Prospects for Improving OMR with Multiple Recognizers

Donald Byrd
School of Informatics and School of Music
Indiana University, Bloomington
donbyrd@indiana.edu

Megan Schindele
School of Music
Indiana University, Bloomington
mhschind@indiana.edu

Abstract

OMR (Optical Music Recognition) programs have been available for years, but they still leave much to be desired in terms of accuracy. We studied the feasibility of achieving substantially better accuracy by using the output of several programs to "triangulate" and get better results than any of the individual programs; this multiple-recognizer approach has had some success with other media but, to our knowledge, has never been tried for music. A major obstacle is that the complexity of music notation is such that evaluating OMR accuracy is difficult for any but the simplest music. Nonetheless, existing programs have serious enough limitations that the multiple-recognizer approach is promising.

Keywords: Optical Music Recognition, OMR, classifier, recognizer

1. Recognizers and Multiple Recognizers in Text and Music

This report describes research on an approach to improved OMR (Optical Music Recognition) that, to our knowledge, has never been tried with music before, though it has been in use for some time in other domains, in particular OCR (Optical Character Recognition).

The basis of an optical symbol-recognition system of any type is a *recognizer,* an algorithm that takes an image that the system suspects represents one or more symbols and decides which, if any, of the possible symbols to be recognized the image contains. The recognizer works by first *segmenting* the image into subimages, then applying a *classifier,* which decides for each subimage on a single symbol or none. The fundamental idea of a *multiple-recognizer* system is to take advantage of several pre-existing but imperfect systems by comparing their results to "triangulate" and get substantially higher accuracy than any of the individual systems. This has been done for OCR by Prime Recognition. Its creators reported a very substantial increase in accuracy (Prime Recognition, 2005); they gave no supporting evidence, but the idea of improving accuracy this way is certainly plausible. The goal of multiple-recognizer OMR (henceforth "MR" OMR) is to do the same with music, and the basic question for such a system is how to merge the results of the constituent single-recognizer (henceforth "SR" OMR) systems, i.e., how to resolve disagreements among them in the way that increases accuracy the most.

The simplest merging algorithm for a MR system is to take a "vote" on each symbol or sequence of symbols and assume that the one that gets the most votes is correct. (Under ordinary circumstances—at least with text—a unanimous vote is likely on most symbols.) This appears to be what the Prime Recognition system does, with three to six SR systems voting on a character-by-character basis. A slightly more sophisticated approach is to test in advance for the possibility that the SR systems are of varying accuracy, and, if so, to weight the votes to reflect that.

But music is so much more complex than text that such simple approaches appear doomed to failure. To clarify the point, consider an extreme example. Imagine that system A always recognizes notehead shapes and flags on notes correctly; system B always recognizes beams correctly; and system C always recognizes augmentation dots correctly. Also say that each does a poor job of identifying the symbols the others do well on, and hence a poor job of finding note durations. Even so, a MROMR system built on top of them and smart enough to know which does well on which symbols would get every duration right! System A might find a certain note—in reality, a dotted-16th note that is part of a beamed group—to have a solid notehead with no flags, beams, or augmentation dots; B, two beams connected (unreasonably) to a half-note head with two dots; C, an "x" head with two flags and one augmentation dot. Taking A's notehead shape and (absence of) flags, B's beams, and C's augmentation dots gives the correct duration. See Figure 1.

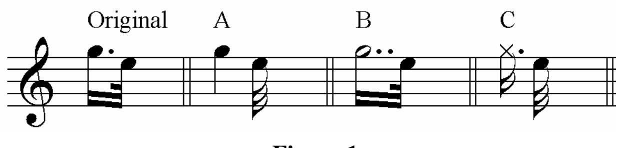


Figure 1.

For music, then, it seems clear that one should begin by studying the pre-existing systems in depth, not just measuring their overall accuracy, and looking for specific rules describing their relative strengths and weaknesses that an MROMR system can exploit.

It should also be noted that fewer high-quality SR systems exist for music, so it is important to get as much information as possible from each.

