A Musical Approach to Monophonic Audio Transcription and Quantization

Jeff Hentschel
Northwestern University
j-hentschel@northwestern.edu

ABSTRACT

The transcription of monophonic audio has been attempted for many years with varying results. This paper proposes a slightly different approach, by optimizing the pitch tracker to a specific instrument, the violin. The transcriber then consolidates and quantizes the data based on different musical assumptions. The result is a midi file that can be opened into any score editor supporting midi and be printed out as sheet music.

1. INTRODUCTION

Accurate music transcription, or converting an audio file into a midi file, is a task that has been tried many different ways over the years. Past work on monophonic transcription has been done with various systems such as that with autocorrelation [1]. Polyphonic transcription less accurate, and therefore will not be discussed. One good example of this has been implemented in the SONIC system using adaptive oscillators [4].

Monophonic transcription can help in a variety of ways. Since the output is a midi file, with onset times, pitches, and intervals, it is much easier to recreate the original score. This can then be used to replay the performance. This is especially helpful in jazz music, where much of the playing is improvised. MIDI files also allow for different kinds of analysis such as chord progression and pattern recognition. With note information readily available, it is also much easier to search for a specific melody if the artist or title is not available.

2. SYSTEM DESCRIPTION

The system this paper proposes is based on the assumption that the instrument played is within the frequency range of a violin (190Hz to 3500Hz). This instrument was chosen because it is a common instrument and is often monophonic by nature.

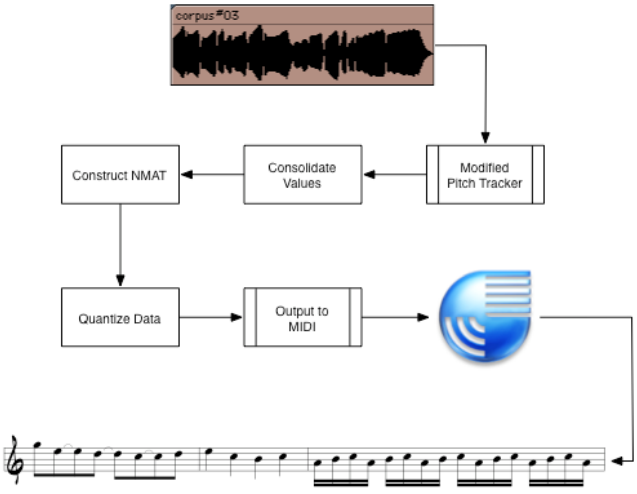

Figure 1. System Overview

The system uses the MIDI toolbox [3], and is broken into three major parts: pitch tracking, note consolidation, and quantization. Figure 1 shows the overview of the system. An audio file is sent through a pitch tracker with a user-inputted tempo. The output is then consolidated into a matrix as used by the MIDI toolbox called a *notematrix*. This creates the notes, note durations, onset times, and amplitudes of each note. The data is then quantized based on different musical commonalities, and exported to a midi file. This file can be read by many applications such as Logic, Sibelius, or Finale, and printed out as sheet music.

2.1 Pitch Tracking

The first step in audio transcription is extracting the pitch values. For this, I use a slightly modified version of Boersma's pitch tracker [2]. The tracker uses the harmonics-to-noise ratio in the autocorrelation domain. I modified the frequency range to 190Hz to 35kHz as this is the approximate range of the violin. The OctaveCost and OctaveJumpCost were set to .11 and .7. The VoicedUnvoicedCost was set to .8. These values were chosen because they gave more accurate pitches than the default values.

2.2 Note Consolidation

Since the pitch tracker only gives a list of the pitches found, it is necessary to extract individual note durations and onset times for the audio. Occasionally, the pitch tracker will output very short spurious notes. To solve this problem, the equation shown in figure 2 is used to determine the shortest note allowed.

$$\frac{60}{BPM} \cdot \frac{1}{tol - (tol \cdot tolVar)}$$

Figure 2. Note Tolerance Value

Here, the BPM is the tempo in beats per minute, *tol* is the shortest note allowed (16 is default), and *tolVar* is a percentage of tolerance (set to .45 by default).

