Sing-a-Ring: a Custom Ring Tone Creation System for Mobile Platforms

ANDREW MOEDINGER, DANIEL PLANTE
Department of Mathematics and Computer Science
Stetson University
421 N. Woodland Blvd Unit 8332, DeLand, FL 32723
USA

Abstract: - Custom ring tones for cellular phones have become extremely popular over the past several years, allowing users to customize their mobile computing experience, expressing their personality or enjoying their favorite tune. In the ring tone selection process, a user generally searches or browses for a ring tone by title in one of several ring tone databases and then purchases his selection. Research into query-by-humming systems may eventually enable users to select a song by singing a segment of it. This project, *Sing-a-Ring*, extends the user ring tone experience by allowing users to create their own ring tones by singing. Unlike current options to record a sound file and set that as a ring tone directly, this application uses an autocorrelation algorithm to convert the recorded sample to a MIDI file and then provides customization options, such as instrument and tempo selectors. This application is implemented on the Java™ 2 Platform, Micro Edition (J2ME), and has been tested on a Motorola RAZR V3i phone. For the current implementation, due to format conversion issues, recordings must be created on a computer and then transferred to the phone. Nevertheless, the current application is a proof of concept with great potential, especially if access to AMR decoding libraries and the digital signal processing chip is available.

Key-Words: - Cell phone, Custom ring tones, Mobile platforms, Music transcription

1 Introduction

In the U.S. in 2005 alone, an estimated $500 million was spent on ring tone purchases, up from $245 million in 2004 [1]. These ring tones are generally selected by searching or browsing for a ring tone by title, genre, or artist and then downloaded to a phone. Research into query-by-humming (QBH) systems offers the possibility of an alternative ring tone selection method. Instead of locating a song by name or other descriptive features, with a QBH system, a user can sing or hum a melody and similar ring tones are returned in a result list. There are a variety of approaches to QBH, and a performance comparison of several popular techniques, including note-interval matching, melodic contour matching, hidden Markov-models, and the CubyHum algorithm can be found in [2].

While current ring tone selection methods have proven very popular, they limit users to the set of songs in the database and have scalability issues. In order to further the customization experience, the system implemented in this paper, *Sing-a-Ring*, allows users to create their own ring tone by singing it. In contrast to current systems that allow a user to record a sound sample and then play it directly as a ring tone, this application transcribes a recorded sample to a MIDI file and then provides the user with the ability to set the instrument, tempo, and transposition of the newly created ring tone.

There are limitations in the current implementation that are not an issue with traditional ring tone selection methods, such as the restriction that melodies can only consist of one voice, the requirement that the user hold a tune with reasonable accuracy, and the requirement that the melody fall within the singing range of the user. These are not small problems, but there are steps to alleviate them. For instance, it would be possible to allow a user to make two recordings, interleaving the two transcribed melodies into one ring tone.

The rest of this paper is organized as follows. Section 2 provides an overview of current transcription techniques. Section 3 discusses the implementation of *Sing-a-Ring* and some challenges

and solutions specific to cell phone development. Application results are presented in Section 4, Section 5 provides a brief conclusion, and Section 6 discusses ideas for future work.

2 Transcription

The conversion of raw audio data from a musical performance into a human readable score has numerous intricacies and challenges, including: identification of the different timbres of different instruments; differentiation of individual notes despite spectral overlap caused by overtones; recognition of precise note durations; and allowance for poor recording quality, imprecise performance, and stylistic performance practices. Both commercial [3], [4] and research-oriented systems exist with varying levels of accuracy. Some recent work includes matrix factorization [5], neural networks [6], Dynamical Bayesian Networks [7], hidden Markov models [8], and direct signal processing methods [9], [10], some of which incorporate various psychoacoustic models. These approaches have yielded highly successful results for monotimbral pieces, pieces that remain in one key, and synthetically generated sounds.

2.1 Onset, Offset, and Rhythm Detection

An important part of any complete transcription system is a method to determine when notes begin and end. For a system that converts a recording to a score, it is also necessary to make an estimate for the meter of the piece. Bello et al. have a summary of current onset techniques in [11]. The basic process of most onset detection algorithms is to begin with a signal, apply some number of preprocessing steps, such as amplitude normalization or half-wave rectification, apply a reduction algorithm to generate a detection function, and then choose onsets with a peak identification algorithm.

After a reduction algorithm has constructed a detection function, there is often some post processing to provide consistency to the detection function and a threshold selection process to determine what constitutes enough activity in the detection function for a note to be considered active.

