An Audio Front End for Query-by-Humming Systems

Goffredo Haus Emanuele Pollastri
L.I.M.-Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione, Università Statale di Milano
via Comelico, 39; I-20135 Milan (Italy)
+39-02-58356222 +39-02-58356297
haus@dsi.unimi.it pollastri@dsi.unimi.it

ABSTRACT

In this paper, the problem of processing audio signals is addressed in the context of query-by-humming systems. Since singing is naturally used as input, we aim to develop a front end dedicated to the symbolic translation of voice into a sequence of pitch and duration pairs. This operation is crucial for the effectiveness of searching for music by melodic similarity. In order to identify and segment a tune, well-known signal processing techniques are applied to the singing voice. After detecting pitch, a novel post-processing stage is proposed to adjust the intonation of the user. A global refinement is based on a relative scale estimated out of the most frequent errors made by singers. Four rules are then employed to eliminate local errors. This front end has been tested with five subjects and four short tunes, detecting some 90% of right notes. Results have been compared to other approximation methods like rounding to the nearest absolute tone/interval and an example of adaptive moving tuning, achieving respectively 74%, 80% and 44% of right estimations. A special session of tests has been conducted to verify the capability of the system in detecting vibrato/legato notes. Finally, issues about the best representation for the translated symbols are briefly discussed.

1. INTRODUCTION

In the last few years, the amount of bandwidth for multimedia applications and the dimension of digital archives have been continuously growing, so that accessibility and retrieval of information are becoming the new emergency. In the case of digital music archive, querying by melodic content received a lot of attention. The preferred strategy has been the introduction of query-by-humming interfaces that enable even non-professional users to query by musical content. A number of different implementations has been presented since the first work by Ghias et al. [4] and a brief overview is introduced in the next section. In spite of this fact, the digital audio processing of an hummed tune has been tackled with naive algorithms or with software tools available on the market. This fact results in a poor performance of the translation from audio signals to symbols. Furthermore, previous query-by-humming systems can be hardly extended to handle sung queries (i.e. with lyrics) instead of hummed queries.

The quality of a query-by-humming system is strictly connected to the accuracy of the audio translation. It is well known that the amount of musical pieces retrieved through a melody grows when the length of the query decreases [8, 12, 13, 22]. Employing representations like the 3-level contour will further lengthen the list of matched pieces. In the same time, we can not expect users to search through very long queries (more than twenty notes long) or to sing perfectly, without errors and approximations. Interval representations show another source of errors, since a misplaced note propagates to the contiguous one. Thus, an accurate translation of the input is surely a basic requirement for every query-by-humming system.

In this paper, we propose an audio front end for the translation of acoustic events into note-like attributes and dedicated to the singing voice. We will focus on the post-processing of the voice in order to minimize the characteristic errors of a singer. In other words, the audio processing will be conducted in a user-oriented way, that is, trying to understand the intention of the singer. This work follows the one presented in [5] where some preliminary work and experiments have been briefly illustrated.

2. RELATED WORK

There are many techniques to extract pitch information from audio signals, primarily developed for speech and then extended to the music domain. The detection of pitch from monophonic sources is well understood and can be easily accomplished through the analysis of the sampled waveform, the estimation of the spectrum, the autocorrelation function or the cepstrum method.

Previous query-by-humming systems employed some basic pitch tracking algorithms with only little pre- and post- processing, if any. For example, Ghias et al. performed pitch extraction by finding the peak of the autocorrelation of the signal [4], McNab et al. employed the Gold-Rabiner algorithm [12], while Prechelt and Typke looked for prominent peaks in the signal spectrum [16]. Rolland et al. [19] applied an autocorrelation algorithm with heuristic rules for post-processing. Some works focused mainly on the matching and indexing stages of the query-by-humming, using software tools available on the market for the audio translation [3,7].

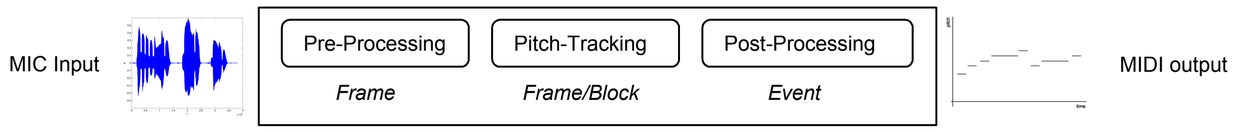


Figure 1. Architecture of the system developed.

Outside of the Music Information Retrieval community, the analysis of the singing voice constitutes an established research field, especially in the framework of voice analysis/re-synthesis. Typical examples are the voice morphing system by Loscos et al. [10], the structured audio approach to singing analysis score driven by Kim [6] and the synthesis of voice based on sinusoidal modeling by Macon et al. [11].

