Evaluation of Optical Music Recognition Software

Alexandre Daigle

Department of Music Technology
McGill University
Montreal, Canada

August 2020

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Arts.

Abstract

Optical Music Recognition (OMR) technology attempts to automate the task of converting musical content from digital images of scores to symbolic music files. There have been many OMR software applications specializing in different music notations developed since research in this area began over 50 years ago. Evaluations of OMR software, however, were seldomly investigated. This thesis attempts to evaluate the ability to recognize music scores by OMR software. An evaluation of all OMR software, including all music notation, is beyond the scope of this thesis. Therefore, it will be mostly limited to evaluating popular commercial OMR software for converting printed Common Western Music Notation.

Three experiments have been conducted to evaluate the output of OMR software. The first experiment is based on algorithmic evaluations of the OMR output using custom software written in Python. This step investigates how different OMR outputs can be automatically compared and the difficulties involved with such evaluations. The second experiment features evaluations of human engravers with and without the assistance of OMR. This experiment revealed that the workflow involving OMR outperformed expert human engravers for the conversion of orchestral scores. In the third experiment, a large-scale database of symbolic music files is automatically created using commercial OMR software using images from the International Music Score Library Project (IMSLP), which contains over half a million scores.

Résumé

La technologie pour la reconnaissance optique de la musique (ROM) tente d'automatiser la tâche de convertir le contenu d'images numériques de partitions de musique en fichiers symboliques. Plusieurs applications logicielles pour la ROM spécialisées en diverses notations musicales ont été développées depuis que la recherche dans ce domaine a été amorcée il y a plus de 50 ans. Cependant, l'évaluation de ces logiciels a rarement été étudiée. Cette thèse vise à évaluer l'aptitude des logiciels pour la ROM à reconnaître des partitions musicales. Une évaluation de tous les logiciels pour la ROM, y compris de toutes les notations musicales, dépasse le cadre de cette thèse. Par conséquent, elle se limitera principalement à l'évaluation des logiciels commerciaux populaires pour la ROM qui se spécialisent sur la conversion de la notation musicale occidentale imprimée (*Printed Common Western Music Notation*).

Trois expériences ont été menées en vue d'évaluer les fichiers que produisent les logiciels pour la ROM. La première est basée sur des évaluations algorithmiques qui évaluent les fichiers produits à l'aide d'un logiciel personnalisé pour la ROM, écrit en langage de programmation Python. Cette étape examine la manière dont les différents fichiers que produisent les logiciels pour la ROM peuvent être comparés automatiquement et les difficultés liées à de telles évaluations. La deuxième expérience présente les résultats obtenus lorsque des graveurs experts accomplissent la tâche de conversion avec et sans l'aide de logiciels pour la ROM. Cette expérience a démontré que les graveurs experts, avec l'aide de la ROM, sont plus efficaces à transcrire les partitions d'orchestre. Dans la troisième expérience, une grande base de données qui contient des fichiers musicaux symboliques est créée automatiquement à l'aide de plusieurs logiciels commerciaux pour la ROM en utilisant des images qui proviennent de l'*International Music Score Library Project* (IMSLP). Le IMSLP contient plus d'un demi-million de partitions.

Acknowledgments

I would like to thank my supervisor Ichiro Fujinaga for guiding me in my research and without whom this thesis would not have been possible. I am thankful for the generous research stipend granted to me by Ichiro Fujinaga and Stephen McAdams. I thank my fellow students from the Music Technology area, my colleagues at the Distributed Digital Music Archives and Libraries laboratory, and especially Néstor Nápoles and Timothy de Reuse for great ideas and conversations. I would like to thank Kit Soden for helping me with the Music Engraving "World Cup" experiment. Thank you to Anthony Wilkes and Chris Newell for taking the time to answer my questions about their commercial products and being transparent with their answers. I would like to thank Dover Publications for allowing me to share publicly the digitization's of scores I used in my experiments on Mupix. Finally, I would like to thank my parents, grandparents, sibling, and my partner for supporting me during this research.

Table of Contents

List of Tables

List of Figures

List of Acronyms

- CCARH – Center for Computer Assisted Research in the Humanities
- CWMN – Common Western Music Notation
- DPI – Dots Per Inch
- EST – Eastern Standard Time
- GB – GigaBytes
- GUI – Graphical User Interface
- IMSLP – International Music Score Library Project
- MEI – Music Encoding Initiative
- MEWC – Music Engraving "World Cup"
- MROMR – Multiple Recognizer Optical Music Recognition
- MS-OMR – Multiple Source Optical Music Recognition
- O^3MR – Object-Oriented Optical Music Recognition
- OMR– Optical Music Recognition
- PDF – Portable Document Format
- TIFF – Tag Image File Format
- URL – Uniform Resource Locator
- UUID – Universally Unique Identifier

Chapter 1 Introduction

There is a lot of music that is unfortunately hidden in plain sight today. This music is locked in pages that are only understood by people who have learned to read the cryptic symbols of music notation. Advancements in modern technology have made searching and analysis of large collections of data easier. Yet, "it is incredible that most analytical tasks that music historians perform remain largely untouched by technology" (Cuthbert and Ariza 2010, 637). There is an expectation that information pertaining to music ought to be searchable and accessible beyond simple indexing as other fields have; however, the field of large-scale music analysis has been held behind due to lack of data.