Many applications also require meter estimation to generate a musical score. One approach is to simply perform long-term autocorrelation estimates, generating peaks at the strong beats of the piece, indicating the meter, or rhythmic periodicity, rather than the pitch periodicity generally obtained when applying autocorrelation [12]. There are also more involved approaches which attempt to identify all note onsets and then estimate the underlying rhythm, often with probabilistic methods [9].

2.2 Pitch detection

Most techniques for monophonic transcription are classifiable into the three categories of spectral-location, spectral-interval, and the "unitary" approach. Spectral-location approaches include autocorrelation of the time domain and explicit pattern matching of the Fourier decomposition of a signal. With autocorrelation, the highest peak of the autocorrelation function is assumed to represent the fundamental frequency. The spectral-interval group of algorithms focuses on the distance between prominent partials, allowing for a more comfortable acceptance of inharmonicity while the "unitary" approach describes a model that attempts to take into account both spectral-location and spectral-interval information as well as psychoacoustic research.

Two examples of fairly effective transcription algorithms are the YIN algorithm proposed by Cheveigne [10], which improves upon simple time-domain autocorrelation, and the Enhanced Summary Autocorrelation Function (ESACF) proposed by Tolonen [13], which is based on a two channel filterbank spectrum autocorrelation approach. The ESACF has been shown to also work with decent accuracy for polyphony of several voices. A more comprehensive examination of monophonic transcription techniques than the brief discussion here is available in [9] and [14].

3 Implementation

3.1 Algorithms

One of the most popular algorithms for transcription is the autocorrelation function (ACF), which is a time shifted cross-correlation of a signal with itself:

$$r_t(\tau) = \sum_{j=t+1}^{t+W} x_j x_{j+\tau} \quad (1)$$

In the equation, t is the time, W is the window size, x is the sample array, and τ is the lag, the amount of displacement in cross-correlation. Many improvements have been made to the simple ACF

algorithm to improve its accuracy. One such improvement weights lower lag values more heavily by reducing the window size for higher lag values, in order to reduce the possibility of an integer multiple of the lag estimating a halved fundamental pitch:

$$r_t'(\tau) = \sum_{j=t+1}^{t+W-\tau} x_j x_{j+\tau} \quad (2)$$

Initially a simple transcription algorithm was developed that performed autocorrelation on each 50 ms time window of a sample, with a simple filtering scheme for the resulting pitch estimates. While this transcription algorithm has decent performance, it is not feasible to run it on a cell phone due to the large number of pitch estimates required. Thus, a new transcription algorithm was developed that imposes a few requirements on the sound sample. The algorithm operates by requiring that each note be separated with a short silence (approximately 50 ms). Using an O(N) algorithm examining the amplitude envelope to detect note onset and offsets, the program assumes that the pitch remains constant for the entirety of the note. This allows one pitch estimate to be made for each note instead of each frame.

The custom onset algorithm maintains a sliding window sum of the peaks of the sample, at each peak adding the sample amplitude value and subtracting the peak value at the current time minus the window size. This generates the window sum:

$$windowSum = \sum_{i=t-windowSize}^{t} peak_i \,. \quad (3)$$

When the window sum is above a certain threshold, it is assumed that a note is active. By examining when the sum crosses the value set as the threshold parameter, it can be determined when notes begin and end. In order to minimize erroneous notes, a minimum note length is imposed on the detection process. The entire onset detection process requires one iteration of the sample array and is O(N). The threshold used for determining note offsets is currently fixed because even a simple form of dynamic thresholding is quite expensive. A fixed threshold for this application should be sufficient as long as the user sings at a moderate volume.

Once the note boundaries have been determined, an autocorrelation estimate is taken at approximately the middle of the note over a 20 ms window with an autocorrelation lag range of 64Hz to 800Hz. Since the J2ME Math library does not contain a native log function, the MIDI number is approximated from a table in static memory with a list of MIDI numbers to frequency values. In order to conserve space in transferring data to the phone via a .jar file, the sound sample was saved as a list of float values representing the sample data from the .wav file.

Once all of the notes have been temporally and spectrally located, the melody is converted to MIDI format using a custom MIDI writing library.

3.2 Cell Phone Platform

The *Sing-a-Ring* system was developed for the Motorola RAZR V3i using the Java™ 2 Platform, Micro Edition (J2ME), the only supported development environment for the V3i. Cell phone applications written with the J2ME are called MIDlets, as they use the Mobile Information Device Profile (MIDP), which is used in conjunction with the Connection Limited Device Configuration (CLDC). There is a cost for the flexibility afforded by J2ME, including runtime overhead incurred by the virtual machine.