3. BACKGROUND

Despite its monophonic nature, singing has proved to be difficult to analyze [21]. The time-varying spectral characteristics of voice are similar during speech and singing. In both cases, we can divide the generated sounds in voiced and unvoiced[1]. In order to have an approximate idea of this property, we can think of the former kind of sounds as consonants[2] and the latter as vowels. Since voiced sounds are constituted by periodic waveform, they are easier to analyze, while unvoiced sounds have a state similar to noise. Luckily, during singing the voiced properties are predominant and contain what we call musical pitches. However, the information held by unvoiced regions are important as well, since they often contain the rhythmic aspect of the performance. Unlike speech, the singing voice shows a slowly-changing temporal modulation both in the pitch and in the amplitude (vibrato). In addition to these acoustic properties, singing voice analysis should deal with human performance that is typically affected by errors and unstable. Previous researches revealed that errors remain constant regardless of the note distance in time and in frequency [9]. We will follow these findings in the post-processing step of the proposed front end.

4. VOICE PROCESSING

An audio front end for a query-by-humming/singing system should contain all the elements needed to perform the transformation from audio to symbols, where audio is the singing voice and symbols are the most likely sequences of notes and durations. It should be able to adapt to the user automatically, i.e. without any user-defined parameter settings. Further, it should not require a particular way of singing, like inserting some little pause between notes or following some reference musical scale or metronome. In a query-by-singing application, the last requirements are important to avoid limiting the number of potential users, who are expected to be most non-professional users [5].

[1] A more rigorous definition is the following: “speech sounds can be voiced, fricative (or unvoiced) and plosive, according to their mode of excitation” [18]. In the present paper, plosive and fricative sounds will be grouped into the unvoiced category.

[2] with the exception of [m][n][l] which are voiced.

We suggest to elaborate the audio signal at three different levels of abstraction, each one with a particular set of operations and suitable approximations:

1- event

2- block

3- frame

At the event level, we estimate starting/ending points of musically meaningful signal, signal gain and, as a last step of computation, pitches and durations. At the block level, a background noise threshold is determined, voiced-unvoiced segments are isolated and pitches are approximated; eventually, effects of vibrato or bending are eliminated. At a frame level, we estimate spectrum, zero crossing rate, RMS power and octave errors. From the above observations, we derived an architecture (Figure 1) in which every uncertainty about the audio signal is resolved with subsequent approximations. The elaboration path is divided into three stages; details of each stage are presented in the following sections. The system developed is designed for offline voice processing and is not currently developed for real-time operations. Thus, audio is captured from a microphone, stored as wave file with sampling frequency of 44100 samples/sec and 16 bit of quantization, and then analyzed.

4.1 Pre-Processing

The first purpose of the audio front end is to estimate the background noise. We evaluate the RMS power of the first 60 msec. of the signal; a threshold for the Signal/Noise discrimination is set to a value of 15% above this level (S/N threshold). If this value is above –30dB, the user is asked to repeat the recording in a less noisy room. Otherwise, two iterative processes begin to analyze the waveform, one from the beginning and another from the end. Both the processes perform the same algorithm: the RMS power of the signal is calculated for frame 440 samples long (about 10 msec.) and compared with the S/N threshold. To avoid the presence of ghost onsets caused by impulsive noise, the value of the n-th frame is compared to the (n+4)-th. The value of 40 msec. is too long for such noise and it is not enough to skip a true note. The forward and backward analysis are then composed giving respectively a first estimate of the onset and offset points. The fragments of signal between each onset and offset represent the musically meaningful events.

Before localizing voiced and unvoiced regions, we calculate the derivative of the signal normalized to the maximum value, so that the difference in amplitude is emphasized. This way, it will be easier detecting the voiced consonants since their energy is most likely to be lower than the energy of vowels. A well-known technique for performing the voice/unvoiced discrimination is derived from speech recognition studies and relies on the estimation of the RMS power and the Zero Crossing Rate [1, 18]. Plosive sounds show high values of zero crossing rate because the spectral energy is almost distributed at higher frequencies. Mean experimental

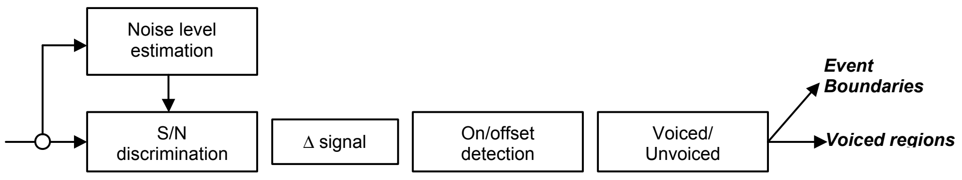


Figure 2. The pre-processing stage of the system developed. An audio signal given in input is segmented into musically meaningful events. Each event is characterized by its location in time (event boundaries) and by its voiced region.

