolution, $b[n]$ represents the Blackman window [9], centred at the origin, and $s_F[k]$ is the smoothed curve. The parameters f_c = 100Hz and W = 9 stand for the cutoff frequency and the length of the filter kernel, respectively. K is chosen so as to provide unity filter gain at DC.

$$s_F[k] = s[k] * h[k]$$
$$h[n] = K \cdot \frac{\sin(2\pi f_c n)}{n\pi} \cdot b[n], \quad n \in \{-W/2, \cdots, W/2\} \tag{7}$$

After that, $s_F[k]$ is normalized into the [0, 1] interval, according to (8), in order to allow for a uniform valley search for the whole set of tracks. There, $s_N[k]$ denotes the normalized curve.

$$s_N[k] = \frac{s_F[k] - \min(s_F[k])}{\max(s_F[k]) - \min(s_F[k])} \tag{8}$$

Then, we iteratively look for all clear local minima and maxima of $s_N[k]$. First, all local minima and maxima are found, as in (9). There, the set $LMin^{(i)}$ contains all local minima found in the curve at iteration i. We also deal with plateaus, where the index of the minimum will correspond to the middle point of the plateau. Eq. (9) is slightly changed for the border frames in the interval $I^{(i)}$, so that only one-side comparisons are performed. As for local maxima, the $LMax^{(i)}$ set is obtained likewise, by changing the relational operator. In the first iteration of the algorithm, $I^{(0)}$={1,2,…,N}, i.e., the first interval contains all the frames in the track.

$$LMin^{(i)} = \left\{ k \in I^{(i)} : s_N[k] < s_N[k-1] \text{ and } s_N[k] < s_N[k+1] \right\} \tag{9}$$

Clear minima are then recursively looked for. We start by finding the global minimum of $s_N[k]$, $g_{min}{}^{(i)}$, as in (10). We also define $k_{min}{}^{(i)}$ as the index corresponding to $g_{min}{}^{(i)}$, at iteration i.

$$g_{min}{}^{(i)} = \min_{k \in LMin^{(i)}} \left(s_N[k] \right) \tag{10}$$

Next, $LMax^{(i)}$ is divided into two subsets, one to the left of $k_{min}{}^{(i)}$, $LMax_{left}{}^{(i)}$, and another to its right, $LMax_{right}{}^{(i)}$. The global maxima for each subset, $g_{max\text{-}left}$ and $g_{max\text{-}right}$, respectively, are then calculated. Formally, it comes (11) and (12), for the left subset. The procedure is analogous for the right subset.

$$LMax_{left}{}^{(i)} = \left\{ k \in LMax^{(i)} : k < k_{min}{}^{(i)} \right\} \tag{11}$$

$$g_{max\text{-}left}{}^{(i)} = \max_{k \in LMax_{left}{}^{(i)}} \left\{ s_N[k] \right\} \tag{12}$$

Then, $k_{min}{}^{(i)}$ is selected as a clear minimum if its prominence, i.e., the minimum distance from $g_{min}{}^{(i)}$ to both the left and right global maxima, is above *minPvd*. Finally, $LMin^{(i)}$ is also divided into two new intervals, to the left and right of $k_{min}{}^{(i)}$, in the same way as $LMax^{(i)}$ was. Eqs. (9) to (12) are repeated recursively for each new interval, $I^{(\cdot)}$, until all clear minima are found. Formally, we define the set of indexes corresponding to clear minima, *CMin*, (13):

$$CMin = \left\{ \begin{array}{l} k \in \bigcup_{i=1}^{nI} k_{min}{}^{(i)} : vm_k \geq minPvd, \\ vm_k = \min\left(\left| g_{min}{}^{(i)} - g_{max\text{-}left}{}^{(i)} \right|, \left| g_{min}{}^{(i)} - g_{max\text{-}right}{}^{(i)} \right| \right) \end{array} \right\} \tag{13}$$

Here, vm_k is the valley magnitude (i.e., the minimum distance between a valley and its adjacent peaks) of the k^{th} global valley found in the successive intervals $I^{(i)}$ and nI is the total number of iterations.

However, this procedure lacks robustness, since the best value for *minPvd* varies from track to track, along different song excerpts. In fact, 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. Thus, we improved the method by performing onset detection and matching the obtained onsets with the initial candidate segmentation points. In this way, we specified a low value for *minPvd*, namely 0.1, so that missing segmentation points are unlikely. Consequently, extra false segmentation points appear, which are eliminated later on via onset matching.