The consolidation algorithm goes through the output of the pitch tracker. It first records an onset time. It then compares each pitch to the previous pitch. If they are the same, it adds to the note duration, and continues. If they are different, it recognizes the segment as a complete note, and, if the duration is more than the tolerance found in figure 2, adds it to the notematrix, and starts calculating the next note. To find the velocity, or amplitude of the note, it uses the maximum found.

Velocities are then rounded and scaled to 0 to 120. All rests (denoted by a midi value of 0) at the end are cut off since they do not affect the midi output.

2.3 Quantization

The final step in the transcription process before outputting to midi is quantization. This system is able to quantize onset times and durations to 4^{th}, 8^{th}, 12^{th}, and 16^{th} notes based on distances to

the closet value. The system also uses commonalities in music and performance to weight certain properties.

The first is based on onset time. Since it takes time for an instrument to create the sound, it is more common for a note to be played late, rather than early. This can create a faulty onset time if the note was played closer to an earlier time. The weight adds .1 to the distance of notes played early. This helps create a more accurate onset time.

The second cost used affects note duration. Performers will usually end a note early in preparation for the next note. To mitigate this problem, .15 is added to all distances longer than the actual duration. This helps create a more accurate note duration.

The last two costs deal with triplets (12^{th} notes). Triplets are often hard to quantize since they can easily be confused with 8^{th} and 16^{th} notes. As triplets are usually in groups of three, the stdTriplet cost is set to .5 (50%) and is used to weight triplets as such. While it is possible, it is uncommon for triplets to start with a rest. When the algorithm finds a note that would have a triplet rest before it, it adds the tripRestCost (50%) to the distance.

Two other logical quantization techniques involve pickup notes, and polyphony. If a pickupBool is set to false (default), all space at the beginning is removed. This can be very useful since there is often some silence before the song starts. If the quantizer finds two notes at the same time, it sets the second note to start immediately after the first note. This is done since the original audio was monophonic and it is impossible for there to be any polyphony.

Once all the distances are measured, the onsets and durations are set to the closest values. The resulting matrix is then exported to a midi file.

3. EXPERIMENTAL SECTION

To test the transcription algorithm, I created a corpus consisting of 24 different audio files averaging around 5 sec. in length. They were played on violin and through a synthesizer patch to get 48 audio files. They were compared to 24 matching midi files.

3.1 Corpus Construction

The real violin data was created with a Carlo Robelli electric violin thru a DigiTech RP200A pedal. The settings were as follows: There was no pickup/wah. The compressor was set to fast, an amount of 10, and a gain of 0. The CLEAN2 amp setting was used with a level of 99 and gain of 30. The equalizer was set to bass=4, mid=2, and treble=5. The cabinet was set to warm 4, with no gate. The reverb was set to club, with a decay of 39 and level of 38. The expression pedal was not used during recording, but is set to the pre-volume setting. These settings were used because they give a nice warm, clean tone with no effects. The audio was recorded in stereo 16-bit, 44.1kHz in Sound Designer II format using Logic Express 6. The files were then later converted into monophonic WAV files using QuickTime Pro.

The synthesized audio corpus is the same music as the real violin. This was done so that a comparison could be made between a live performance and a synthesized performance. The audio was recorded in Logic Express 6 using the ES2 synthesizer. The 003 STR Easy Bowing HM patch was used to create the sounds. The audio clips were then bounced to monophonic WAV format.

3.2 Evaluation method

To evaluate the performance of the transcription program, two types of distance functions from the MIDI toolbox were used. The *pcdist1* function measures the distance between the pitch classes. The *durdist1* function measures the distance between the durations of the original and transcribed MIDI files. Since both pitch and note duration are important in music, the average of the two values was used. The range of possible values is from 0 to 1. Both transcribed violin files and the transcribed synth files were compared to the original files to determine whether human performance affected the system.

3.3 Results

Overall, the transcription program seemed to work well. Table 1 shows the complete results. A graph of the results is shown in figure 3. The best results are bold italicized in red. The top 5 results are italicized in red.