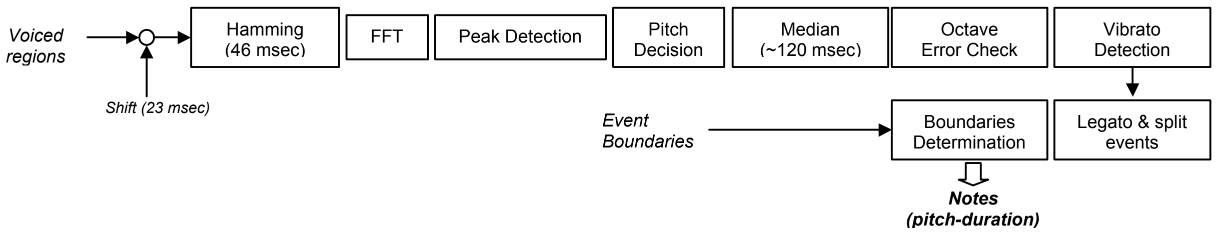


Figure 3. The proposed pitch-tracking stage; pitch detection is followed by a quantization step in which median approximation, vibrato suppression and legato detection are applied. The output is a sequence of pitches and durations.

values of average number of zero crossings are 49 for unvoiced sounds and 14 for voiced sounds in a 10 msec window. The task is not trivial for other speech utterance like weak fricatives. A better technique employs mean and standard deviation of the RMS power and zero-crossing rate of both background noise and signal as thresholds. Moreover, heuristic rules about the maximum duration admitted for each utterance are used. For example, events longer than 260 msec can not be unvoiced. These methods are applied to the derivative of the signal, detecting voiced consonants, unvoiced sounds and vowels. Thanks to this procedure, we can refine the on/offset estimation. In Figure 2 the process explained so far is illustrated.

4.2 Pitch-Tracking

As we said, the pitch of a sung note is captured by its voiced region and in particular by vowels. Thus, we will estimate pitch only on those fragments. Compared to unvoiced sounds, voiced sounds exhibit a relatively slowly-changing pitch. Thus, the frame size can be widen. For each voiced fragment identified in the segmentation step discussed above, the signal is divided into half-overlapping Hamming windows of 46 msec (2048 samples) (see Figure 3). A FFT algorithm is performed for each frame and the most prominent peaks of the estimated spectrum are passed to the next step. Here, it is taken the decision of pitch at the frame level. The algorithm is a simplified version of the one presented in [15]. The basic rule is quite simple: the candidate peak centered at a frequency in the range 87 Hz – 800 Hz that clearly shows at least two overtones is the fundamental frequency. Then, fundamental frequencies within an event are mediated along each three subsequent frames (median approximation) and are checked for octave errors. A group of four contiguous frames with similar fundamental frequencies constitutes a block. This further level of abstraction is needed to look for vibrato and legato (with glissando), which are slowly changing modulations in pitch and in amplitude. In the case of singing, vibrato is a regular modulation with rate of 4/7 Hz (i.e. with a 150/240 msec period or about 1/2 blocks) and depth between 4% and 15% [14, 20]. Legato is detected when adjacent blocks have pitches more than 0.8 semitones apart. This former case is resolved generating two different events; otherwise, the adjacent blocks are joint to form an event. For each event, pitch values are set to the average of the pitches of the constituting blocks. These information are gathered with the relative positions of consonants and the exact bounds of each note are estimated.

4.3 Post-Processing

The most critical stage is the post processing where the information captured by earlier stages are interpreted as pitch and duration. The intonation of the user is rarely absolute[3] and the transcription process has to take into account a relative musical scale. Pitches are measured in fraction of semitones to improve the importance of the relative distance between tones in the frame of the tempered musical scale. We use the definition of MIDI note; the number resulting from the following equation is rounded off to three decimal places:

$$Note_{MIDI} = \frac{1}{\log \sqrt[12]{2}} \log \frac{f}{f_0} \qquad \textbf{Eq. 1}$$

[3] Only about 1 in 10,000 people claim to have tone-Absolute Pitch [17]

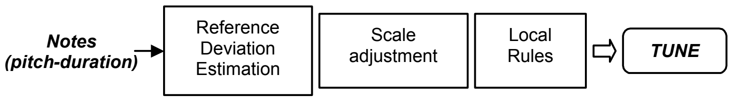


Figure 4. The post-processing stage of the system; the sequence of notes estimated in the previous stages is adjusted by means of a relative scale and four local rules. The definitive tune is given in output.

