HUMAN-INTERACTIVE OPTICAL MUSIC RECOGNITION

Liang Chen
Indiana University Bloomington
chen348@indiana.edu

Erik Stolterman
Indiana University Bloomington
estolter@indiana.edu

Christopher Raphael
Indiana University Bloomington
craphael@indiana.edu

ABSTRACT

We propose a human-driven Optical Music Recognition (OMR) system that creates symbolic music data from common Western notation scores. Despite decades of development, OMR still remains largely unsolved as state-of-the-art automatic systems are unable to give reliable and useful results on a wide range of documents. For this reason our system, *Ceres*, combines human input and machine recognition to efficiently generate high-quality symbolic data. We propose a scheme for human-in-the-loop recognition allowing the user to constrain the recognition in two ways. The human actions allow the user to impose either a pixel labeling or model constraint, while the system re-recognizes subject to these constraints. We present evaluation based on different users' log data using both *Ceres* and *Sibelius* software to produce the same music documents. We conclude that our system shows promise for transcribing complicated music scores with high accuracy.

1. INTRODUCTION

Optical Music Recognition (OMR), the musical cousin of Optical Character Recognition (OCR), seeks to convert score images into symbolic music representations. Success in this endeavor would usher music into the 21st century alongside text, paving the way for large scale symbolic music libraries, digital music stands, computational musicology, and many other important applications.

Research in Optical Music Recognition (OMR) dates back to the 1960s with efforts by a large array of researchers on many aspects of the problem [3, 5, 10–14, 17, 18, 20, 21, 25, 29] including several overviews [6, 23], as well as well-established commercial efforts [2] [1]. While evaluation of OMR is a challenging task in its own right [8], it seems fair to say that the problem remains largely unsolved, in spite of this long history. The reason is simply that OMR is very hard, representing a grand challenge of document recognition.

One reason OMR is so difficult stems from the heavy tail of symbols used in music notation. While a small core of symbols account for the overwhelming majority of ink on the printed page, these are complemented by a long list of familiar symbols that may be absent in many or most pages. These include repeat signs, D.S. and D.C. directives, a wide variety of possible ornaments and articulations, harmonics, fingerings, pedaling, arpeggiation, fermati, double sharps and flats, pedaling, 1st and 2nd endings, repeats, etc. While each of these symbols can be recognized with reasonable accuracy by fairly standard means, the symbols are rare enough that the unavoidable false positives they produce often outweigh the value of the correct detections we may find. This constitutes one of the the essential paradoxes of OMR: we cannot simply ignore unusual symbols, though their inclusion often leads to worse performance overall.

We sometimes refer to the heavy tail described above as the "sprawl of OMR." This sprawl is not limited to the range of symbols, but also includes the many exceptions to familiar notational rules. For instance, in standard notation beamed groups, notes and chords carry the majority of musical content, thus their recognition must be central to any OMR effort. The construction of these symbols is highly rule-bound, arguing strongly for recognition approaches that exploit the symbols' grammatical nature. The difficulty here comes from the many special cases we must account for. For example, note heads usually lie on one side of the stem, though chords with note heads on adjacent staff positions are usually rendered with "wrong side" note heads; beamed groups usually have closed note heads though measured alternations between two pitches is often abbreviated with two open note heads in a beamed group; beam groups usually have stems that go in a single direction from the beam, though one occasionally sees both directions from a single beamed group; beamed groups are usually associated with a single staff, but can span both staves of a grand staff when necessary; augmentation dots are generally placed to the right of the note heads they belong to, though dense voicing can force them to wander far off their nominal positions. As with the heavy tail of symbols, these special cases can all be modeled and recognized, though the results are not what one would hope for. The majority of notation will *not* contain these rarer cases; allowing for these exceptions when they do not occur begs for trouble, reliably degrading the recognized results. However, these exceptions are common enough that we doubt a useful OMR system can be developed without accounting for them somehow. This is essentially the same paradox as that posed by the heavy tail: we must recognize these unusual cases but cannot allow them to result in degraded performance overall. Dealing with this sprawl is a

central issue we address in this paper.

In light of these (and other) obstacles we doubt that any fully automatic approach to OMR will ever deal effectively with the wide range of situations encountered in real life recognition scenarios. For this reason we formulate the challenge in terms of *human-in-the-loop computing*, developing a *mixed-initiative system* [15, 19, 27, 28] fusing both human and machine abilities. The inclusion of the human adds a new dimension to the OMR challenge, opening a vast expanse of unexplored potential.