3.3 Limitations and Challenges

Processor speed, memory size, and memory speed are three of the primary limiting factors for cell phone applications. The V3i took approximately 10 and 30 seconds to perform 1,000,000 additions and multiplications respectively. Since autocorrelation, in the process of cross-correlating a signal with itself, requires a large number of both additions and multiplications, each autocorrelation pitch estimate takes a noticeable amount of time, often slightly over a second. Data access speed is also a problem, especially when the data must be saved in the MIDlet's .jar file, because the data must first be read from the phone's memory and then decompressed with the phone's processor. The limited heap size is also problematic for a transcription algorithm, as raw sampled data can quickly consume considerable amounts of memory.

Java security privileges proved to be an issue as well. Certain portions of the J2ME APIs are only exposed to MIDlets which have been signed by a certificate granted by Motorola. An example is the JSR75 FileConnection API [15] which allows a MIDlet to read and write from the phone's main memory. Without a valid Motorola approved development certificate, a MIDlet cannot access the

phone's memory. Thus all data to be read by the phone must be packaged in the .jar file with the MIDlet, and no data can be saved back to the phone.

The main obstacle, however, to a fully functional program was the lack of an exposed API for decoding AMR streams. Unfortunately, AMR is the only recording format supported on the V3i, and while AMR streams can be played through the JSR135 API, there is no way to decode the stream to extract the raw sampled data, according to a Motorola representative [16]. Thus, it has not been possible to decode AMR data recorded on the phone and all recording must be done on a computer.

4 Results

4.1 Transcription Results

In the forthcoming discussion, the initial transcription algorithm introduced and the algorithm optimized for the cell phone will, creatively, be referred to as algorithms 1 and 2 respectively. Results from three example recordings illustrate the different problems with each algorithm. The three samples are excerpts from the beginnings of the well-known tunes, *Somewhere Over the Rainbow* (**Some**), *Jingle Bells* (**Jingle**), and M*ary Had a Little Lamb* (**Mary**), sung and recorded by the author. All three samples were recorded as 16-bit 8kHz samples to match the audio format on a V3i. Table 1 describes the properties of each sample.

Tables 2 and 3 show the accuracy of each algorithm with the number of correct notes, omitted notes, extra notes, and pitch errors for each sample. As shown, algorithm 1 omits many of the notes, though this is caused by a lack of onset detection, and thus repeated notes are mistakenly identified as one note of longer duration. Ignoring errors in note repetition, algorithm 1 correctly identifies all of the pitches at approximately the correct time, but it also mistakenly transcribes erroneous pitches that make it aurally difficult to identify the melody. Algorithm 2, in contrast, does not spectrally locate all of the correct pitches. Since it is only making one pitch estimate per note, it is assuming that the ACF estimate will be correct, though it sometimes is not. As expected, algorithm 2 outperforms algorithm 1 in terms of total errors for these samples as they were created with the constraints of algorithm 2 in mind. Neither transcription algorithm presented herein is a robust solution, but each is better suited for specific tasks: algorithm 1 functions better for sound samples generated from a MIDI synthesizer, and algorithm 2 is a decent initial algorithm for recorded voice samples given the computational limits of a cell phone. Algorithm 2 could be improved by taking several pitch estimates for each note at different sample frames where the note is active and averaging the estimates. This would, however, also increase the execution time substantially.

Table 1. Audio Samples

Sample	Length	Intonation	Notes
Some	7.4 sec	Good	7
Jingle	7.1 sec	Fair	11
Mary	8.6 sec	Poor	13

Table 2. Algorithm 1 Performance

Sample	Correct	Omit.	Extra	Pitch err.
Some	7	0	3	0
Jingle	6	3	2	0
Mary	8	5	9	0

Table 3. Algorithm 2 Performance

Sample	Correct	Omit.	Extra	Pitch err.
Some	7	0	0	0
Jingle	8	0	0	3
Mary	7	0	0	6

4.2 Execution Time Analysis

In addition to outperforming algorithm 1 on accuracy for the selected samples, algorithm 2 also outperforms it in terms of computational requirements, largely due to the reduced number of autocorrelation estimates required. Table 4 illustrates the average total run times for both algorithms for the three samples shown in Table 1 on a computer, and Table 5 displays the run times for algorithm 1 for the three samples of Table 1 on a Motorola RAZR V3i. When executed on a cell phone, algorithm 2 takes approximately two orders of magnitude longer than when executed on a computer, from one quarter of a second to one and a half minutes. Combined with the results of Table 4 this would seem to indicate that algorithm 1 should be able to run in approximately twice the time, or 3 minutes. However, the seven pitch estimates for the **Some** sample take about 20 seconds, 2.8 seconds per pitch estimate. Assuming an autocorrelation window of size 400 samples – 20 ms – with continuous pitch estimates, the 7.4 second **Some** sample would require 132 pitch estimates.