where f_0 is the frequency in Hertz associated to the MIDI note zero, that is:

$$f_0 = 440 \Big/ 2^{69/12} \cong 8.1758 Hz \qquad \textbf{Eq. 2}$$

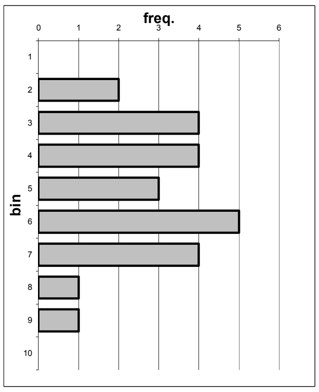
To our knowledge, only Mcnab et al. [12] introduced a procedure to adjust the scale during transcription. They used a constantly changing offset, initially estimated by the deviation of the sung tone from the nearest tone on the equal tempered scale. The resulting musical scale is continuously altering the reference tuning, in relation to the previous note. They relied on the assumption that singers tend to compress wide leaps and expand sequences of smaller intervals, suggesting that errors accumulate during singing. On the contrary, in this work we assume to deal with constant sized errors in accordance with the experiments conducted by Lindsay [9].

The tune estimated by the pitch-tracking is adjusted by means of three different steps: estimation of a reference deviation, scale adjustment and local refinement (Figure 4). The construction of a relative scale is based on the following idea: every singer has its own reference tone in mind and he/she sings each note relatively to the scale constructed on that tone. There are two important consequences: errors do not propagate during singing and are constant, apart some small increases with the size of the interval. These observations suggest to look for the reference tone of the singer through the estimation of his/her most frequent deviations from any given scale. In order to estimate the reference value for the construction of a relative scale, the semitone is divided into ten overlapping bins, each one being 0.2 semitone wide with an overlapping region of 0.1 semitone. We compute the histogram of the deviations from an absolute scale, which are the decimal digits of the estimated MIDI notes. The mean of the deviations that belong to the maximum bin is the constant average distance in semitones from the user's reference tone. Thus, the scale can be shifted by this estimated amount. An example is illustrated in Figure 5.

With the relative scale just introduced, we achieved results always better than rounding to the nearest MIDI note or implementing the algorithm by McNab et al. [12] (see next section for quantitative results). It is worth noting that the minimization of error has been obtained out of the whole performance of a singer. A further refinement is possible considering some local rules. When the reference deviation is between 0.15 and 0.85 semitones or there is more than one maximum bin in the histogram, the approximation introduced by the relative scale could be excessive. In particular, notes that have a deviation from 0.3 to 0.7 semitones on the relative scale are said to be critical. In this case, other four hypothetic melodies are considered; they reflect the following assumptions:

- a singer tends to correct its intonation from a note to the following one.
- some singers show a stable, even if slight, sharp or flat tuning with larger intervals (5 semitones and higher).
- rounded off absolute pitches and rounded intervals can further adjust an imperfect intonation

A very simple rule allows to remove single-note mistakes. The rounded melody on the relative scale is compared with the ones just calculated: a note *n* on the relative scale is replaced by the *nth* value given by three of the four representations above, when this value is the same in the three.



	Bin Range	Notes within a bin	Average Deviation
Bin 1	0.0-0.199		0
Bin 2	0.1-0.299	61.255; 58.286	0.27
Bin 3	0.2-0.399	61.255;58.286;58.328; 63.352	0.305
Bin 4	0.3-0.499	58.328;63.352;56.423;56.435	0.384
Bin 5	0.4-0.599	56.423;56.435; 61.537	0.465
Bin 6	0.5-0.699	61.537; 56.693; 56.623; 56.628; 56.644	**0.625**
Bin 7	0.6-0.799	56.693; 56.623; 56.628; 56.644	0.647
Bin 8	0.7-0.899	60.872	0.872
Bin 9	0.8-0.999	60.872	0.872
Bin 10	0.9-0.099		0

Figure 5. Example of calculation of the reference deviation (in bold style). The deviations (right) within the highest bin (left) are averaged.

