Benchmarking Wave-to-MIDI Transcription Tools

Henrique B. S. Leão, Germano F. Guimarães, Geber L. Ramalho and Sérgio V. Cavalcante
Centro de Informática
Universidade Federal de Pernambuco
Cx Postal 7851 CEP 50732-970 Recife-PE Brazil
{hbsl,gfg,glr,svc}@cin.ufpe.br

Abstract

This paper describes a methodology for building a benchmark to evaluate automatic music transcription tools, i.e., softwares that can extract music information from digital audio signals. There are numerous applications to these tools, like data acquisition for music virtual libraries and computer-aided music learning for instance. The main purpose of this benchmark is to present a method to ranking the current tools, as well as the characterisation of their mistakes and the respective causes. We expect that the results will be valuable both to users, guiding their choice for the most suitable tool according to their needs, and to tools' developers, showing any essential improvements required.

1 Introduction

An important issue in current computer music research is automatic music transcription, i.e., the extraction of note information such as pitch, duration and intensity from digital audio signals [10]. Such a transcription may be applied for several purposes, including data acquisition for music virtual libraries, automatic music accompaniment systems, computer-aided music learning, etc.

Motivated by our interests in music accompaniment systems [11], we started a project whose initial goal was to implement a transcription tool by improving one of the existing algorithms. However, after some research on the subject, it was noticed that there is no methodology or tool either to validate/compare the results of these algorithms, or to characterise the errors.

The solution to this lack of formal validation framework can be achieved by the use of benchmark techniques [12]. A benchmark is defined as a set of tests aiming either at measuring a system performance with respect to given tasks, or at comparing systems that use different technologies for the same task. A benchmark can also be used as monitoring and diagnosis tools, since it provides data to discover and correct common errors as well as to assess the impact of any important change implemented in the system under development.

This article describes a methodology for building a benchmark to evaluate automatic music transcription tools. The main goal of this benchmark is to provide a ranking of the current tools, as well as the characterisation of their mistakes and the respective causes. We hope that the results will be useful both to users, guiding their choice for the most appropriate tool according to their needs, and to tools' developers, showing any critical improvements needed.

Section 2 shows the conceptual project of the benchmark step-by-step. Section 3 discusses some implementation details and initial results. The results are not conclusive, since the project is just beginning. However they show that the adopted methodological approach is reasonable. In Section 4, we make some final remarks and point out directions for the project development.

2 Conceptual Project of the Benchmark

The next Sections present the phases of a methodology we are proposing for general benchmark construction in chronological order. However, these phases do not need to be accomplished obligatorily in the presented order. In fact, as in software development methodologies, an initial prototype is built and then the necessary refinements are continuously implemented following a spiral process [13]. Each phase is presented both in a general way and as it is applied to the case of the music transcription benchmark.

2.1 Analysis Subject Definition

The initial phase is dedicated to defining the benchmark subject domain, as well as the objectives to be achieved. The benchmark goals may include the classification of existent systems, the identification of pros and cons of each system, etc.

Our case study focuses on the analysis of wave-to-MIDI transcription tools (also known as WAV2MIDI), i.e., those performing the transcription of audio signals, stored in format PCM, into MIDI format [8]. The objective of this benchmark is to classify the existent softwares, to identify their pros and cons, and to relate their common mistakes to the input audio signal features. For simplicity, at this stage we are not working with polyphonic signals.

2.2 Existent Software Tools and Devices Identification

The next step of the methodology is to list the existent academic or commercial systems working on the chosen domain. This phase can be optional, mainly if the benchmark tests goal is only the improvement of a given system.

For our benchmark, there are several available software tools. Table 1 shows some of them. The "Polyphony" feature indicates that the software can transcribe polyphonic files. "Real-time" refers to software tools able to capture the sound while it is being generated and convert it continuously. In this case, each note is transcribed within a period of time that is not noticed by the user.