Most existing databases of musical scores only index and store digital images of scores. Very few databases, which go beyond digitized images, have full music searches. Digitized images must be converted from a visual format into a descriptive format (i.e., a symbolic music file) before full music searches can be performed. Creating this data—commonly done manually—is both time consuming and costly. One way to solve this is to use the technology of Optical Music Recognition.

1.1 Introduction to Optical Music Recognition

Pattern Recognition is a field in Computer Science devoted to creating software capable of classifying data automatically. Optical Recognition is a subset of Pattern Recognition, tasked specifically with recognizing patterns in images. Further subsets of Optical Recognition include Optical Character Recognition (OCR), which is concerned with recognizing characters from natural languages, and Optical Music Recognition (OMR), which is concerned with recognizing music notation. Thus, OMR converts images of music scores into to computer-readable symbolic music files.

The PhD dissertations of Pruslin (1966) and Prerau (1970) were the first research effort into OMR. Recent reviews of OMR technologies can be found in Rebelo et al. (2012) and Calvo-Zaragoza et al. (2020).

1.2 Music Notation

Music notation refers to a collection of visual symbols wherein each symbol represents an action, or a manner to execute an action on a musical instrument (including the voice). Each symbol in music notation is ordered in a sequence that signifies the guidelines for a musician to decode and perform its interpretation. Many different music notation systems exist, with the most common over the last two centuries being Common Western Music Notation (CWMN).[1]

CWMN was not invented by a singular musician, mathematician, or formal entity, instead it is the product of centuries of gradual adjustments to the precursors of CWMN. Music Notation is essential for preserving the musical history of non-aural traditions (Calvo-Zaragoza et al. 2020) and early forms of notation are considered to be cryptic in their interpretation without the associated oral tradition (Stayer 2013). The practice of writing music notation has changed dramatically from its early traditions engraved on paper to the most sophisticated digital music notation editing software of the 20th and 21st century we have today.

1.3 Motivation

Large-scale analyses of music are uncommon because of the lack of large and publicly accessible collections of symbolic files. There are small collections in existence largely comprising of the most famous composers. Many collections, however, are stored in different formats, and in different countries with different copyright restrictions. Even when combining the entire collection of available symbolic files online, these only represent the tip of the iceberg of human kind's recorded musical history. As public interest and awareness in old and ancient music diminishes, it becomes less likely that these pieces of music will ever be converted into

[1] In the literature, CWMN is sometimes called Common Music Notation (CMN), or Common Western Notation (CWN) among others variations. For this thesis I will use CWMN exclusively.

modern symbolic formats. The motivation of this thesis, therefore, is to prevent this possible scenario by creating a large database of symbolic files, beginning with printed CWMN.

1.4 Challenges

There are three challenges in this thesis, all related to evaluating commercial OMR software for printed CWMN. First, an evaluation method was created to automatically calculate the accuracy of the output of OMR software (Chapter 3). Second, in order to evaluate the usefulness of OMR software, an experiment was devised to compare manual encoding by human encoders against the incorporation of OMR software in the encoding process (Chapter 4). Third, a framework was developed for automatically creating a large collection of symbolic music files using OMR software. In the next chapter, a literature review on evaluating OMR software is presented.

Chapter 2 A Background on Evaluation of Optical Music Recognition

Optical Music Recognition (OMR) research has seen over 50 years of development, during which optical recognition for many types of different music notation has been explored. Many authors believe it is challenging to evaluate OMR because of a lack of standardized testing and common terminology (Jones et al. 2008, Byrd and Simonsen 2015, Hajič et al. 2016, Calvo-Zaragoza et al. 2020).

The evaluation of OMR is frequently performed by the authors of the same OMR and do not often compare its recognition against other OMR software.

I will present in this chapter, the papers concerned with evaluating OMR, describe the MusicXML file format, and underscore other research related to creating large databases using commercial OMR software. All research presented here is concerned with OMR software for printed scores in Common Western Music Notation (CWMN). The first section (2.1) of this chapter will present a manual evaluation of commercial OMR software, while Section 2.2 presents automated evaluations of OMR software using custom software. Section 2.3 presents automated evaluations of multiple OMR software as a side effect of trying to improve recognition accuracy by using a form of voting on each OMR output. The last section of this chapter presents works where large databases of symbolic music files were created using commercial OMR software, as this is the topic of Chapter 5.