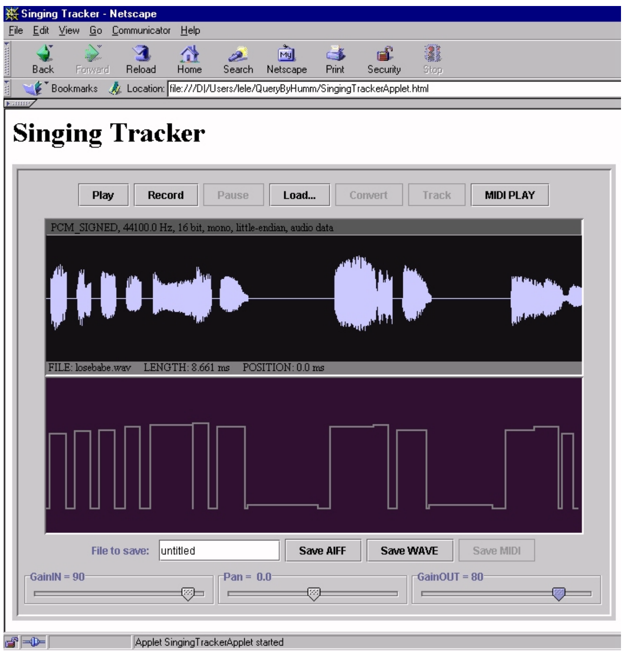
5. REPRESENTATION ISSUES

The proposed front end can translate an acoustic input into a sequence of symbols represented by MIDI note numbers and durations in absolute time (milliseconds). This representation could not be best suited for querying by melodic content because it is not invariant to transpositions and different tempi. Musical intervals are the most likely representation for searching by similarity, as it is normally implemented by current query-by-humming systems. Since we expect to have a good approximation of the absolute pitches for each note, intervals can be naturally obtained as difference between a pitch value and the previous one.

A different matter concerns the rhythmic information, for which an acceptable representation is not known. Singers are likely to make great approximations on tempo, probably larger than the errors introduced by an imperfect estimation of note boundaries. Thus, the introduction of a stage for tempo quantization should be encouraged. For example, the measured lengths in msec of each note event could be smoothed by means of a logarithmic function. We suggest to use the following definition that is invariant to different tempi:

$$ratio\,(i) = round\left(10 \cdot \log_{10}\left(\frac{duration\ (i+1)}{duration\ (i)}\right)\right) \qquad \textbf{Eq. 3}$$

The alphabet is constituted by integers in the range [–10÷10]; for instance, the value 6 corresponds to the transition from a sixteenth note to a quarter note and –6 is the reverse transition; equal length transitions are represented by the symbol 0. Since it leads to a very detailed description of rhythm, this definition can be easily relaxed for handling approximate or contour-based representation of durations.


Singing Tracker - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///D|/Users/lele/QueryByHumm/SingingTrackerApplet.html
Singing Tracker
Play Record Pause Load... Convert Track MIDI PLAY
File to save: untitled
Save AIFF Save WAVE Save MIDI
GainIN = 90
Pan = 0.0
GainOUT = 80

PCM_SIGNED, 44100.0 Hz, 16 bit, mono, little-endian, audio data

FILE: losebabe.wav LENGTH: 8.661 ms POSITION: 0.0 ms

Applet SingingTrackerApplet started

Figure 6. Screen capture of the applet Java developed running in Netscape Navigator. It is illustrated an example of the translation of melody 1 (MIDI note on vertical axis; value 0 indicates pauses; value 5 represents silence).

6. EXPERIMENTAL RESULTS

The audio front-end illustrated in section 4 has been implemented in Matlab code and Java applet. The first prototype allowed us to adjust a proper set of parameters, while the second one has been employed with human subjects. The Java applet provides a graphical interface that allows users to record their voice through the sound card and to store it as a wave file. The recorded audio can be tracked and the translation can be stored as midifile or played. The system produces two warnings in case of too low or too high recording gain. Input and output gain can be easily adjusted by means of two sliders. The audio waveform and a graphical representation of the melody are given in output on the screen (see Figure 6).

Five subjects were asked to participate to the experiment. None of them was a musician or experienced singer, although they declared to feel themselves inclined to sing. Three subjects were male and two were female. The subjects sung in the tonality they preferred but without lyrics (i.e. singing 'na-na', 'ta-ta', 'pa-pa'), simulating a query-by-humming session at home. The choice of removing lyrics was suggested to take out another possible source of errors that is difficult to quantify. Note segmentation is too much dependent on the way users remember the metric of a song. The experiment has been conducted in a non acoustically treated room, thus, in medium noisy condition. The audio card was a Creative Sound Blaster Live and the microphone was a cheap model by Technics.

Four simple melodies were chosen among the ones that the subjects proved to remember very well. The knowledge of the melodies doesn't hold a particular importance; it assures the equivalence of the tunes to be evaluated. After the recording sessions, a musician was asked to transcribe the melodies sung by all the subjects without taking care of his memory of the melodies. The aim was to keep as much as possible of the original intention of the singers, and not their ability in remembering a music fragment. A different interpretation occurred only with a subject in three ending notes of a tune, but the same amount of notes has been sung and rhythm has been preserved; for this reason, that performance was included in the test. The transcriptions constitute the reference melodies for the evaluation of the front end. Each tune has been chosen for testing a particular block of the system. The main characteristics of each melody are described in the following:

- Melody number 1: three legato notes (with bending)
- Melody number 2: three sustained notes
- Melody number 3: very long, but well-known as well, sequence of notes (28 notes)
- Melody number 4 ("Happy Birthday"): well-known tune always sung in a different key

For each tune, the output of the front end is compared with the transcriptions by the musician. Each different pitch accounts for an error. In the case of round-interval tunes, the comparison has been made on the transcribed intervals. For the evaluation of the estimated durations, we consider only segmentation errors (i.e. a note split into two or more notes and two or more notes grouped into a note).