Software	**Real-Time**	**Polyphony**	**Availability**	**Demo Version**
Akoff Composer [1]	Yes	Yes	Commercial	Yes
Sound2Midi [14]	Yes	No	Commercial	Yes
AutoScore [3]	Yes	Yes	Commercial	No
Inst2Midi [7]	No	No	Commercial	Yes (Solaris)
WaOn [16]	No	No	Free	Yes (Linux)
Music Recognizer [9]	No	No	Free	Yes
AmazingMidi [2]	No	Yes	Commercial	Yes
Digital Ear [4]	No	No	Commercial	Yes
IntelliScore	Yes	Yes	Commercial	Yes

Table 1- Music transcription softwares

During the initial tests, we are using only three tools, that by ethical reasons they will not be disclosed, but for the final benchmark we intend to test all softwares listed above, if available.

2.3 Definition of The Test Vectors Variables

Test vectors variables are those properties used as input for the systems being studied by the benchmark. The behaviour of the system under test is analysed by modifying those variables to cover different input scenarios. In other words, the idea is to use different values and combinations of these variables and measure the impact on the system performance.

The test vector variables and their respective values define an n-dimensional space whose size depends on the benchmark study domain. The bigger the test space, the harder is the construction of the benchmark, since a great amount of vectors (scenarios) is needed to cover all possible combinations.

After scrutinising the literature and the transcription tools' manuals in order to find what could in principle influence the performance/precision of the transcription algorithms [5][6], we have chosen the test vector variables shown on Table 2.

Variable	Values
Notes frequency range	Low (C0 to F#3), Medium (G3 to C7), High (C#7 to G10)
Events (note and silence) duration	Fast (50ms to 100ms), Medium (100ms to 200ms), Long (200ms and above)
Notes intensity	Low (ppp to p), Medium (mp to mf), and High (f-fff)
Timbre	Violin, guitar and bass (representing strings), brass and saxophone (representing brass), flute and clarinet (representing woods), piano (representing keyboards), etc.
Melodic Continuity (jumps)	No (less than a minor third), Small (major third to octave), and Big (more than an octave)

Table 2 - Test vector variables and value ranges

A music transcription benchmark is particularly hard to be built due to the number of variables and the fact that variables can influence one another. For instance, low frequency notes with small duration are more difficult to be correctly transcribed than those with larger duration or higher frequencies. If one takes all possible combinations of all variables values, associated with the possible modifications of the transcription tools parameters, the test space becomes quite large.

To simplify this dimensionality problem, we grouped the values into 3 ranges (as shown in Table 2) and used a sort of "least-dependence strategy". This strategy consists of modifying only one variable value while keeping all other variables fixed at a "default" value where the algorithms generally have the best performance. In other words, we avoid using more than one variable in their critical range in order to isolate the impact of one single variable. For instance, to test the impact of frequency range, we generate a test vector whose notes belong to the three frequency ranges, have a medium duration and intensity, and presents no jumps in the melody.

2.4 Evaluation Variables Definition

The evaluation variables are those used as the basis for results comparison and analysis after the submission of the test vectors to the transcription tools. For instance, the benchmarks used for comparison of computational power of CPUs generally rely on measurements of processing speed. Hence, the evaluation variable in this case is the time the CPU under test takes to execute specific tasks.

In our benchmark the evaluation variables to be observed are:

- **Inclusion of notes –** how many new notes were introduced;
- **Exclusion of notes –** how many original notes were excluded;

- **Modification of notes** – how many original notes have any of their characteristics changed;

These are what we call basic evaluation variables (or error variables), since other errors may be derived from the observation of these ones. For more details see Section 2.6.

2.5 Test Vectors Definition

During this stage, test vectors are specially developed or derived from real problems to be used in the benchmark. The test vector format must be chosen so that it can be used as input by the systems under test. For instance, a benchmark for comparison of real-time scheduling tools may use test vectors that are composed by sets of processes specially chosen so that they encompass pre-defined characteristics that are used to stress the pros and cons of each tool.

After producing a test vector, it is necessary to validate it by making sure that it produces a feasible solution. In the scheduling tool example, it could be necessary to perform scheduling analysis to know whether the test vectors' process sets are in fact schedulable.