However, there is another reason for the human-computer formulation we favor. While there may be some uses for moderate quality symbolic music data, we believe the most interesting applications require a level of accuracy near that of published scores. The human will not only play the important role of guiding the system toward the required level of accuracy, but must also "bless" the results when they are complete. We expect symbolic music data lacking this human imprimatur will be of dubious value.

Commercial systems often deal with this issue by pipelining their results into a score-writing program, thus allowing the user to fix the many recognition problems. This approach creates an artificial separation into the two phases of recognition and correction. Rather, since we require a human to sign-off on the end result, we propose to involve her in the heart of the process as well.

In what follows we present the view of human-in-the-loop OMR taken in our *Ceres* system. Our essential idea is to allow the user two axes of control over the recognition engine. In one axis the user chooses the *model* that can be used for a given recognition task, specifying both the exceptions to the "rules" discussed above as well as the relevant variety of symbols to be used. In the other, the user labels misrecognized pixels with the correct primitive type, allowing the system to re-recognize subject to user-imposed constraints. This provides a simple interface in which the user can provide a wealth of useful knowledge without needing to understand the inner-workings and representations of the system. Thus we effectively address the *communication* issue of mixed-initiative literature. Our work has commonalities with various other human-in-the-loop efforts such as [26, 30], though most notably with other approaches that employ constrained recognition as driven by a user [4, 7, 24].

2. HUMAN-INTERACTIVE SYSTEM

The main part of our *Ceres* OMR system deals with symbol recognition, which occurs after the basic structure of the page has been identified. For each type of symbol or group of symbols we use a different *graphical model*. Here we depict only the isolated chord graph in Fig. 1 which serves as a template for the others, and refer the more detailed discussions to our previous papers [9, 22].

Each individual music symbol (beamed group, chord, clef-key-signature, slur, etc.) is grammatically constrained based on a generative graph, enabling automatic, model-driven, symbolic recognition. Perhaps more difficult than

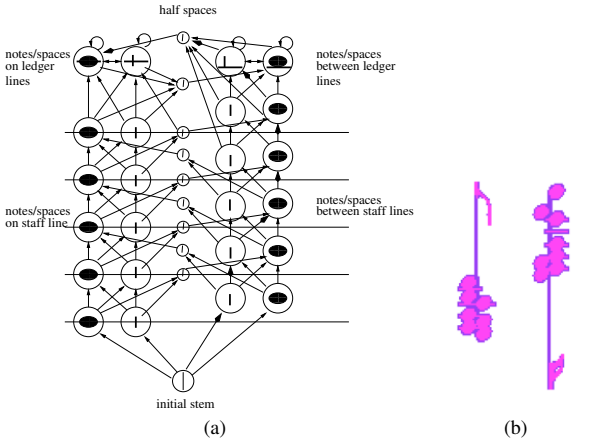


Figure 1: (a) Graphical model for isolated chord. (b) Symbol samples generated via random walk over the graphical model shown in (a).

individual symbol recognition is the challenge of decomposing the image into disjoint, grammatically-consistent symbols. To address this problem, our system identifies *candidates* for the different symbol types such as chords, beamed groups, dynamics, slurs, etc. The user chooses some of these candidates for interactive recognition. After each recognition task is completed the resulting symbol will be saved, thus constituting a constraint for future candidate detection and recognition — we constrain our system to identify mostly non-overlapping symbols.

Our current human-driven system performs recognition in a symbol-by-symbol fashion as opposed to the measure-based version proposed in [9]. The symbol-based scheme allows for a responsive and efficient interface, which functions with the symbol recognizers implemented in our system . Here the human is allowed to interact with all three steps: candidate identification, interactive recognition, and post-processing. In the candidate identification and post-processing steps, the user directs the decision-making process by either selecting a system-proposed candidate, adding a missing candidate, or deleting an incorrectly saved symbol. In the interactive recognition step, the user actions impose extra labeling or model constraints to the recognizer, while the system automatically invokes re-recognition *subject to these constraints*.