1.1 Alignment

Music as compared to text presents a difficulty of an entirely different sort. With any type of material, before you can even think of comparing the symbolic output of several systems, you must know which symbols output by each system correspond, i.e., you must *align* the systems' output. (Note that we use the verb "to align" in the computer scientist's symbol-matching sense, not the usual geometric sense.) Aligning two versions of the same text that differ only in the limited ways to be expected of OCR is straightforward. But with music, even monophonic music, the plethora of symbols and of relationships among them (articulation marks and other symbols "belong" to notes; slurs, beams, etc., group notes horizontally, and chords group them vertically) makes it much harder. And, of course, most music of interest to most potential users is not monophonic. Relatively little research has been done to date on aligning music in symbolic form; see Kilian & Hoos (2004).

2. MROMR Evaluation and Related Work

Obviously, the only way to really demonstrate the value of a MROMR system would be to implement one, test it, and obtain results showing its superiority. However, implementing any system at all, on top of conducting the necessary research to design it, was out of the question in the time available for the study described here. Equally important, the evaluation of OMR systems in general is in a primitive state. Not much progress has been made in the ten years since the groundbreaking study by Nick Carter and others (Selfridge-Field, Carter, et al, 1994). A paper by Droettboom & Fujinaga (2004) makes clear the difficulty of evaluating OMR, pointing out that "a true evaluation of an OMR system requires a high-level analysis, the automation of which is a largely unsolved problem." This is particularly true for the "black box" commercial SROMR programs, offering no access to their internal workings, against which we would probably be evaluating the MROMR. And the manual techniques available without automation are, as always, costly and error-prone. Two interesting recent attempts to make OMR evaluation more systematic are Bellini et al (2004) and Ng et al (2004). Bellini et al propose metrics based on weights assigned by experts to different types of errors. Ng et al describe methodologies for OMR evaluation in different situations.

Automation aside, it is not clear whether an evaluation should consider the number of errors or the amount of work necessary to correct them. The latter is more relevant for many purposes, but it is very dependent on the tools available, e.g., for correcting the pitches of notes resulting from a wrong clef. Also (and closely related), should "secondary errors" clearly resulting from an error earlier in the OMR process be counted or only primary ones? For example, programs sometimes fail to recognize part or all of a staff, and as a result miss all the symbols on that staff. Finally, with media like text, it is reasonable to assume that all symbols are equally important; with music, that is not even remotely the case.

Under the circumstances, all we can do is to describe the basis for an MROMR system and comment on how effective it would likely be.

2.1 Methodology

2.1.1 Programs Tested

We studied three of the leading commercial programs available as of spring 2005: PhotoScore 3.10, SmartScore 3.3 Pro, and SharpEye 2.63. All are distributed more-or-less as conventional "shrink wrap" programs, effectively "black boxes" as the term is defined above. We also considered Gamut/Gamera, one of the leading academic-research systems (MacMillan, Droettboom, & Fujinaga, 2002). But Gamut/Gamera's great flexibility—it is fully user-trainable for the "typography" of any desired corpus of music—meant that it would have taken too long to get started.

2.1.2 Test Data and Procedures

While the "largely unsolved problem" of "a true evaluation" is a serious obstacle for building a working MROMR system, determining rules for *designing* one is rather different from the standard evaluation situation. The idea here is not to say how well a system performs by some kind of absolute measures, but to say in detail how the SROMR systems compare to each other.

What are the appropriate metrics for such a detailed comparison? We considered three questions, two of them already mentioned under Evaluation. (a) Do we care more about minimizing number of errors, or about minimizing time to correct? Also (and closely related), (b) should we count "secondary errors" or only primary ones? Finally, (c) how detailed a breakdown of symbols do we want? In order to come up with a good multiple-recognizer algorithm, we need a precise description of errors. Therefore the answer to (a) is *minimizing number of errors;* to (b) is, *to the greatest extent possible, we should not count secondary errors.* For item (c), consider the "extreme example" of three programs attempting to find note durations we discussed before, where all the information needed to reconstruct the original notes is available, but distributed among the programs. So, *we want as detailed a breakdown as possible.*