The test vectors for our music transcription benchmark are *wave* format files, each containing a monophonic musical phrase, which are transcribed into and recorded in *MIDI*-format files by the transcription tools under evaluation. Each phrase is used to explore a part of the test space, as it was explained in Section 2.3.

Currently, the wave-format test vectors are being generated by MIDI-to-wave conversion tools [1][2][9] that have as input MIDI-format test vectors. The actual results obtained from the wave-to-MIDI transcription tools under evaluation and the expected results stored on the original MIDI files are then compared and analysed to extract performance information, as it is explained in the next Section.

The test vectors for our benchmark could also be generated by professional musicians playing real instruments and following scores containing test-vector musical phrases. This would eliminate possible mistakes introduced by the MIDI-to-wave conversion tools but could, off course, introduce human errors. In order to choose the best way to tackle this problem, we are using analysis tools and devices, such as spectrum analysers to validate our test vectors.

2.6 Analysis Strategies Definition and Implementation

It is necessary to create tools that make it possible the acquisition and analysis of the evaluation variables defined previously. In order to encompass most variables, the data structure chosen for our benchmark was a bi-dimensional matrix of bits, where the vertical axis represents the frequencies and the horizontal axis the time. Currently we are not considering the notes' intensity analysis. If it were considered, the data structure would become a three-dimensional matrix (the third dimension would be the intensity), what would increase substantially the analysis complexity.

The test vectors contain monophonic musical melodies and so the music transcription tools under test should produce monophonic results. Thus, the initial idea was to use techniques of sequence comparison to seek for inserts and exclusions of notes and to analyse any changes introduced in the notes. However, it was observed that some transcription tools could introduce notes in any time and of variable length, generating polyphonic spaces that disable the analysis through sequence comparison techniques. Although the test vectors are not real melodies, they were produced to provide information about the transcription tools' limitations.

The analysis tool implemented so far uses as input both a MIDI-format test-vector file and the MIDI file created from the conversion of the corresponding wave-format test vector by the transcription tool under test. Both files are turned into matrices of bits. It was verified that some transcription tools create MIDI files with temporal and/or frequency shifts, what we call *general displacements*. So, the analysis tool starts by verifying the existence of any general displacement. This is accomplished by an exhaustive search for the better match between both MIDI files. To better understand the problem, compare the original MIDI file on the left side of Figure 2 with the converted MIDI file on the right.

After finding the better displacement, a comparison is performed between the matrices to calculate the amount of note inserts, exclusions, and matches. The benchmark tool provides a graphical visualisation of the results where the matches, inserts and exclusions are presented in different colours. In order to perform the analysis, it is necessary to define the time granularity, i.e., the maximum amount of time within which errors are disregarded. For example, in our case we split the entire test vector in 10ms units. This means that if a note was originally produced with 100ms and the conversion tool produced a 105ms note, no error is considered. However, if a 125ms note is produced it is considered that the conversion tool produced an error of 2 time units.

In order to characterise the transcription tool under test it is necessary to analyse the patterns formed by inserts, exclusions and matches. Depending on the test vector used, several characteristics can be studied, such as: frequency range, maximum allowable jump, minimum note time duration, accepted instruments (timbre), and so on. At the same time, it was found out that some errors are recurrent, what suggests an error taxonomy. Listed below there are some common errors we have found so far:

- **Maximum polyphony:** maximum number of simultaneous notes in a given moment. Since the input test vectors are monophonic there should be no polyphony at the output.
- **Harmonic insertion degree:** quite often extra notes are inserted with frequencies that area harmonics of the original notes (see Figure 3).
- **Note duration lengthening:** some tools either stretch or shorten the note duration in absolute or relative amounts.
- **Note frequency lengthening:** similar to the note duration lengthening but regarding frequency. In this case notes are created with frequencies situated in the vicinities of the original ones generating non-harmonic polyphony.
- **Time shift:** this is related to the general displacement discussed before but considers only the time dimension.
- **Frequency shift:** similar to the time shift but regarding frequency.