Table 1-2. Comparison of different methods for approximating an estimated tune. With the exception of the last row, values indicate the absolute number of wrong notes.

ALL SUBJECTS	Round MIDI	Moving Tuning	Round Intervals	Proposed without local rules	Proposed with local rules
Melody 1 (13 notes)	15	38	7	5	3
Melody 2 (8 notes)	4	15	4	3	3
Melody 3 (28 notes)	38	89	34	24	21
Melody 4 (12 notes)	19	30	8	8	4
ALL MELODIES					
Subject 1	22	26	12	8	10
Subject 2	9	42	8	8	5
Subject 3	14	35	12	8	6
Subject 4	17	32	10	6	3
Subject 5	14	37	11	10	7
Average Error (%)	**24.9%**	**56.4%**	**17.4%**	**13.1%**	**10.2%**

Tests have been carried out with different approximation methods for a direct comparison of the proposed one with the following: rounded midi note, McNab's moving tuning [12] and rounded intervals. The proposed method has been also tested without local rules (see Section 4.3), in order to assess their contribution. Results are illustrated in Table 1 and 2, ordered respectively by melody and by subject and without considering segmentation errors. In Table 3 the overall error rates (with segmentation errors) are summarized.

As previously noticed, the relative scale introduced in this work is always better than any other approximation method. The moving scale developed by McNab et al. [12] has the worst performance (56.4% of wrong approximations), confirming that errors do not accumulate. Rounding to the nearest tone on an absolute scale (round-MIDI) lead to an error in 26.6% of the sung notes, showing a performance comparable to the proposed method only in the second melody. Here, the deviations from the MIDI scale are close to zero, thus indicating the simple round as a valid approximation. The round-interval tunes perform better as expected (17.4% of wrong approximated notes), since it confirms the previous work by Lindsay [9]. However, the segmentation errors have an unwanted side effect on intervals, since a single error propagates. Thus, the overall error rates increase more than the number of the segmentation errors, going from 17.4% of wrong pitches to 20.3% of wrong notes (pitch and segmentation).

Table 3. Summary of performances for the five methods employed. Error rates account for both pitch and segmentation errors

	Round MIDI	Moving Tuning	Round Intervals	Proposed without local rules	Proposed with local rules
Nr.of Wrong Notes (total number of notes=310)	81	177	63	45	36
Error Rate (%)	**26.1%**	**57.1%**	**20.3%**	**14.5%**	**11.6%**

The introduction of the four local rules bring some benefits, in fact the error rate is reduced from 13.1% to 10.2%. In absolute terms, these heuristic rules permit us to make the right approximation for ten notes more and introduce wrong approximations for only a note.

The recognition of note events has been very successful: only 5 notes were split into two events, thus identifying a total number of 310 notes instead of 305. Such a negligible error rate can be easily fixed by a somewhat fuzzy algorithm for melody comparison and retrieval, for example in a hypothetic next stage of the audio front end. As already said, in the case of round-interval the segmentation errors lead to heavier costs.

An example of translation is reported in Table 4. It is shown the

Table 4. Approximation of melody 4 (first twelve notes of "Happy Birthday"); actual notes come from the transcription by a musician. Sung melody represents the sequence of pitches given in output by the second stage of the front end.

Actual Melody	56	56	58	56	61	60	56	56	58	56	63	61
Sung Melody	56.693	56.623	58.328	56.628	61.255	60.872	56.423	56.435	58.286	56.644	63.352	61.537
Round-MIDI Melody	**57**	**57**	58	**57**	61	**61**	56	56	58	**57**	63	**62**
Dev. Round-MIDI Melody	-0.307	-0.377	0.328	-0.372	0.255	-0.128	0.423	0.435	0.286	-0.356	0.352	-0.463
Adjusted Melody	56.068	55.998	57.703	56.003	60.63	60.247	55.798	55.81	57.661	56.019	62.727	60.912
Round Adjusted Mel.	56	56	58	56	61	60	56	56	58	56	63	61
Dev. Adjusted Melody	0.068	-0.002	-0.297	0.003	-0.37	0.247	-0.202	-0.19	-0.339	0.019	-0.273	-0.088
Rounded Intervals		0	2	-2	5	**0**	-4	0	2	-2	7	-2
Moving Tuning	**57**	56	58	**57**	**62**	**61**	56	56	58	**57**	63	62