With 2.8 seconds per estimate, this comes out to more than 6 minutes just for pitch estimates in algorithm 1. Thus, it would appear that there are different limiting factors for the phone and the computer.

Table 4. Execution times on a computer

Sample	Algorithm 1	Algorithm 2
Some	430 ms	300 ms
Mary	500 ms	210 ms
Jingle	500 ms	300 ms

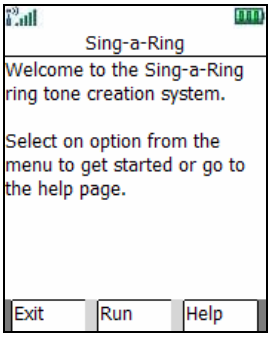
Table 5. Execution for algorithm 2 on a V3i

Sample	Init	Read	Onset	ACF	Total
Some	2.5 s	32.3 s	14.8 s	19.8 s	69.6 s
Mary	2.5 s	38.0 s	17.5 s	37.1 s	95.5 s
Jingle	2.1 s	31.1 s	14.7 s	31.5 s	79.7 s

4.3 Application Interface

The *Sing-a-Ring* application gives the user several options once the transcription process has been completed. The user may set the instrument and tempo of the ring tone and transpose it up or down. Figs. 1-4 show the user interface as it appears on the V3i emulator. At startup, the user is presented with the ability to exit, run, or display help. Once the user selects the Run option from the main menu, seen in Fig. 1, the MIDlet opens the data stream containing the sample data, reads the data into heap memory, detects note onsets, estimates the pitches of each note, and creates a MIDI stream for the transcribed melody. While the melody is being transcribed, a progress bar shows the approximate percentage that has been completed with a status message indicating the current operation, as seen in Fig. 2.

Once the transcription process is complete, the result screen displays the transcribed melody in iMelody format and shows the total time to transcribe the melody. From the result screen, the user can select to play the newly created ring tone, play the original recording, or edit properties of the new ring tone. When the edit option is selected, the edit screen, shown in Fig. 3, is displayed, and the user can interact with the tempo gauge, to increase or decrease the playback tempo of the melody, or the transposition gauge. The instrument is also selectable, with some of the options shown in Fig. 4. Changes to the settings are saved to the MIDI file once the Save command is selected, and the updated melody can be played from the result screen.



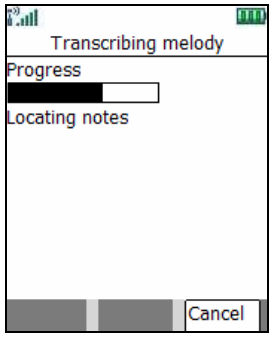
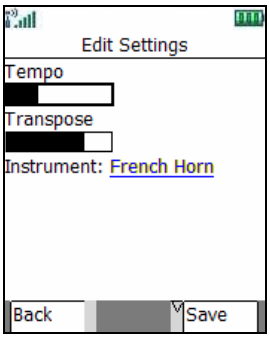


Fig. 1 Main menu; Fig. 2 Progress screen



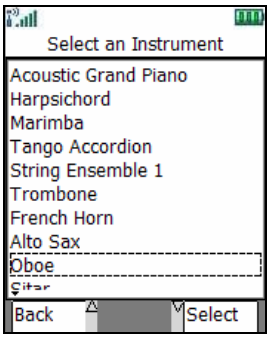


Fig. 3 Settings screen; Fig. 4 Instrument screen

5 Conclusion

Sing-a-Ring is a novel system enabling users to create ring tones. Simply by humming or singing a melody, a user can create a ring tone with any melody. This provides an alternative to the current ring tone selection methods that require users to select from a database of preexisting ring tones. In addition to providing ring tone creation capabilities, *Sing-a-Ring* allows users to modify created ring tones by changing the tempo or instrument and by transposing the entire melody to a higher or lower key. While there are limitations in the current *Sing-a-Ring* implementation, namely substantial execution time, inability to complete the entire creation process on the phone, and transcription inaccuracy, all of these problems could be alleviated, or even eliminated, with access to the digital signal processing (DSP) chip. If all of the DSP algorithms could be executed on the DSP chip, it is likely that real-time performance could be achieved, decoding from the AMR format made possible, and more advanced transcription techniques made feasible. *Sing-a-*Ring presents a strong proof of concept for a custom ring tone creation cell phone application.