2.7 Tests Environment Definition

It is very important that the environment where the benchmark is run be the same for all tests. This environment may include details of the auxiliary software tools and hardware equipment. For instance, when testing performance of data compacting software, where execution time is an evaluation variable, the tests must be performed in the same hardware platform to avoid producing false results.

In our case, the environment definition is not very complicated. Since the time to transcribe musical files is not being considered, the hardware platform is not important. Nevertheless, the auxiliary software tools must be the same. The same MIDI-to-wave conversion tool generates all wave-format test vectors and the same software tools perform all analyses.

2.8 Benchmark Validation and Refinement

Like most software development processes [13], benchmark designing is an activity where all design stages are constantly re-visited. For example, it is possible that, after running some tests, new error types be found. If considered relevant enough, they are included in the error taxonomy.

In the music transcription benchmark, all errors included in the taxonomy mentioned in Section 2.6 were found out after running initial benchmarking tests.

3 Implementation and Results

Several tools were used for the development and execution of the music transcription benchmark, some of them are commercial ones while others were specially developed. This tool set is composed of:

- **MIDI-to-wave transcription tool**, for wave-format test vector generation. The WinGroove tool was used [17];
- **Spectrum analyser**, to validate the quality of the generated wave-format test vectors. The chosen tool was Spectrogram [15];
- **Music Bench**, an analysis tool specially developed and able to perform all analyses previously defined in Section 2.6. It extracts information about note insertion, deletion and matching, as well as, error classification. The user can define the time granularity used for analysis. It was developed using Delphi 5 tool suit.

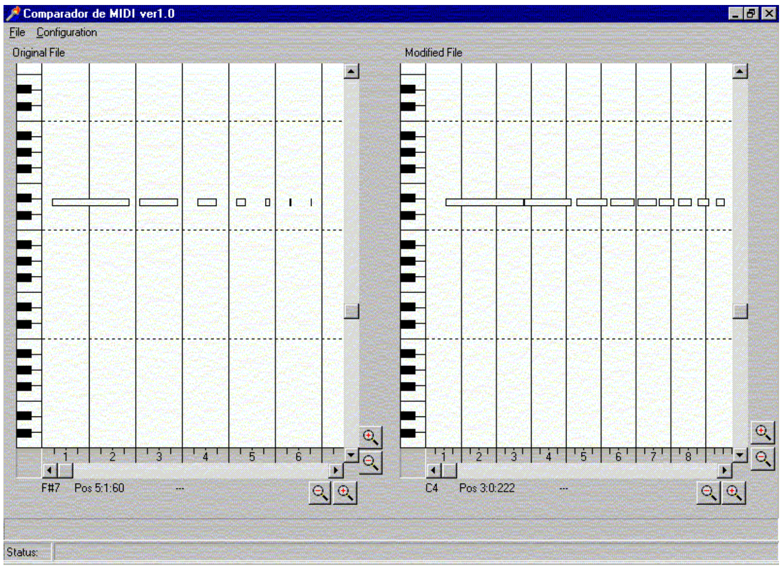

Figure 1 - Music Bench tool showing the original time test vector and the converted MIDI file generated by the tool *A*, on the left and on the right, respectively.

So far three wave-to-MIDI transcription tools have been analysed using the music transcription benchmark proposed in this article (named A, B and C). The test vectors used are tailored to produce results about timing, frequency, intensity and timbre. The results are still preliminary but sufficient to draw some conclusions.

The experiment aimed at analysing the tools behaviour with respect to notes duration changes. The test vector contained several notes with this same frequency (within the default range), but different duration, from 2 seconds to 5 milliseconds, the next note lasting half of the former.