We adopted a strategy of using as many approaches as possible to gathering the data. We assembled a test database of about five full pages of "artificial" examples, including the well-known "OMR Quick-Test" (Ng & Jones, 2003), and 20 pages of music from published editions. The database has versions of all pages at 300 and 600 dpi, with eight bits of grayscale; we chose these parameters based on information from Fujinaga and Riley (2002, plus personal communication from both, July 2005). In most cases, we scanned the pages ourselves; in a few, we used page image files produced directly via notation programs or sent to us. With this database, we planned to compare fully-automatic and therefore objective (though necessarily very limited) measures with semi-objective hand error counts, plus a completely subjective "feel" evaluation.

We also considered documentation for the programs and statements by expert users. We collected advice from skilled, regular users of each program as to the optimal settings of their parameters, among other things, intending to rely on their advice for our experiments. However, it took much longer than expected to locate experts on each program and to get useful advice from them; as a result, some of the settings we used probably were not ideal, at least for our repertoire.

2.1.3 Automatic Comparison and Its Challenges

The fully-automatic measures were to be implemented via approximate string matching of MusicXML files generated from the OMR programs. With this automation, we expected to be able to test a large amount of music. However, the fully-automatic part ran into a series of unexpected problems that delayed its implementation. For example, with PhotoScore, comparing the MusicXML generated to what the program displays in its built-in editor often showed serious discrepancies. SharpEye did not have this problem, but—since it has no "Print" command—we opened its MusicXML files and printed them with Finale. Finale sometimes misrepresented SharpEye's results, causing much confusion until we realized was happening.

2.1.4 Hand Error Count

The hand count of errors, in three pages of artificial examples and eight of published music, was relatively coarse-grained in terms of error types. We distinguished only seven types of errors, namely:

- Wrong pitch of note (even if due to extra or missing accidentals)
- Wrong duration of note (even if due to extra or missing augmentation dots)
- Misinterpretation (symbols for notes, notes for symbols, misspelled text, slurs beginning/ending on wrong notes, etc.)
- Missing note (*not* rest or grace note)
- Missing symbol other than notes (and accidentals and augmentation dots)
- Extra symbol (other than accidentals and augmentation dots)
- Gross misinterpretation (e.g., missing staff)

For consistency, we evolved a fairly complex set of guidelines to be used in the process of scanning and examining the music. All the hand counting was done by the creators of the guidelines and nearly all by a single person (Schindele), so we are confident our results have a reasonably high degree of consistency.

2.1.5 "Feel" Evaluation

We also did a completely subjective "feel" evaluation of a subset of the music used in the hand error count, partly as a so-called reality check on the other results, and partly in the hope that some unexpected insight would arise that way. The eight subjects were music librarians and graduate-student performers affiliated with a large university music school. We gave them pairs of pages of music—an original, and a version printed from the output of each OMR program of a 300-dpi scan—to compare. There were two originals: a monophonic page of Bach, and a page of Mozart piano music. Thus, with the three OMR versions of each original, each subject saw a total of six pairs. The Mozart page is the same one used by Sapp (2005); in fact, we used his scans of the page.

Based on results of the above tests, we also created and tested another page of artificial examples, "Questionable Symbols", intended to highlight differences between the programs: we will say more about this later.

2.1.6 Intellectual Property Rights and Publication Dates

An important consideration for any serious music-encoding project, at least for academic purposes, is the intellectual-property status of the publications to be encoded. Obviously the safest approach, short of potentially spending large amounts of time and money to clear the rights, is to use only editions that are clearly in the public domain. In general, to cover both Europe and the U.S., this means that the composer must have been dead for more than 70 years, and the edition must have been published before 1923. This restriction is somewhat problematic because music-engraving practice has changed gradually over the years, and perhaps less gradually since the advent of computer-set music, and the commercial OMR programs are undoubtedly optimized for relatively-recent editions.