The interactive recognition begins by performing fully automatic recognition of the selected candidate. In many cases the result will be correct and will be saved. Otherwise the process iterates between human action and machine re-recognition until it yields the correct result. In each iteration of this process the user will click on a particular pixel to input the corresponding *primitive* label (solid note head, stem, beam, etc.) or change the model settings to an appropriate choice. The whole process requires only basic knowledge of music notation and thus extends the range of potential users.

Our recognizers seek hypotheses that give the highest probability to the pixel intensities, $g(x)$, where x is a particular location. More explicitly, we regard a hypothesis, h,

as a labeling of pixel sites, x, with models, $M_h(x)$, where $M_h(x) \in \mathcal{L} = \{b, w, t, 0\}$. The labels in $\mathcal{L}$ correspond to a probability models for black, white, transitional, and background pixels (P_b, P_w, P_t, P_0). We measure the quality of a hypothesis, h, by

$$S(h) = \sum_{x \in D(h)} \log \frac{P_{M_h(x)}(g(x))}{P_0(g(x))} \quad (1)$$

where $D(h)$ is the support of the hypothesis.

As mentioned, the hypotheses are highly constrained and we express these constraints through graphs as in Figure 1. We denote the possibility that a graph G generates a hypothesis h by $G \Rightarrow h$. Thus our recognizers seek to compute

$$H^*_G = \underset{\{h|G \Rightarrow h\}}{\arg\max}\, S(h) \quad (2)$$

By normalizing by the background model, P_0, in Eqn. 1, the result $S(h) = 0$ means that h explains the pixels no better or worse than the background model (iid sample from the overall gray-level distribution), thus calibrating the interpretation of our scoring function and providing a natural threshold for detection. This data model is more explicitly explained in our previous works [9, 22].

2.1 Label Constraints

When the user labels a pixel, x, with a primitive (the most fundamental unit that constitutes a symbol), l, such as stem, flag, open note head, etc., we create a constraint of the form (x, l). If several such user labelings are required to correctly recognize a symbol we have a collection of constraints of the form $L = \{(x_i, l_i) : i = 1, \ldots, n\}$. Each time we get a new constraint the system re-recognizes the current candidate *subject* to these user-imposed constraints. Thus we modify our original scoring function to be

$$S(h) = \sum_{x \in D(h)} \log \frac{P_{M_h(x)}(g(x))}{P_0(g(x))} + t(x, Q_h(x)) \quad (3)$$

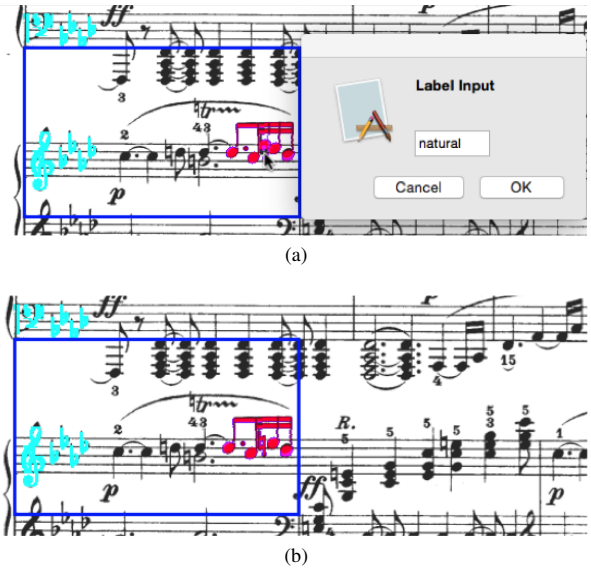
where

$$t(x, Q_h(x)) = \begin{cases} -\infty & x = x_i, Q_h(x) \neq l_i \text{ some } i \\ 0 & \text{otherwise} \end{cases}$$

Here $Q_h(x)$ represents the primitive label of hypothesis h at location x, thus $t(x, Q_h(x))$ disallows h if the pixel label is inconsistent with the hypothesis.

Fig. 2 illustrates a use case for a label constraint. In this example the original recognition, shown in the top panel of the figure, misidentifies the natural sign modifying the 'e' as an additional note head. In the top panel the user clicks on this pixel, labeling it with the "natural" primitive type. The bottom panel shows that re-recognition subject to the constraint fixes the problem.