2.1 Manual Evaluations of Commercial Optical Music Recognition Software

Presented here are the few papers that manually counted the errors in the output of multiple OMR software to evaluate their accuracy against each other. A complete list of commercial OMR software is available in Appendix B

2.1.1 Selfridge-Field (1994a)

This research sent a survey to 36 OMR developers in the early 1990s with questions about the developers' general aim and accomplishments thus far. Seven developers responded to the survey. Some developers reported that their work was not developed enough, while others

believed the survey asked for information that, if published, would reveal proprietary information about their software. The developers who responded are Kia-Chuan Ng (AMSR, or Automatic Music Score Recognizer), Christopher Newell (MIDISCAN), William McGee (Music Reader), Cindy Grande (NoteScan), Martin Roth (OMR), Elizabeth Botha (SAM), and Nicholas Carter (SightReader).

The participants were asked to subjectively rate their OMRs ability to recognize different types of sources (e.g., printed parts/band or orchestra, printed parts/early music, printed lead sheets, etc.). Selfridge-Field (1994a, 138) reports that although it is easy to count the number of correct objects on a page and convert it as a percentage, "the misrecognition of [an] object can obviate the correct recognition of another that is contingent." For this reason, subjective evaluations from the developers were included in addition to timed evaluations of the OMR software. The hardware used by the developers for their evaluations were not consistent between reports, making comparisons between different OMRs inaccurate. The results are shown in Table 2-1, Table 2-2, and Table 2-3.

Table 2-1 shows the reported subjective evaluations from the developers. The developers were also asked to run their OMR on one or both of two examples provided by the author and report a subjective evaluation of the results. The first excerpt selected by the author was a single violin part from Handel's opera "Radamisto," and the second was eight measures of Clementi's "Sonata in G Major" (piano music). The developers were asked to provide a subjective evaluation of the software's ability on a 5-point scale (e.g., "1" is very easy to recognize, and "5" is very difficult to recognize) for the two excerpts and the responses are available in Table 2-2.

Table 2-1 Self-reported capabilities of OMR software; from Selfridge-Field (1994a, 119).

Responses from developers that include the word "tested" indicate that the feature was researched but did not elaborate on the ability of the recognition software. The remaining text is the subjective rating given by the developers with the exception of the final row which describes functionality in the software.

Test sources	AMSR	Midi-Scan	Music-Reader	Note-Scan	OMR	SAM	Sight-Reader
Printed parts/band or orchestra	easy	somewhat easy	easy	tested	tested	being tested	tested
Printed parts/early music	—	—	—	—	—	—	tested
Printed lead sheets	—	easy	—	—	tested	—	tested
Printed scores/chamber music	somewhat easy	—	somewhat easy	tested	—	being tested	tested
Printed scores/orchestral	somewhat difficult	somewhat easy	somewhat easy	tested	—	being tested	tested
Printed scores/piano vocal (popular/folk)	easy	easy	somewhat difficult	tested	tested	easy	tested
Printed scores/choral	moderate	somewhat easy	somewhat easy	—	—	—	tested
Facsimiles/early printed music	—	—	—	—	—	—	—
Manuscripts/recent	difficult	somewhat easy	—	—	—	—	—
Manuscripts/early	—	—	difficult	—	—	—	—
Maximum number of parts in score	no maximum	16	9	no maximum	not relevant	not stated	not stated

Table 2-2 Self-reported individual accuracy of individual music symbols; from Selfridge-Field (1994a, 120).

Responses from developers marked with an X indicate that the feature was available but did not elaborate on its subjective recognition accuracy. "1" is very easy to recognize, and "5" is very difficult to recognize.

Object	Midi-Scan	Music-Reader	Note-Scan	OMR	SAM
white notes	x	5	5	4	2
black notes	1	1	1	2	2
rests	2	2	3	3	2
stems	1	1	1	2	2
beams	2	1	2	3	3
clefs	1	3	5	4	2
barlines	1	1	2	2	1
braces	x	x	x	x	x
sharp signs	1	1	2	2	2
natural signs	1	1	2	2	2
fermatas	x	x	x	x	x
slurs	x	4	x	x	x

Table 2-3 Self-reported recognition time from developers on pieces selected by the evaluator; adapted from Selfridge-Field(1994a, 129–34).