For this study, we used a mixture of editions from the public-domain and non-public-domain periods, plus some very recent computer-set examples.

3. Results

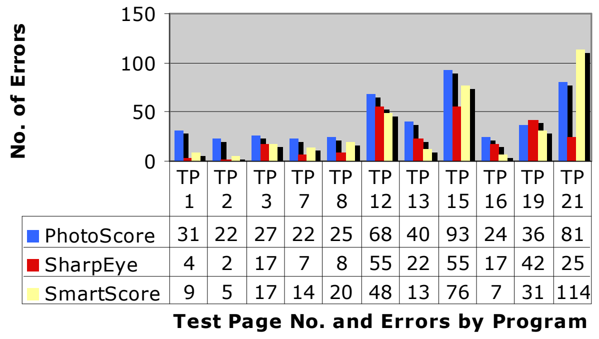
As a result of the difficulties with the fully-automatic system, we ended up relying almost entirely on the hand error counts, plus expert opinions and documentation. The following results attempt to reveal the strengths and weaknesses of each program, with an eye toward integrating that data into a future MROMR system.

3.1 Hand Error Count Results

Results of the hand error counts may be seen in Figures 2 through 5. The tables and graphs demonstrate some of the main points:

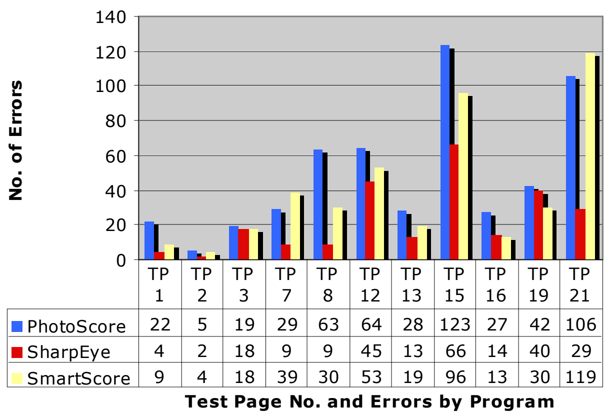
- With our test pages, the higher resolution did not really help. This was as expected, given that none of the test pages had very small staves. In fact, it had little effect on SharpEye and SmartScore. PhotoScore was much less predictable: sometimes it did better at the higher resolution, sometimes worse.
- The programs generally had more trouble with more complex and more crowded pages. This was also as expected.
- Despite its generally good accuracy and its high rating in the "feel" evaluation (see below), SharpEye made many errors on note durations. Most of these are because of its problems recognizing beams.
- By the guidelines of Byrd (2004), which seem reasonable for high-quality encodings, all three programs were well above acceptable limits in terms of note pitch and duration for multiple pages. For example, SharpEye had error rates for note duration over 1% on several of our test pages, and over 4% on one (see Figure 5). Byrd's guidelines say 0.5% is the highest acceptable rate. For note pitch, SharpEye had an error rate over 1% for one test page and nearly 1% for another (Figure 4), while Byrd's limit for pitch error is only 0.2%.

We also did a more detailed count of errors in a few specific symbols other than notes: C clefs, text, hairpin dynamics, pedal-down marks, and 1st and 2nd endings. These are included in the Total Errors graphs, figures 2 and 3.



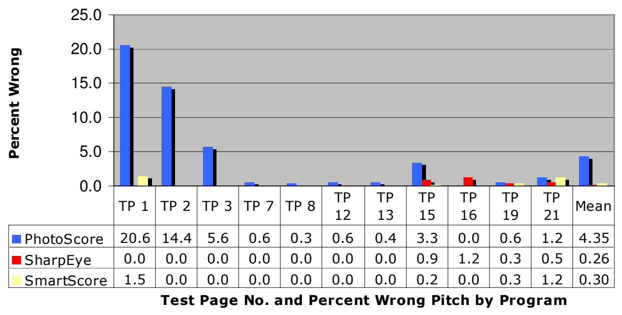
	TP 1	TP 2	TP 3	TP 7	TP 8	TP 12	TP 13	TP 15	TP 16	TP 19	TP 21
🟦 PhotoScore	31	22	27	22	25	68	40	93	24	36	81
🟥 SharpEye	4	2	17	7	8	55	22	55	17	42	25
🟨 SmartScore	9	5	17	14	20	48	13	76	7	31	114