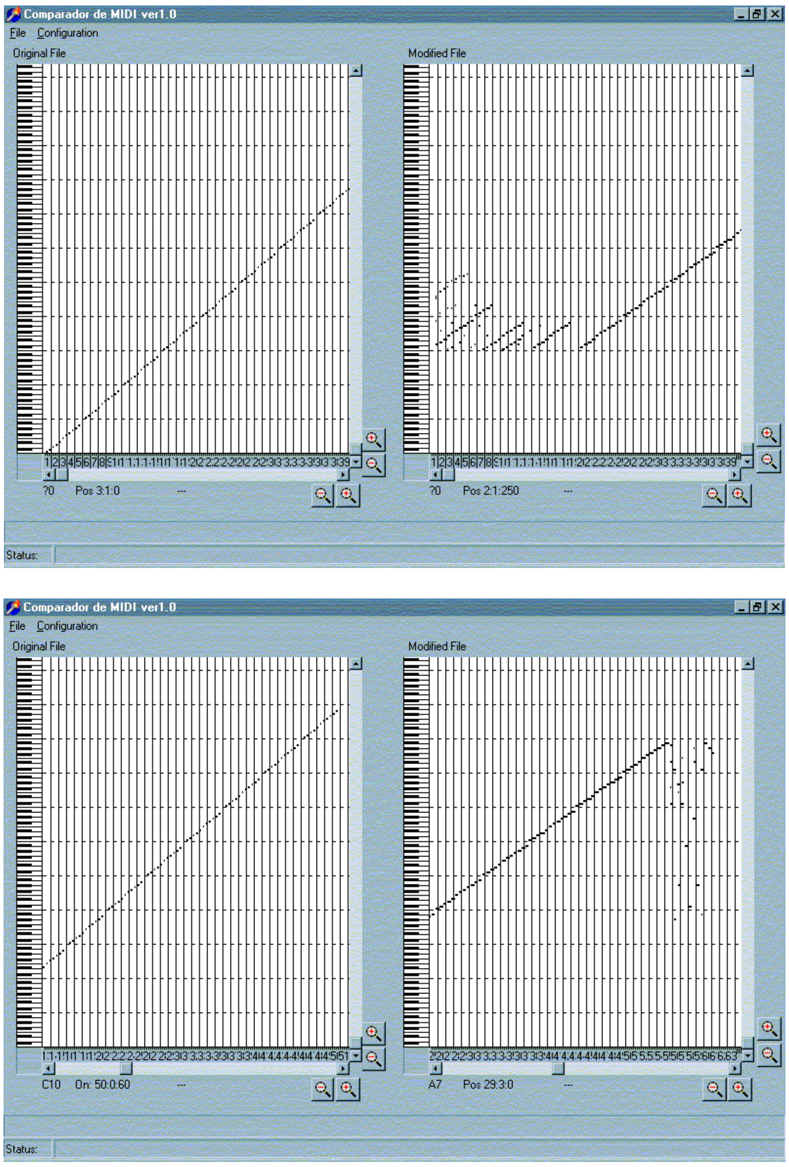

Figure 2 - The original test vector and the converted MIDI file generated by the tool *A*, on the left and on the right, respectively. The top figure illustrates the beginning of the file and the bottom one the remainder.

The results obtained for the tool *A* can be seen in the Figure 1. It shows, on the right-hand side, the MIDI file generated. It is easy to see that the notes changed their duration and start time by a small shift, i.e., the notes start and finish a bit later. The silence period between the first and second notes also almost disappears.

The test vector used to generate the results shown above was driven to produce results about the errors produced by the transcription tool related to the note frequency. The vector contains several notes in an ascending frequency order starting from the lowest possible note in MIDI-format (note 0) and ending in the highest (note 127). Each note has a period of 0.5 seconds with the same amount of time of silence between notes, as it can be seen in Figure 2.

The results obtained for each tool can be seen in Figures 2 and 3. Figure 2 shows, on the right-hand side, the results obtained with the tool *A*. It is easy to see the frequency range within which it is able to produce good transcriptions. Below frequency C3, it captures only harmonics of the real notes (shown on the left side of the figure). Above note C9 it produces what seems to be random results. In between these notes the results are very good. In fact, these are the best results among all three tools tested, although there was found a small time shift, what we considered a minor error.