In addition to primitive labels, the system also allows the user to label a rectangle of pixels as "white space", thus disallowing the recognizer from covering these pixels with any primitive symbol. In practice this turns out to be one of the most powerful constraints as it addresses the common case in which our recognition "spills over" into adjacent symbols.



(a)

(b)

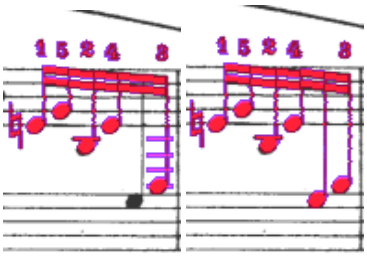
Figure 2: (a) During the initial recognition the natural was misidentified as a note head. The user is adds the correct primitive label to any location within the the natural. (b) Re-recognition correctly identifies the whole symbol by using the user-imposed label constraint.

2.2 Model Constraint

Model constraints change the graph, G, over which we optimize. Our interface contains a number of toggle switches each enabling a special case of recognition thus enlarging the graph. In general, the recognition works best when the minimal possible graph is chosen, though in many cases an overly permissive graph structure still produces the desired result without help from the user. In this case we still optimize Eqn. 2, though with a new graph, G', playing the role of G.

Fig. 3 gives an example in which an inappropriately restrictive graph generates a result with many primitive errors (top panel). In this case the original recognition was done without allowing note and chords to span the grand staff, as they do in this example. The result completely misses the penultimate note in the beamed group, while recognizing the last note with extraneous ledger lines which would be syntactically necessary when the note belongs to the upper staff. After enabling the grand staff ability we get the correct result in the bottom panel of the figure.

After the interactive recognition of a symbol is complete the user can save the symbol. When this occurs, we reset the list of pixel constraints placed by the user — these do not carry forward to future recognition tasks. The pixels involved in the recognized hypothesis are considered *unavailable* in subsequent symbol recognition, thus express-



(a) (b)

Figure 3: (a) Recognition using a beamed group graph that does not allow notes to span the grand staff. (b) Recognition using grand-staff beamed group graph.

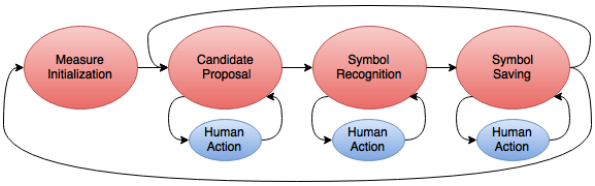


Figure 4: Human-driven (blue) and Machine-driven (red) actions in *Ceres* system. The possible state transitions are shown by arrowed connections.

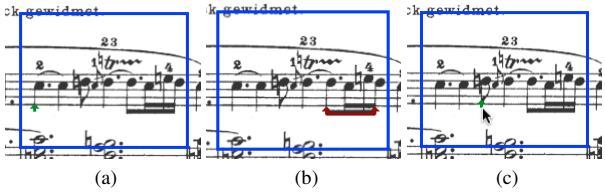
ing the notion that the symbols cannot touch. Of course, symbols do sometimes touch in practice, so we allow the user to label a rectangle of pixels as "reuse," thus allowing the user to override the basic non-overlapping constraint when needed.

3. USER INTERFACE

Our interface allows the user to control and direct the recognition process. The overall process is organized in terms of measures, while the interaction flow within a measure is described in the "action graph" of Fig. 4.

In the *candidate proposal* step (2nd level of the figure), the user can switch between the different candidate types (beamed group, isolated chord, slur, dynamics, etc.). Within each candidate type the user is presented with a left-to-right sequence of candidates she may choose to recognize or skip over. The color of the highlighted candidate reflects the candidate's type, also showing the direction of the stem, beam, and slurs with arrow signs, as this information is needed by the recognizers. The interface for this phase is shown in Fig. 5.

After a candidate is chosen, the system moves to the *symbol recognition* step. In this step, the system collaborates with user to improve the recognition in an iterative process. In each iteration the user either accepts the current recognition or imposes a new constraint (label or model), as discussed in Section 2. For a labeling constraint, the user inputs the pixel labels through a message box after clicking on the desired pixel position, as in Fig. 2a. For a model constraint, the user changes the current settings on the checkboxes or pull-down menu. The interface is shown in Fig. 6.