Variance Dev. Round MIDI	*0.134*
Reference Dev. Melody	*0.625*

Variance Dev. Adjusted Mel.	*0.036*

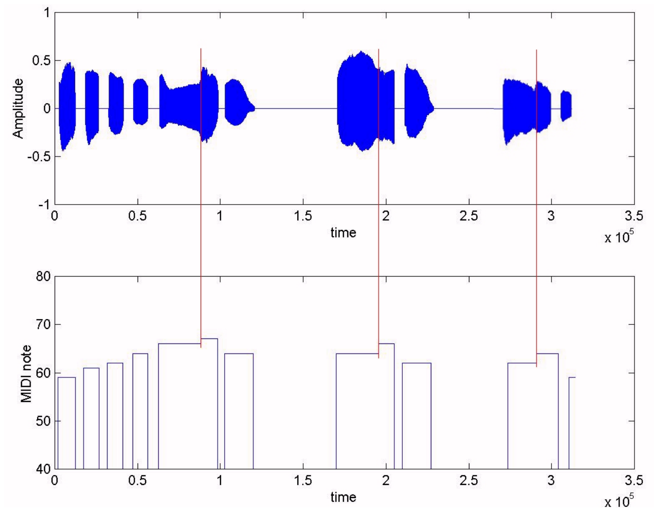


Figure 7. Example of legato notes detection (melody 1).

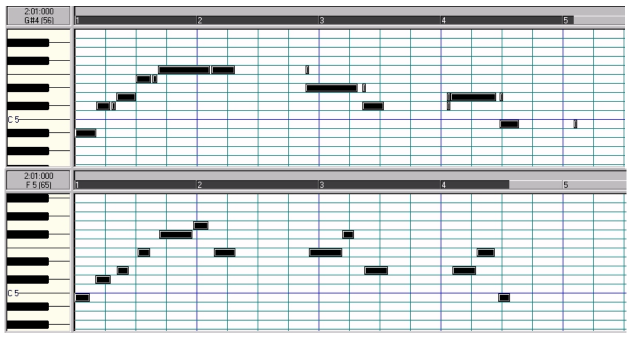


Figure 8. Transcription of melody 1 by a software tool available on the market and by the system developed here. Actual notes coincide with the sequence on the bottom.

reference deviation on which the relative scale is built; errors are indicated in bold type. Without employing any local rules, the melody is perfectly approximated on the relative scale, while the round interval and moving-tuning approximations account respectively for an error and six errors.

A hard problem for pitch-tracking algorithms are notes sung legato, for which there is neither a noticeable change in energy nor an abrupt modification in pitch. In Figure 7, the sampled waveform of the melody nr.1 is depicted with its translation. Three vertical lines highlight the estimated legato tones. The approximation introduced by the front end is able to capture the performance, splitting the legato notes in a natural way. The same file has been translated by means of Digital Ear® by Epinoisis Software [2]. Since this software tool allows smart recognition of onsets and recovery of out-of-tune notes, different settings have been employed. In Figure 8, one of the resulting MIDI files (top figure) is compared to the translation obtained with our system (bottom figure). Although it is not made clear in the figure, the actual notes coincide with the latter tune; a number of errors both in the segmentation and pitch-tracking can be noted in the former translation.

7. CONCLUSION AND FURTHER WORK

The need of dedicated singing voice processing tools strongly arises in the context of query-by-humming systems. The translation of the acoustic input into a symbolic query is crucial for the effectiveness of every music information retrieval system.

In the present work, well-known signal processing techniques have been combined with a novel approach. Our goal is the realization of an audio front end for identifying, segmenting and labeling a sung tune. The labeling stage constitutes the novelty; it enables to adjust a human performance out of a set of hypothesis on the most frequent errors made by singers. The adjustment follows two steps: global tuning and local rules. Both methods have been tested with twenty human performances (four tunes, five singers). We achieved the detection of some 90% of right notes with both steps. Previously employed methods like rounding to the nearest absolute tone or interval, and the moving tuning by McNab et al. [12], were outperformed, since they respectively accounted for about 74%, 80% and 44% of right notes. A special session of tests has been carried out to verify the ability of the pitch tracking stage in detecting vibrato and legato effects. An example has been reported in comparison with a software tool available on the market. The proposed front end roughly identified all the notes sung legato in our dataset. Quantitative results could not be presented, since it is impossible to classify as right/wrong the splitting point between two legato tones.

Much work needs to be done in different directions. First, we are developing a new pre-processing stage for the detection of noise. The aim is twofold: improving the estimation of the background noise level and filtering the noisy sources from the singing voice. This pre-process should be very robust since we are looking to applications like query-by-singing by cellular phones or other mobile devices.

In the post-processing stage, we relied on assumptions derived from the cited work of Lindsay [9]. Although these assumptions have been confirmed, a more rigorous model should be formalized. Moreover, we employ four local rules that have been introduced from experimental results but we don't know how these rules can be arranged in a more general model.