Figure 2. Total Errors at 300 dpi



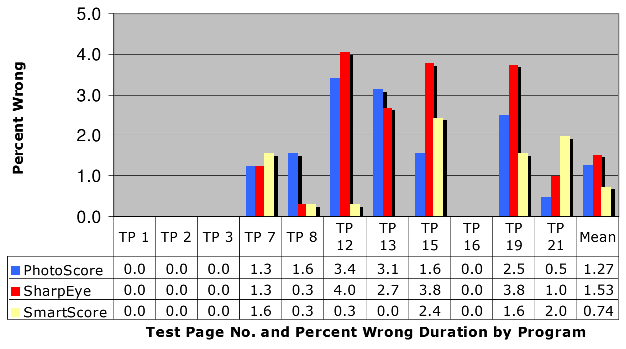
	TP 1	TP 2	TP 3	TP 7	TP 8	TP 12	TP 13	TP 15	TP 16	TP 19	TP 21
🟦 PhotoScore	22	5	19	29	63	64	28	123	27	42	106
🟥 SharpEye	4	2	18	9	9	45	13	66	14	40	29
🟨 SmartScore	9	4	18	39	30	53	19	96	13	30	119

Figure 3. Total Errors at 600 dpi



	TP 1	TP 2	TP 3	TP 7	TP 8	TP 12	TP 13	TP 15	TP 16	TP 19	TP 21	Mean
🟦 PhotoScore	20.6	14.4	5.6	0.6	0.3	0.6	0.4	3.3	0.0	0.6	1.2	4.35
🟥 SharpEye	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	1.2	0.3	0.5	0.26
🟨 SmartScore	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.3	1.2	0.30

Figure 4. Note Pitch Errors at 300 dpi



	TP 1	TP 2	TP 3	TP 7	TP 8	TP 12	TP 13	TP 15	TP 16	TP 19	TP 21	Mean
🟦 PhotoScore	0.0	0.0	0.0	1.3	1.6	3.4	3.1	1.6	0.0	2.5	0.5	1.27
🟥 SharpEye	0.0	0.0	0.0	1.3	0.3	4.0	2.7	3.8	0.0	3.8	1.0	1.53
🟨 SmartScore	0.0	0.0	0.0	1.6	0.3	0.3	0.0	2.4	0.0	1.6	2.0	0.74

Figure 5. Note Duration Errors at 300 dpi

3.2 "Feel" Evaluation Results

As a "reality check", this confirmed the hand error count results showing that all of the programs did worse on more complex music, and, in general, that SharpEye was the most accurate of the three programs. One surprise was that the subjects considered SmartScore the least accurate, but not by much: its rating and PhotoScore's were very close. The gap from SharpEye to PhotoScore was not great either, but it was considerably larger.

The rating scale of 1 to 7 was presented to the subjects as follows: 1 = very poor, 2 = poor, 3 = fair, 4 = fairly good, 5 = good, 6 = very good, and 7 = excellent.

For OMR accuracy on this scale, SharpEye averaged 3.41 (halfway from "fair" to "fairly good") across both pages; SmartScore, 2.88 (a bit below "fair"), and PhotoScore, 3.03 (barely above "fair"); see Table 2. Several of the subjects offered interesting comments, though no great insights.

Table 2. Subjective OMR Accuracy

	Test Page 8 (Bach)	*Test Page 21 (Mozart)*	*Average*
PhotoScore	3.94	2.13	3.03
SharpEye	4.44	2.38	3.41
SmartScore	3.88	1.88	2.88

3.3 Integrating Information from All Sources

We created and tested another page of artificial examples, "Questionable Symbols", intended to demonstrate significant differences among the programs. This brief example of musical notation, including fingering markings, differently shaped noteheads, tremolo markings, and several other common music notations, was created from our testing and other information sources on the programs' capabilities: their manuals, statements on the boxes they came in, and comments by experts. Of course claims in the manuals and especially on the boxes should be taken with several grains of salt, and we did that. While this data is not included in the figures in 3.1, it did provide some insight about various quirks of the three programs.