Fig. 3 illustrates our prominent valley detection algorithm for a pitch track of Claudio Roditti's "Rua Dona Margarida", where 'o' represents correct segmentation candidates and '*' denotes extra segmentation points.

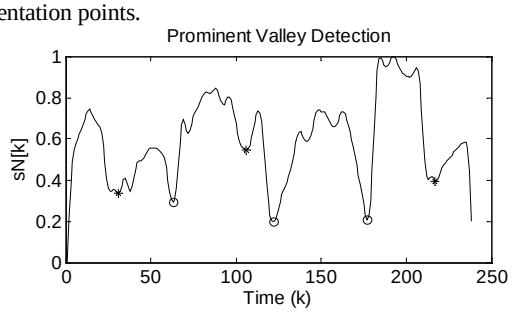

Fig. 3. Pitch salience segmentation: candidate points.

As for onset detection, we based ourselves on [4] and [8], where onsets are detected following a band-wise processing approach. A bank of nearly critical band filters is chosen, which covers the frequencies from 44Hz to the Nyquist frequency, in a total of 18 filters. Elliptic filters are employed, so as to guarantee a maximally sharp cutoff in the transition band. Since it is important to maintain the temporal properties of the signal in each band, zero-phase should be a requirement. Thus, we perform bi-directional filtering [9], (14):

$$S_B^i(z) = S(z) * H^i(z) * H^i(z^{-1}) \tag{14}$$

Here, $S_B^i(z)$, $H^i(z)$ and $S(z)$, represent, respectively, the filtered output and the filter transfer function at band i, and the original signal, all in the Z-domain. Due to bi-directional filtering, we specified filter parameters in terms of the desired transfer function: 3rd order filters, with 1.5dB ripple in the pass-band and 20dB of rejection in the stop-band. The design parameters are approximately doubled as a result of bi-directional filtering, e.g., 6th order filters result. As for the cutoff frequencies, the lowest three filters are one-octave band-pass filters, whereas the remaining are third-octave band-pass filters, with no overlapping. Bi-directional filtering slightly changes the cutoff frequencies, which was not problematic in this case.

After filtering, onset components are computed at each band. First, we extract the amplitude envelope of each output via rectify-and-smooth [8]. In order to ease calculations, the output of each band is decimated to 200 Hz. Then, the outputs are full-wave rectified and smoothed with a 100ms zero-phase half-Hanning window [8], $w[n]$, of corresponding width $W/2$, as in (15).

$$s_H^i[k] = \left| s_B^i[k] \right| * w[k]$$
$$w[n] = K \cdot 0.5 \cdot \left[1 - 0.5\cos\left(\frac{2\pi\left(n + W/2\right)}{W} \right) \right], \ \forall_{n \in \{-W/2, \cdots, 0\}} \quad (15)$$

Then, onset components are detected in each band by finding the points of maximum slope in the smoothed curves. In other words, we look for clear maxima in the derivatives of the smoothed outputs $s^i_H[k]$. This is accomplished in a similar way to eqs. (8) to (13) above, except that now we look for peaks instead of valleys. Since some onset candidates may be very close, we delete the ones that are closer than 50ms to a more intense component [4].

Next, we merge the onset components found at each band. These are all sorted in time order and combined by summing the magnitudes of all onsets in a 50ms' neighbourhood of a given candidate, as in (16). There, *OM*[*j*] and *OI*[*j*] denote, respectively, the magnitude and index of the j^{th} onset component, in a total of *nO* components, and Δt represents the time interval (seconds) corresponding to a given number of frames. Finally, onset components closer than 50ms to a more intense component are again eliminated.

$$\forall_{j,k \in \{1,\cdots,nO\}}, \ OM[j] = \sum_{k:\Delta t\left(OI[k] - OI[j]\right) \le 0.05} OM[k] \quad (16)$$

After onset detection, our goal is to validate the candidate segmentation points obtained above. First, all clear salience minima are kept as definitive segmentation points, according to (17), where *VCMin* denotes the set of clear salience minima, i.e., valleys whose prominence is above *clearValley* = 0.4.

$$VCMin = \left\{ k \in CMin : vm_k \ge clearValley \right\} \quad (17)$$

Then, for all unclear valleys, onset matching is performed. Hence, if a candidate valley has a clear onset closer than 50ms, that valley is kept as a segmentation point, as in (18). There, *clearOn* = 0.4 is the threshold for definition of clear onsets and *VUMin* denotes the set of unclear salience minima. Finally, the set of all segmentation points is defined as *VCMin* ∪ *VUMin*.

$$VUMin = \left\{ \begin{array}{l} k \in CMin : vm_k < clearValley \ \text{ and } \ \exists_{j \in \{1,\cdots,nO\}} : \\ OM[j] \ge clearOn \ \text{ and } \ \Delta t\left(k - OI[j]\right) \le 0.05 \end{array} \right\} \quad (18)$$

Another advantage of onset detection is that note beginnings are adjusted to the found onsets, for a maximum deviation of 50ms.