(a) (b) (c)

Figure 5: (a) The system detects and indicates a stem-up chord candidate for the user; (b) The system detects and indicates a stem-up beamed group candidates for the user; (c) The user adds a missing chord candidate.

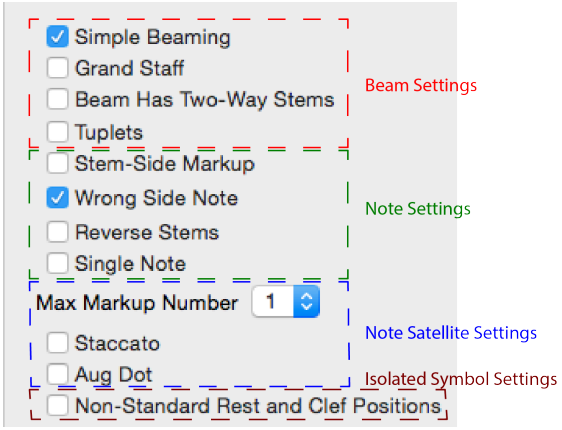


Figure 6: Checkboxes and pull-down menu for different model settings.

The system uses different colors to distinguish the current symbol from saved symbols. When the user wants to revisit an incorrectly saved symbol, she can select and delete the symbol before redoing the recognition.

Ceres has shortcut keys designed for the user to move from one step to another one, and also has a "cancel" key that allows the user to exit this process while moving to a "default" state. These interface units together constitute the visible part of *Ceres'* human-in-the-loop system.

4. EVALUATION

We evaluate our system both in terms of accuracy and speed. Both criteria are important since we believe the most interesting applications of OMR require accuracy on par with published scores, while it won't be possible to create large quantities of such data through OMR unless the process is highly efficient.

We measure accuracy here at the primitive level (beams, flags, note heads, accidentals, etc.), rather than, say, in terms of pitch and rhythm as in [16]. We prefer primitive evaluation because it is generic (all symbols are evaluated in the same way), it allows for all symbols our recognizer treats — not just those carrying pitch and rhythm information, and it is easy to relate primitive error analysis to specific aspects of the recognition engine.

The test set consists of first three pages of the *Breitkopf and Härtel* 1862-90 edition of Beethoven's Piano

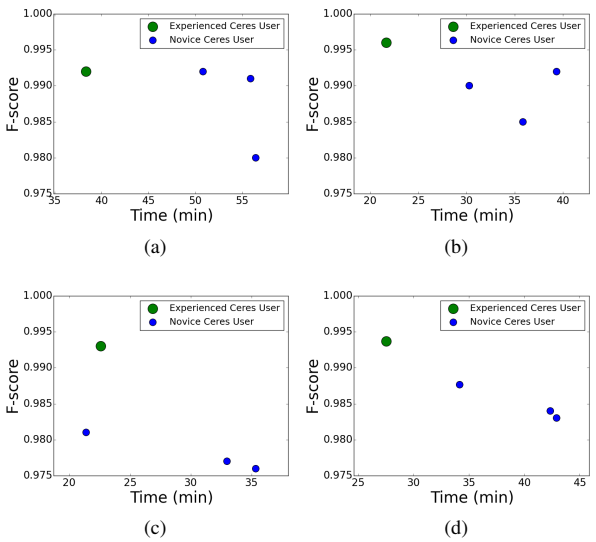


Figure 7: Clock time versus accuracy for novice and experienced *Ceres* users to generate one page: (a) page 1, (b) page 2, (c) page 3, (d) average performance.

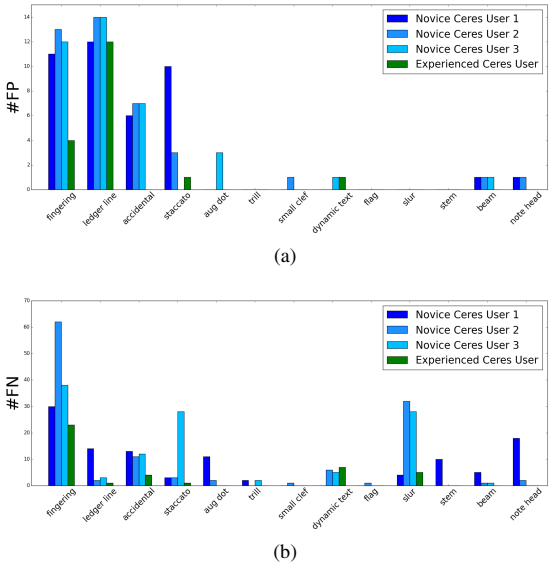


Figure 8: Distribution of *Ceres*-user-generated (a) false positive and (b) false negative errors with respect to their primitive labels on all the three pages.