3.4 Rules for MROMR

Based on our research, we developed a set of 17 possible rules for an MROMR system. Here are four, stripped of justification:

1. In general, SharpEye is the most accurate.
2. PhotoScore often misses C clefs, at least in one-staff systems.
3. For text (excluding lyrics and standard dynamic-level abbreviations like "pp", "mf", etc.), PhotoScore is the most accurate.
4. SharpEye is the worst at recognizing beams, i.e., the most likely to miss them completely.

These rules are, of course, far too vague to be used as they stand, and they apply to only a very limited subset of situations encountered in music notation; 50 or even 100 rules would be a more reasonable number. Both the vagueness and the limited situations covered are direct results of the fact that we inferred the rules by inspection of just a few pages of music and OMR versions of those pages, and from manually-generated statistics for the OMR versions. Finding a sufficient number of truly useful rules is not likely to happen without examining a much larger amount of music—say, ten times the 15 or so pages in our collection that we gave more than a passing glance, if not more. This would be a very arduous task to perform manually.

4. Conclusions: Prospects for Building a Useful System

It seems clear that by far the best way of examining enough music to infer an adequate set of rules would be with automatic processes. This is especially true because building MROMR on top of independent SROMR systems involves a moving target. As the rules above suggest, SharpEye was more accurate than the other programs on most of the features we tested, so, with these systems, the question reduces to one of whether it is practical to improve on SharpEye's results by much. One way in it clearly *is* practical is for note durations. A more precise statement of our rule 4 (above) is "when SharpEye finds no beams but the other programs agree on a nonzero number of beams, use the other programs". This rule would cut SharpEye's note-duration errors by a factor of 3, i.e., to a mean of 0.51%, improving its performance in this respect from the worst of the three programs to by far the best. However, since the conclusion of our study, major upgrades to both PhotoScore and SmartScore have been released. Other upgrades are very likely forthcoming, and, of course, a new program might do well enough to be useful. In any case, the moving-target aspect should be considered in the design process. In particular, devoting more effort to almost any aspect of evaluation is probably worthwhile, as is a convenient knowledge representation, i.e., way of expressing whatever rules one wants to implement.

One factor we did not consider seriously enough at the beginning of this study is the fact that all of the commercial systems have built in "split-screen" editors to let users correct the programs' inevitable mistakes: these editors show the OMR results aligned with the original scanned-in image. As general-purpose music-notation editors, it is doubtful they can compete with programs like Sibelius and Finale, but for this purpose, being able to make corrections in a view that makes it easy to compare the original image and the OMR results is a huge advantage. The ideal way might be to write a new split-screen editor; that would be a major undertaking, though it could be extremely useful for other purposes, e.g., an interactive music version-comparing program. Otherwise, it could probably be done by feeding "enhanced" output in some OMR program's internal format back into it. Documentation on SharpEye's format is available, so the problem should not be too difficult.

4.1 Chances for Significant Gains and a Different Kind of MROMR

Leaving aside the vagueness and limitations of our rules, and the question of alignment before they could be applied, more work will be needed to make it clear how much rules like these could actually help. But it is clear that help *is* needed, even with notes: as we have said, in our tests, all of the programs repeatedly went well beyond reasonable limits for error rates for note pitch and note duration.

This suggests using a very different MROMR method, despite the fact that it could help only with note pitches and durations: we refer to using MIDI files as a source of information. This is particularly appealing because it should be relatively easy to implement a system that uses a MIDI file to correct output from an OMR program, and because there are already collections like the Classical Archives (www.classicalarchives.com), with its tens of thousands of files potentially available at very low cost. However, the word "potentially" is a big one; these files are available for "personal use" for almost nothing, but the price for our application might be very different. Besides, many MIDI files are really records of performances, not scores, so the number that would be useful for this purpose is smaller—probably much smaller—than might appear.