6 Future Work

Options for improving the current system include a more robust autocorrelation function, improved note onset and offset detection for improved rhythmic accuracy, and melodic interval tracking to adjust for the user singing out of tune. Adjusting for out of tune singing is a vast topic unto itself with approaches from simple interval tracking to advanced probabilistic models using knowledge based systems that incorporate Western music theory. For a robust transcription solution, given the current processing capabilities of mobile computing platforms, it appears necessary that a server be used with more advanced algorithms including signal pre-processing, probabilistic modeling, band-wise processing, and use of the frequency domain. Also, where the current implementation does not support sound samples recorded on the phone, a server could decode the AMR data recorded on the phone.

In order to make a pure client model commercially viable, it would be necessary to work with Motorola or another cell phone manufacturer to gain access to the dedicated DSP chip and to AMR decoding libraries or record directly into a format with sample data available. Running as a native application would also provide a speed improvement over J2ME. Lastly, as cell phones improve in memory and processing, so will the number of features possible in a ring tone creation system.

References:

[1] Broadcast Music Inc., "BMI Forecasts U.S. Ringtones Sales to Hit $600 Million in 2006," 3 Apr. 2006 [Online], Available: http://www.bmi.com/news/entry/334746

[2] R. Dannenberg and N. Hu, "Understanding search performance in query-by-humming systems," In *Proc. of ISMIR 2004*, pp. 236-241, Barcelona, Spain, 2004.

[3] Widisoft, "WIDI Recognition System," [Online], Available: http://www.widisoft.com/

[4] Innovative Music Systems, Inc., "intelliscore Polyphonic WAV to MIDI Converter," [7 Apr. 2006] [Online], Available: http://www.intelliscore.net/

[5] S.A. Abdallah and M. D. Plumbley, "Polyphonic transcription by non-negative sparse coding of power spectra," In *Proc. of ISMIR 2004*, pp. 318-325, Barcelona, Spain, Oct. 10-14, 2004.

[6] A. Pertusa and J. Iñesta, "Polyphonic monotimbral music transcription using dynamic networks," *Pattern Recognition Letters*, Special Issue on Artificial Neural Networks in Pattern Recognition, 26.12:1809-1818, Sep. 2005, Eds. Simone Marinai and Marco Gori.

[7] A.T. Cemgil, B. Kappen, and D. Barber, "Generative model based polyphonic music transcription," In *Proc. of the 2003 IEEE Workshop on Apps. of Signal Processing to Audio and Acoust.*, Oct. 19-22, 2003, New Paltz, NJ.

[8] M. Ryynänen and A. Klapuri, "Polyphonic music transcription using event note modeling," *Apps. of Signal Processing to Audio and Acoust.*, 2005 *IEEE Workshop on*, pp 319-22, Oct. 2005.

[9] A. Klapuri, "Signal processing methods for the automatic transcription of music," Ph.D. thesis, Institute of Signal Processing, Tampere University of Technology, Tampere, Finland, March 2004.

[10] A. Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am*. 111.4, April 2002.

[11] J. Bello et al., "A tutorial on onset detection in music signals," *IEEE Trans. on Speech and Audio Processing*, 13.5.2:1035-1047, Sep. 2005.

[12] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," In *Proc. of IEEE International Conf. on Acoust., Speech, and Signal Processing*, 1999. 6:3089-92, Phoenix, AZ, Mar. 15-19, 1999.

[13] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *IEEE Trans. on Speech and Audio Processing*, 8.6:708-16, Nov. 2000.

[14] T. Viitaniemi, "Probabilistic Models for the Transcription of Single-Voice Melodies," M.Sc. thesis, Institute of Signal Processing, Tampere University of Technology, Tampere, Finland, May 2003.

[15] Sun Microsystems, "JSR-000075 PDA Optional Packages for the J2ME™ Platform," 07 Jun. 2004 [Online], Available: http://jcp.org/aboutJava/communityprocess/final/jsr075/

[16] Motorola, Inc., "Ask a Question", http://motocoder.custhelp.com