Table 1. Transcription Results

	Audio		
	Violin	Synth	Diff
File 01	*0.9272*	0.9035	0.0237
File 02	0.7888	0.8709	-0.0821
File 03	***0.9589***	***0.9947***	-0.0358
File 04	*0.8824*	*0.9398*	-0.0574
File 05	0.8617	0.9170	-0.0553
File 06	*0.9309*	*0.9492*	-0.0183
File 07	0.8699	0.8926	-0.0227
File 08	0.5000	0.4006	0.0994
File 09	0.4249	0.6167	-0.1918
File 10	0.8099	0.9169	-0.1070
File 11	0.8497	0.8180	0.0317
File 12	0.6593	*0.9810*	-0.3217
File 13	0.7256	*0.9959*	-0.2703
File 14	0.3269	0.9145	-0.5876
File 15	0.6621	0.7982	-0.1361
File 16	0.4445	0.4875	-0.0430
File 17	0.7229	0.5958	0.1271
File 18	0.7567	0.6620	0.0947
File 19	0.8554	0.6136	0.2418
File 20	0.7975	0.7827	0.0148
File 21	0.8704	0.7084	0.1620
File 22	0.7732	0.6844	0.0888
File 23	0.6914	0.5515	0.1399
File 24	*0.9033*	0.8452	0.0581
Average	0.7497	0.7850	-0.0353
Median	0.7932	0.8316	-0.0205
Std. Dev.	0.1720	0.1723	0.1779

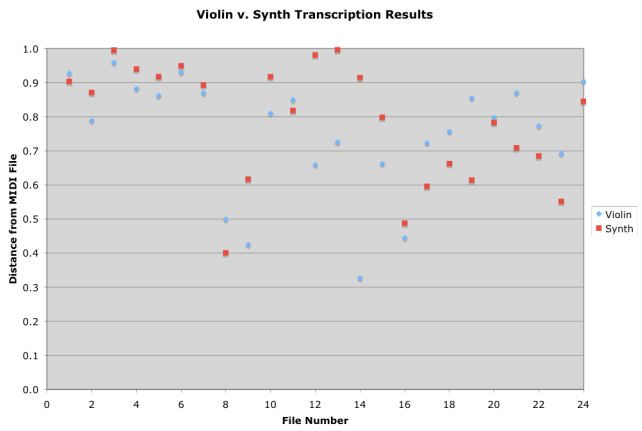

Figure 3. Transcription Results

The third corpus data file was transcribed the most accurately in both versions, with the .9589 and .9947 for the violin and synth versions respectively. The worst transcribed violin file was file 14, which only had a similarity measure of .3269. It was surprising that the synth version had a similarity of .9145. Such a drastic difference can be explained by the method used to perform the clip. The clip on the violin was played by plucking the strings, or *pizzicato*. It is possible that since this produced shorter notes, the pitch tracker had trouble getting the fundamental frequency, and the quantization step had trouble if the notes were shorter.

Another large problem had to do with octave errors and repeated note errors. File 8 had several octave jumps and repeated note errors and did badly on both violin and synth versions.

4. CONCLUSIONS AND FUTURE WORK

While the synthesized versions did slightly better than the real violin versions, human errors did not have any consistent affect on the accuracy of the results. Eighty-three percent of the transcriptions made using the system described in this paper had similarities over .6 and over 71% had a similarity of at least .7. Most of the problems seemed to be with octave jumps and repeated notes.

To further improve the system, a separate onset detector could be implemented to help solve the repeated note errors. It would also be good to look into low polyphony. For many stringed instruments such as the violin, the maximum polyphony that can be attained is four. Marolt suggested using adaptive oscillators and a collection of specially tuned neural networks to determine polyphony [4]. In this system, the user has to input the tempo for the transcription to work accurately. Later updates would have a beat-tracker that automatically found the tempo. Finally, it would be beneficial to test the system using different instruments in the same range as the violin.

5. ACKNOWLEDGMENTS

I would like to thank Prof. Bryan Pardo for his time spent and his willingness to help with the project.

6. REFERENCES

[1] Monti, Giuliano and Sandler, M. Monophonic Transcription with Autocorrelation. Dept. of Electric Engineering, King's College London. COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy. 2000

[2] Boersma, Paul. Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-noise Ratio of a Sampled Sound. Institute of Phonetic Sciences, University of Amsterdam, Proceedings 17 (1993), 97-110.

[3] Erola T. and Toiviainen, P. MIR in MATLAB: The MIDI Toolbox. Dept of Music, University of Jyväskylä, Finland. 2004.

[4] Marolt, Matija. A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music. University of Ljubljana. 2001