The described procedure for pitch salience segmentation is illustrated in Fig. 4, for an excerpt from Claudio Roditti's "Rua Dona Margarida". The grey horizontal lines represent the original annotated notes, whereas the black lines denote the extracted notes. The small grey vertical lines stand for the correct segmentation points and the black vertical ones are the obtained results of our algorithm. It can be seen that there is an almost perfect match in this excerpt.

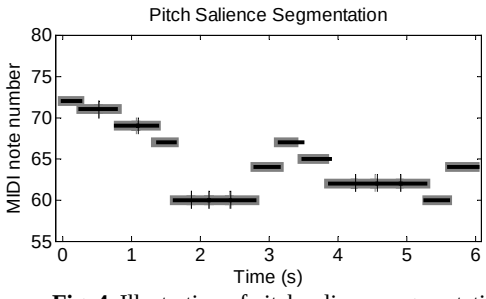

Fig. 4. Illustration of pitch salience segmentation.

4. VALIDATION EXPERIMENTS

In order to evaluate the melody detection system, and particularly the segmentation algorithm, we collected excerpts of about 6 seconds from 12 songs, encompassing several different genres (pop, rock, jazz, classical, latin, …) [6]. The selected songs contain a solo (either vocal or instrumental) and accompaniment parts (guitar, bass, percussion, other vocals, etc.). Also, the selected excerpts contain glissando and frequency modulated notes, as well as consecutive notes with the same MIDI value.

The results for frequency segmentation were very good. All glissandos and frequency-modulated notes were correctly captured (e.g., Fig. 2), except for a short ornamental note found in one excerpt from Battlefield Band's "Snow on the Hills". As for the timings, they matched very well our manually annotated database. Most deviations were smaller than 20ms, which may even have resulted from annotation errors. Only a few higher deviations occurred in tracks with transitions zones with many missing frequency values.

As for pitch salience segmentation, the results were also generally good (e.g., Figs. 3 and 4). However, some tracks were more problematic, as a result of missing and extra onsets in some excerpts (e.g., Enya's "Only Time", with a lot of reverb). Onset detection in polyphonic recordings is itself a complex task, and so salience segmentation may be improved if onset detection algorithms become more reliable.

As a final word, our melody extraction system obtained 87.52% average pitch detection accuracy for the melody notes. Some extra notes are still present, a problem that will be addressed in the future. Yet, the obtained results are encouraging.

5. CONCLUSIONS

We proposed a method for segmentation of pitch tracks as a requirement for melody detection in polyphonic musical signals. The obtained results were very good, especially for frequency segmentation. Pitch salience segmentation may be improved as soon as more robust onset detection algorithms become available.

6. ACKNOWLEDGEMENTS

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

REFERENCES

[1] D. Bainbridge, C. Nevill-Manning, I. Witten, L. Smith and R. McNab, "Towards a digital library of popular music", in *Proc. ACM Intl. Conference on Digital Libraries*, 1999, pp 161–169.

[2] J. Eggink and G. J. Brown, "Extracting melody lines from complex audio", in *Proc. Intl. Conf. on Music Information Retrieval - ISMIR 2004*, 2004.

[3] M. Goto, "A predominant-F0 estimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models", in *Proc. IEEE ICASSP 2001*, 2001.

[4] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge", in *Proc. IEEE ICASSP 1999*, 1999.

[5] M. Marolt, "On finding melodic lines in audio recordings", in *Proc. Intl. Conf. on Digital Audio Effects – DAFX 2004*, 2004.

[6] R. P. Paiva, T. Mendes, and A. Cardoso, "An auditory model based approach for melody detection in polyphonic musical recordings", in *Computer Music Modelling and Retrieval – CMMR 2004*, U. K. Wiil, ed. LNCS (to appear).

[7] J.-C. Pérez and E. Vidal, "An algorithm for the optimum piecewise linear approximation of digitized curves", in *Proc. Intl. Conf. on Pattern Recognition – ICPR 1992*, 1992, pp. 167–170.

[8] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals", *Journal of the Acoustical Society of America*, Vol. 103, No. 1, pp. 588–601, Jan. 1998.

[9] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. Address: California Technical Publishing, 1997.