Sonata No. 23, (the "Appassionata"), having 1606, 1501, and 1651 primitives respectively. We annotated the symbol primitives by hand with an interactive tool, thus creating our ground truth. In doing this we left out a few symbols appearing in the document that our system does not yet handle: grace notes, text and *fermati*. Our ground truth accounts for the overwhelming majority of what appears on the pages.

After recognition we decompose our structured results into an unstructured list of primitives, counting both false positives and false negatives. A recognized symbol counts as a false positive if its distance to all ground truth primitives of the same type is greater than some threshold. Analogously, a false negative occurs if a ground truth primitive is not sufficiently close to any recognized primitive. All symbol-to-symbol distances are measured in terms of a fixed reference point on each symbol.

Our subjects contained both novice users having about an hour of training, and more experienced users who were involved in the development of the user interface. Figure 7 shows both clock time and accuracy (an F-score) measured for these test subjects separately on each page. The figure shows overall error rates in the range of 1%, short of our eventual goal, but also showing that highly accurate results are within reach. The effect of user experience is evident both in accuracy and speed, though it is worth noting that the novice users were still able to get usable results from *Ceres*.

A primitive-level breakdown of errors is detailed in Figure 8, which counts both false positives and negatives by each class and user. A number of the errors are due to small symbols, such as augmentation dots, *staccato* markings, and short slurs. As illustrated in Section 3, our system superimposes the recognized results over the original image, usually making recognition errors obvious, though they are occasionally hard to see with small symbols. This highlights the need to explore better ways of visualizing results. The fingering errors were mostly due to our system's inability to recognize markups in non-standard positions such as to the side of a note head — an issue we have since accounted for. One can also see that a number of the errors come from ledger lines. This is due to a bug causing our system to occasionally produce syntactically impossible configurations of these primitives. These observations show the virtue of primitive-based evaluation since the errors are easily traced to their root causes.

We wanted to compare with a system other than our own, though between-system comparisons in OMR are challenging due to differences in the representations of both intermediate and end results. While commercial score-writing programs have different goals than OMR systems, both create symbolic representations of music documents that can be used to generate score images. Aside from these basic similarities there are a great many differences that may call comparisons into question. Still, in order to gain a point of reference for evaluation we compared our results with the commercial score-writing program, *Sibelius*.

Due to the steep learning curve involved with this program we engaged a professional *Sibelius* user with many years of professional experience, charging him with the task of recreating the original notation from scratch according to our test images. Even when directed otherwise, music copyists can substitute equivalent or nearly equivalent notation making it impossible to find a one-to-one correspondence between primitives. Thus we could not make meaningful accuracy comparisons with *Sibelius*.

However, we can compare the time to create the symbolic results as in Figure 9. This figure shows the necessary clock time to create the various pages. The results vary

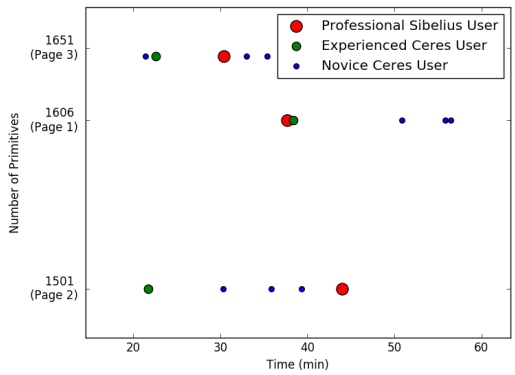


Figure 9: Clock time versus number of primitives for each page. Each point represents a single user from one of the following three groups: professional Sibelius user, experienced Ceres user or novice Ceres user.