Query-by-singing is a straightforward extension of querying through hummed tones. Preliminary tests show that the task is not trivial and should need further experiments for the detection of note boundaries. As we said, language articulation could cause a wrong estimation of both the number of events and the rhythmic aspects of a performance.

Finally, current implementation suffers from the known performance deficiencies of Java. The computation time is about the same of the play time (i.e. length of the audio file) on a Pentium III, 450MHz running Windows NT 4.0. Thus, a complete re-engineering of the package is necessary and we can not exclude the possibility of migrating to other software platforms.

8. ACKNOWLEDGMENTS

The authors wish to thank Fabrizio Trotta who performed most of the preparatory and programming work for this paper. Special thanks to Giulio Agostini, Andrea D'Onofrio and Alessandro Meroni for their precious help and good advice.

This project has been partially supported by the Italian National Research Council in the frame of the Finalized Project "Cultural Heritage" (Subproject 3, Topic 3.2, Subtopic 3.2.2, Target 3.2.1).

9. REFERENCES

[1] Deller, J. R., Porakis, J. G., Hansen, J. H. L. Discrete-Time Processing of Speech Signals. Macmillan Publishing Company, New York, 1993

[2] Digital Ear®, Epinoisis Software, www.digital-ear.com

[3] Francu, C. and Nevill-Manning, C.G. Distance metrics and indexing strategies for a digital library of popular music. Proc. IEEE International Conf. on Multimedia and Expo, 2000.

[4] Ghias, A., Logan, D., Chamberlin, D., Smith, S.C. Query by humming – musical information retrieval in an audio database. in Proc. of ACM Multimedia'95, San Francisco, Ca., Nov. 1995.

[5] Haus, G. and Pollastri, E. A multimodal framework for music inputs. In Proc. of ACM Multimedia 2000, Los Angeles, CA, Nov. 2000.

[6] Kim, Y. Structured encoding of the singing voice using prior knowledge of the musical score. In Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, Oct. 1999.

[7] Kosugi, N. et al. A practical query-by-humming system for a large music database. ACM Multimedia 2000, Los Angeles, CA, Nov. 2000.

[8] Lemstrom, K. Laine, P., Perttu, S. Using relative slope in music information retrieval. In Proc. of Int. Computer Music Conference (ICMC'99), pp. 317-320, Beijing, China, Oct. 1999

[9] Lindsay, A. Using contour as a mid-level representation of melody. M.I.T. Media Lab, M.S. Thesis, 1997.

[10] Loscos, A. Cano, P. Bonada, J. de Boer, M. Serra, X. Voice Morphing System for Impersonating in Karaoke Applications. In Proc. of Int. Computer Music Conf. 2000, Berlin, Germany, 2000.

[11] Macon, M., Link, J., Oliverio, L., Clements, J., George, E. A singing voice synthesis system based on sinusoidal modeling. In Proc. ICASSP 97, Munich, Germany, Apr. 1997.

[12] McNab, R.J., Smith, L.A., Witten, C.L., Henderson, C.L., Cunningham, S.J. Towards the digital music libraries: tune retrieval from acoustic input. in Proc. of Digital Libraries Conference, 1996.

[13] Melucci, M. and Orio, N. Musical information retrieval using melodic surface. in Proc. of ACM SIGIR'99, Berkeley, August 1999.

[14] Meron, Y. and Hirose, K. Synthesis of vibrato singing. In Proc. of ICASSP 2000, Istanbul, Turkey, June 2000.

[15] Pollastri, E. Melody retrieval based on approximate string-matching and pitch-tracking methods. In Proc. of XIIth Colloquium on Musical Informatics, AIMI/University of Udine, Gorizia, Oct. 1998.

[16] Prechelt, L. and Typke, R. An interface for melody input. ACM Trans. On Computer Human Interaction, Vol.8 (forthcoming issue), 2001.

[17] Profita, J. and Bidder, T.G. Perfect pitch. American Journal of Medical Genetics, 29, 763-771, 1988.

[18] Rabiner, L.R. and Schafer, R.W. Digital signal processing of speech signals. Prentice-Hall, 1978.

[19] Rolland, P., Raskinis, G., Ganascia, J. Musical content-based retrieval: an overview of the Melodiscov approach and system. In Proc. of ACM Multimedia'99, Orlando, Fl., Nov. 1999.

[20] Rossignol, S., Depalle, P., Soumagne, J., Rodet, X., Collette, J.L. Vibrato: detection, estimation, extraction, modification. In Proc. of DAFX99, Trondheim, Norway, Dec. 1999.

[21] Sundberg, J. The science of the singing voice. Northern Illinois University Press, Dekalb, IL, 1987.

[22] Uitdenbogerd, A. and Zobel, J. Melodic matching techniques for large music databases. In Proc. of ACM Multimedia'99, Orlando, Fl., Nov. 1999.