The results of the tools B and C can be seen in Figure 3. It is possible to see the frequency limits accepted by each tool as well as the existence of note streams parallel to the original one that are harmonics of the original sequence. Also these tools produced note frequency lengthening (see Section 2.6), as it can be verified in the figures.

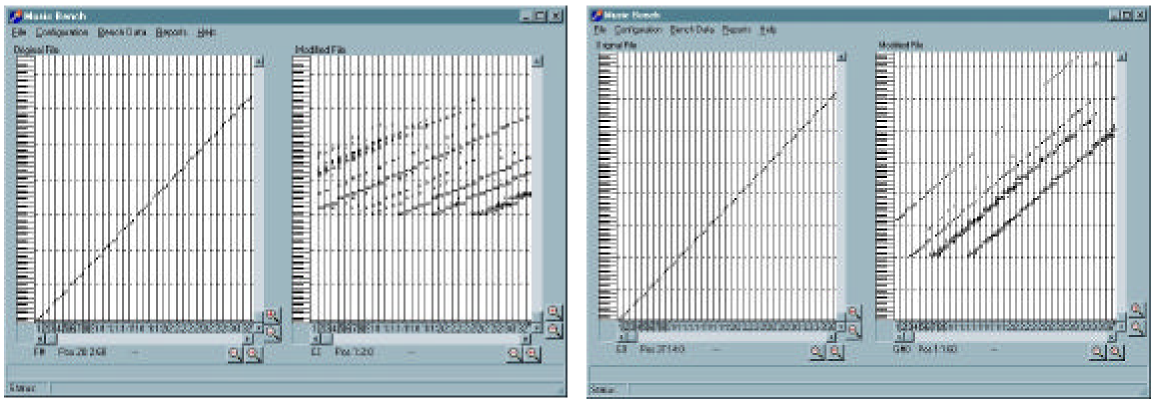

Figure 3. Midi Files generated, respectively by tools B and C (on the right-hand sides)

Table 3 presents the amount of inserts, exclusions, and matches as well as the time and frequency shift for each analysed tool using the frequency test vector. The time granularity used was 10ms.

	Tool A	**Tool B**	**Tool C**
Number of insertion points	8750	21719	29225
Number of exclusion points	2562	2008	2478
Number of matches	3507	4602	3591
Frequency shift	0	0	0
Time shift (ms)	500	0	160

Table 3 - Test results using the frequency-range test vector.

Through these data and the shown figures we arrived to the conclusion that the tool *A* is the best of the tested tools for this test vector. It produced a number of inserts much smaller than the other tools over the entire frequency range. Within the range from C3 to C9 it is far better than the other tools, not inserting or excluding any notes, and so producing a perfect match with the original musical phrase.

The results presented on Table 4 show the tools' performance over intensity variation. It was defined to test this variation in three different frequency notes: low frequency (G#1 -midi note 20), average frequency (C5 - midi note 60) and high frequency (D9 - midi note 110). In addition, these test vectors were produced using different timbres (Piano, Flute, Brass and Violin). Currently, these test vectors were only analysed in two transcription tools (A and B).

Transcription Tool: A						
Musical Instrument	**Note**	**Matches**	**Insertions**	**Exclusions**	**Freq. Shift**	**Time Shift (ms)**
Piano	Low	0	23410	3302	0	0
	Average	2631	2050	671	0	650
	High	0	866	3302	0	0
Flute	Low	0	1323	3302	0	0
	Average	2743	2049	533	0	680
	High	0	4443	3302	0	0
Brass	Low	0	11567	3302	0	0
	Average	2202	9803	1100	0	650
	High	0	4432	3302	0	0
Violin	Low	-	-	-	-	-
	Average	2710	2140	592	0	700
	High	0	4915	3302	0	0
Transcription tool: B						
Musical Instrument	**Note**	**Matches**	**Insertions**	**Exclusions**	**Freq. Shift**	**Time Shift (ms)**
Piano	Low	0	3701	3302	0	0
	Average	1034	1378	2268	0	180
	High	0	3571	3302	0	0
Flute	Low	0	0	3302	0	0
	Average	1079	2862	2197	0	130
	High	0	2265	3302	0	0
Brass	Low	0	4242	3302	0	0
	Average	0	1886	3302	0	0
	High	0	13406	3302	0	0
Violin	Low	0	2105	3302	0	0
	Average	2203	1948	1099	0	0
	High	0	3313	3302	0	0

Table 4 - Test results using the intensity and timbre test vectors.