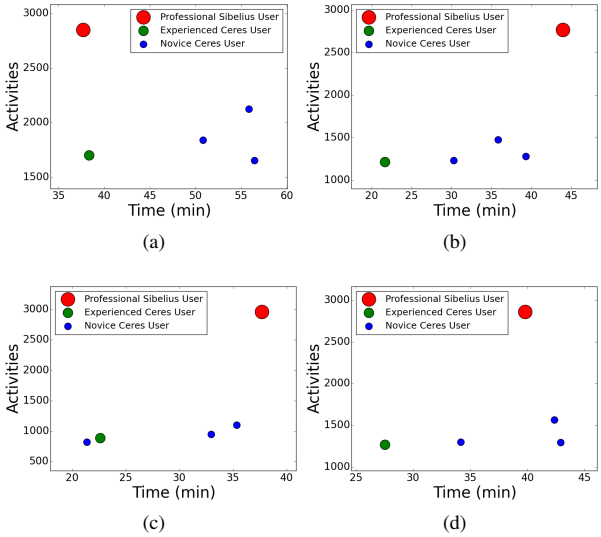


Figure 10: Clock time versus number of mouse and key activities used for each page: (a) page 1, (b) page 2, (c) page 3, (d) average performance. Each point represents a single user from one of the following three groups: professional Sibelius user, experienced Ceres user or novice Ceres user.

significantly from page to page, but show *Ceres* as competitive in all cases, and on two out of three pages showing significantly faster results than *Sibelius* with far less experienced users. Fig. 10 shows a similar comparison using keystrokes and mouse clicks as the measure of user effort. Here *Ceres* is seen to be considerably more efficient than *Sibelius* producing the results with much less user activity. This is because notation programs require detailed supervision while our system offloads as much work as possible to the machine. We also see from this figure that the experienced *Ceres* user makes more efficient strategies by minimizing the number of mouse and key actions.

The representations produced by these two systems have little in common, with different strengths and weaknesses for generating notation. *Sibelius* understands more of the inherent relations between symbols which must be preserved under renotation. *Ceres* captures a great deal of information about notational decisions (groupings, stem directions, spacing, etc.) which is also useful in renotation. The images at the website below [1] show notation generated from the two representations thus allowing for a subjective comparison of the two representations for renotation.

5. CONCLUSION

We have proposed a human-in-the-loop scheme for OMR that addresses several of the core difficulties of OMR. By allowing the user to select parameters of the models and symbol vocabulary, we deal with the heavy tail of rare symbols and notational exceptions. We also address the fundamental challenge of segmentation through recognition — now facilitated by a human guide. Finally, we demonstrate a feasible means to achieve the level of accuracy we believe is essential for successful application of OMR. The experiments show that our system has the potential to produce high-quality symbolic data more efficiently than a score-writing system, though we believe we must still improve significantly on this benchmark for our system to gain acceptance.

One promising application of *Ceres*-generated data is *renotation*. The current renotated pages are basically a one-to-one reconstruction of the original score, essentially denoising the image. We continue to explore more general renotation problems allowing various transformations of existing notation such as reformatting into arbitrarily-sized rectangles, transposition, and construction of parts from a score. This line of work will facilitate the flexible rendering of scores for electronic music readers.

We also remain engaged with improving the performance of our system. On one hand, we must continue to refine the core recognition abilities of our system, as these promise to improve both accuracy and speed of our system by handling a greater proportion of the work through automatic means. On the other hand we see considerable room for improvement and creativity in constructing the interface. We are interested in intelligent interactive approaches that increase the efficiency of the approach, e.g. automatic planning of human-machine-collaborated actions to minimize time cost, active prediction of human labeling and model selection, intelligent aggregation of multiple constrains through MIDI keyboard input, or adaptive learning to better recognize new documents. These interesting discussions are a part of our future plan.

[1] http://music.informatics.indiana.edu/papers/ismir16/

6. REFERENCES

[1] Sharpeye. `http://www.music-scanning.com/sharpeye.html`.

[2] Smartscore. `http://www.musitek.com`.

[3] P. Bellini, I. Bruno, and P. Nesi. Assessing optical music recognition tools. *Computer Music Journal*, 31(1):68–93, 2007.

[4] Taylor Berg-Kirkpatrick and Dan Klein. Improved typesetting models for historical ocr. In *ACL (2)*, pages 118–123, 2014.

[5] H. Bitteur. Audiveris. `https://audiveris.kenai.com`, 2014.

[6] D. Blostein and H. S. Baird. A critical survey of music image analysis. In H. Bunke H. S. Baird and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 405–434. 1992.