4.2 Advancing the State of the Art

The results described above are encouraging enough that we plan to continue working along the same lines towards an implementation of a MROMR system in the near future. To help advance the state of the art of OMR, we also plan to make our test collection (the files of scanning images, groundtruth files, and OMR-program output files) and findings available to the community in an informal way.

For those interested in learning more about our work, an extended version of this report is available, together with several supporting documents (exact results of the hand error counts, "Questionable Symbols", the full list of possible rules, etc.) on the World Wide Web at http://www.informatics.indiana.edu/donbyrd/MROMRPap.

5. Acknowledgements

It is a pleasure to acknowledge the assistance of Michael Droettboom and Ichiro Fujinaga (information on OMR technology, especially Gamut/Gamera); Bill Clemmons, Kelly Demoline, Andy Glick, and Craig Sapp (advice as experienced users of the programs we tested); Graham Jones (SharpEye); Don Anthony, Andy Glick, Craig Sapp, and Eleanor Selfridge-Field (obtaining test data); Tim Crawford and Geraint Wiggins (project planning, among other things); Anastasiya Chagrova (the automatic-comparison program, etc.); and, Don Anthony, Mary Wallace Davidson, Phil Ponella, Jenn Riley, and Ryan Scherle (advice and/or support of various kinds).

This project was made possible by the support of the Andrew W. Mellon Foundation; we are particularly grateful to Suzanne Lodato and Don Waters of the Foundation for their interest in our work.

References

[1] Bellini, P., Bruno, I, and Nesi, P. (2004) Assessing Optical Music Recognition Tools. Available at http://www.interactiveMUSICNETWORK.org/wg_imaging/upload/assessingopticalmusicrecognition_v1.0.doc

[2] Byrd, Donald (2004). Variations2 Guidelines For Encoded Score Quality. Available at http://variations2.indiana.edu/system_design.html

[3] Byrd, Donald, and Schindele, Megan (2006). MROMR (Multiple-Recognizer Optical Music Recognition) Web page. http://mypage.iu.edu/~donbyrd/MROMRPaper

[4] Droettboom, Michael, & Fujinaga, Ichiro (2004). Micro-level groundtruthing environment for OMR. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004),* Barcelona, Spain, pp. 497–500.

[5] Fujinaga, Ichiro, & Riley, Jenn (2002). Digital Image Capture of Musical Scores. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR 2002),* pp. 261–262.

[6] Kilian, Jürgen, & Hoos, Holger (2004). MusicBLAST—Gapped Sequence Alignment for MIR. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004),* pp. 38–41.

[7] MacMillan, Karl, Droettboom, Michael, & Fujinaga, Ichiro (2002). Gamera: Optical music recognition in a new shell. In *Proceedings of the International Computer Music Conference,* pp. 482–485.

[8] Ng, Kia C., & Jones, A. (2003). A Quick-Test for Optical Music Recognition Systems. 2nd MUSICNETWORK Open Workshop, Workshop on Optical Music Recognition System, Leeds, September 2003.

[9] Ng, Kia, et al. (2004) CIMS: Coding Images of Music Sheets. Interactive MusicNetwork working paper. Available at www.interactivemusicnetwork.org/documenti/view_document.php?file_id=1194

[10] Prime Recognition (2005). http://www.primerecognition.com

[11] Sapp, Craig (2005). SharpEye Example: Mozart Piano Sonata No. 13 in B-flat major, K 333. http://craig.sapp.org/omr/sharpeye/mozson13/

[12] Selfridge-Field, Eleanor, Carter, Nicholas, et al. (1994). Optical Recognition: A Survey of Current Work; An Interactive System; Recognition Problems; The Issue of Practicality. In Hewlett, W., & Selfridge-Field, E. (Eds.), *(Computing in Musicology,* vol. 9, pp. 107–166. Menlo Park, California: Center for Computer-Assisted Research in the Humanities (CCARH).