Through the results analysis, is possible to see that both tools have difficulties in transcribing low and high notes. The tool *A* have more matches in all tests, but the time shift is bigger. It also seems to have a similar performance in all instruments used as input, while the tool B have difficulties in transcribing Brass sounds.

4 Final considerations

This paper presented the steps taken so far towards the development of a benchmark for the comparison of wave-to-MIDI transcription tools. The main goal of this benchmark is to provide a classification of the current available tools, with a characterisation of their features and errors. We think that the results will be useful both to users, guiding their choice for the most appropriate tool according to their needs, and to tools' developers, showing any critical improvements needed. Besides the benchmarking results, transcription error taxonomy is being produced as a side effect of the development process. Although this is an ongoing research the initial results presented let us think that we are in the right track.

During the development of this work, an attempt was made to define a general methodology for benchmarking design, since we have not found any work on this area. This methodology is still being defined and needs to be validated by implementing other benchmarks and also comparing the existing benchmarks with what could be obtained by applying the methodology.

In regards to the music transcription benchmark development there are still many tasks to be done such as error taxonomy improvement, and implementation of analysis tools and test vector generation that follow that taxonomy. Also, we think of using web technologies for automatic submission and benchmarking of softwares. The idea is that the tool developer can receive test vector files, use them as input to his/her tool and send back the generated files to be automatically analysed. The results would be sent back by email.

5 References

[1] Akoff Sound Labs, Inc., http://www.akoff.com

[2] AmazingMidi. Araki Software, http://www.pluto.dti.ne.jp/~araki/amazing

[3] AutoScore. Driftwood, http://www.dw.com.au/autoscore/index.htm

[4] Digital Ear. Epinoisis Software., http://digitalear.iwarp.com

[5] Goldstein, J. 1973, "An optimum processor theory for the central formation of the pitch of complex tones." *Journal of the Acoustical Society of America* 54(6): 1946-1516

[6] Hermes, D. 1992. "Pitch analysis." In M. Cooke e S. Beet, eds. *Visual Representations of Speech Signals.* New York: John Wiley and Sons

[7] Inst2Midi. Nerds.de GbR Schmitt D., Thöne S., http://www.nerds.de/nerds/english/-index.html

[8] MMA – MIDI Manufacturers Association (1996), The Complete MIDI 1.0 Detailed Specification. Los Angeles, CA

[9] Music Recognizer. Andreenko S., Kurgansky D., http://www.chat.ru/~andreenk

[10] Rabiner, L.R., Cheng, M. J., Rosengerg A. E., & McGonegal, C. A. (1976). A comparative performance study of several pitch detection algorithms. IEEE Transaction on *Acoustics, Speech and Signal Processing,* vol. ASSP-24, No. 5, (Oct 1976), pp 399-418

[11] Ramalho, G., Rolland, P.-Y., & Ganascia, J.-G. (1999). An Artificially Intelligent Jazz Performer. *Journal of New Music Research* 28(2). pp. 105-128. Swets & Zeitlinger: Amsterdam

[12] Reinhold P. Weicker, "A detailed look at some popular benchmarks", Parallel Computing, No. 17 (1991), 1153-1172

[13] Sommerville, I. Software Engineering, 4th Edition, Addison-Wesley, 1992.

[14] Sound2Midi. AudioWorks Ltd., http://www.audioworks.com

[15] Spectrogram, http://www.mnsinc.com/rshorne/gram.html

[16] WaOn. Ichiki, K., http://www.geocities.com/SiliconValley/Ridge/4180/waon.html

[17] WinGroove, http://www.cc.rim.or.jp/~hiroki/english/