[7] Nicholas J Bryan, Gautham J Mysore, and Ge Wang. Source separation of polyphonic music with interactive user-feedback on a piano roll display. In *ISMIR*, pages 119–124, 2013.

[8] Donald Byrd and Megan Schindele. Prospects for improving omr with multiple recognizers. In *ISMIR*, pages 41–46, 2006.

[9] Liang Chen and Christopher Raphael. Human-directed optical music recognition. *Electronic Imaging*, 2016(17):1–9, 2016.

[10] G. S. Choudhury, T. DiLauro, M. Droettboom, I. Fujinaga, B. Harrington, and K. MacMillan. Optical music recognition system within a large-scale digitization project. In *ISMIR*, 2000.

[11] B. Couasnon and B. Retif. Using a grammar for a reliable full score recognition system. In *ICMC*, pages 187–194, 1995.

[12] H. Fahmy and D. Blostein. A graph-rewriting paradigm for discrete relaxation: Application to sheet-music recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(6):763–799, 1998.

[13] I. Fujinaga. Optical music recognition system which learns. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 1785, pages 210–17, 1993.

[14] I. Fujinaga. Exemplar-based learning in adaptive optical music recognition system. In *ICMC*, volume 12, pages 55–56, 1996.

[15] Ferguson G. and Allen G. Trips: An intelligent integrated problem-solving assistant. In *Proc. of the 15th National Conf. on Art. Intel.*, pages 567–573, 1998.

[16] Rong Jin and Christopher Raphael. Interpreting rhythm in optical music recognition. In *ISMIR*, pages 151–156, 2012.

[17] G. Jones, B. Ong, I. Bruno, and K. Ng. Optical music imaging: Music document digitisation, recognition, evaluation, and restoration. *Interactive Multimedia Music Technologies*, pages 50–79, 2008.

[18] G. E. Kopec, P. A. Chou, and D. A. Maltz. Markov source model for printed music decoding. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 2422, pages 115–25, 1995.

[19] Myers K. L., Jarvis P. A., Tyson W. M., and Wolverton M. J. A mixed-initiative framework for rubust plan sketching. In *Proc. 13th Int. Conf. on Automated Planning and Scheduling*, pages 256–265, 2003.

[20] K. C. Ng and R. D. Boyle. Recognition and reconstruction of primitives in music scores. *Image and Vision Computing*, 14(1):39–46, 1996.

[21] L. Pugin, J. A. Burgoyne, and I. Fujinaga. Map adaptation to improve optical music recognition of early music documents using hidden markov models. In *ISMIR*, pages 513–516, 2007.

[22] Christopher Raphael and Jingya Wang. New approaches to optical music recognition. In *ISMIR*, pages 305–310, 2011.

[23] A. Rebelo, G. Capela, and J. S. Cardoso. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition*, 13:19–31, 2009.

[24] Verónica Romero, Alejandro H Toselli, Luis Rodríguez, and Enrique Vidal. Computer assisted transcription for ancient text images. In *Image Analysis and Recognition*, pages 1182–1193. 2007.

[25] F. Rossant and I. Bloch. Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection. *EURASIP Journal on Applied Signal Processing*, 2007.

[26] Branson S., Wah C., Babenko B., Schroff F., Welinder P., and B̃elongie S. Perona P. Visual recognition with humans in the loop. In *ECCV*, 2010.

[27] Cox M. T., Edwin G., Balasubramanian K., and Elahi M. Multiagent goal transformation and mixed-initiative plannng using prodigy/agent. In *Proc. 4th Int. Multiconf. on Systemics, Cybernetics and Informatics*, volume 7, pages 1–6, 2001.

[28] Cox M. T. and Veloso M. M. Supporting combined human and machine planning: An inteface for planning by analogical reasoning. In *Proc. 2nd Int. Conf. on Case-Based Reasoning*, pages 531–540, 1997.

[29] V. Viro. Peachnote: Music score search and analysis platform. In *ISMIR*, pages 360–363, 2011.

[30] Alexander Waibel, Rainer Stiefelhagen, Rolf Carlson, J Casas, Jan Kleindienst, Lori Lamel, Oswald Lanz, Djamel Mostefa, Maurizio Omologo, Fabio Pianesi, et al. Computers in the human interaction loop. In *Handbook of Ambient Intelligence and Smart Environments*, pages 1071–1116. 2010.