Scores circulated were the first violin part of Handel's opera Radamisto, and an 8-bar excerpt of Celemnti sonatina in G major. Developers sometimes declined (D) to answer, reported times for neither (N) suggest scores, and for whom Selfridge-Field (1994a) had little time to get a response before publishing (S). Screen Correction Time is the time taken to fix incorrectly recognized symbols. Output Time is the time to save the file to disk. It should also be repeated that the hardware was not consistent between the reports from the developers. Time is in the form minutes:seconds.

Handel							
OMR Software	**AMSR**	**MIDISCAN**	**MusicReader**	**NoteScan**	**OMR**	**SAM**	**SightReader**
Input Time	D	0:12	0:16	S	-	-	N
Image Processing Time	D	0:20	4:16	S	0:22	-	N
Screen Correction Time	D	0:30	3:05	S	-	-	N
Output Time	D	0:20	0:38	S	-	-	N
Total Elapsed Time	**D**	**1:22**	**8:15**	**S**	**0:22**	**-**	**N**
Clementi							
OMR Software	**AMSR**	**MIDISCAN**	**MusicReader**	**NoteScan**	**OMR**	**SAM**	**SightReader**
Input Time	D	-	0:22	S	-	5:15	N
Image Processing Time	D	-	8:50	S	1:03	6:10	N
Screen Correction Time	D	-	6:28	S	-	5:55	N
Output Time	D	-	0:30	S	-	2:20	N
Total Elapsed Time	**D**	**-**	**16:10**	**S**	**1:03**	**19:40**	**N**

The author expressed that the reports from the developers provided "little basis for comparison" because there was much variation in the quality of each report. Some developers chose to submit an evaluation using a self-selected score instead of using the Handel or Clementi excerpts, and some reports did not provide precise numbers (no additional detail was provided about this issue). The developers submitted the time required to perform a recognition on the pieces in addition to the five-point scale for the selected pieces (e.g., Handel, Clementi, both, or neither).

The question of how to accurately evaluate OMR software's performance was of interest to Selfridge-Field (1994a). She asked the developers how accuracy might be assessed, and their responses were quite similar to her own (e.g., by calculating the total number of symbols on a page and express the number of correct symbols as a percentage). The author underscored two areas that should be considered regarding this method. In the OMRs evaluated, none of the OMRs would attempt to recognize all the symbols listed by the author. If the formula (the number of correct symbols divided by the number of total symbols) was used to evaluate OMR, then there would be an advantage or disadvantage to some OMR software because they did not possess the features to recognize all of the objects. The total number of symbols attempted for recognition would differ, making an uneven evaluation across OMR software, but it did not explore how much these missing recognitions could skew the results. The problem was further complicated by the fact that the same categories were not used to describe objects in each of the OMR software.

2.1.2 Selfridge-Field (1994b)

Selfridge-Field (1994b) evaluated an OMR software against manually inputting music into notation software by recording the time needed to engrave the same piece using either method. The evaluation compared the time required to transcribe a symphony (aided by OMR) against encoding the same score manually; both evaluations included a correction stage. The *Breitkopf und Härtel* edition (1907) of Haydn's first symphony in D major in three movements, which contained ten pages, was used for the evaluation. In this study, the OMR software was SightReader, written by Nicholas P. Carter from the University of Surrey in Guildford, UK.

Both proof-reading and proof-hearing were methods used to verify the data engraved by the Center for Computer Assisted Research in the Humanities at Stanford University (CCARH)

software was correct. The “SCORE [format] is designed solely to capture and replicate an existing score” and “MuseData software can work from a score or from single parts or even from quite disheveled manuscript sources to a representation that supports not only notation but also sound output and certain kinds of analytical pursuits.” (Selfridge-Field 1994b, 159). For the evaluation, the author divided the scores by movement because of the different limitations in the file formats. SightReader was timed on two conversion stages, first, from TIFF file to OMR output, and second, screen editing in software. The CCARH system was also timed on two stages: “pitch and duration acquisition and syntax checking,” and second, “merging parts, printing a draft score, correcting it, and reprinting it.” The results of the evaluation are included in Table 2-4 and Table 2-5.

Table 2-4 Time taken by SightReader team to create and correct OMR output; from Selfridge-Field (1994b, 161).

The first column is the name of the movement, the second is the time required to OMR the movement, the third column is the time required to convert a SCORE file to MuseData and perform error correction, and the fourth column is the total. All times in the table are in the hours:minutes:seconds format.

Movement	TIFF –> SCORE	Hand-editing	Total
Presto	0:05:49	5:46:00	5:51:49
Andante	0:04:55	2:24:05	2:29:00
Finale	0:02:24	1:34:05	1:36:29
Total	0:13:08	9:44:10	10:07:18

Table 2-5 Time taken by the manual engraver team for engraving (stage one) and correct (stage two) the same score; from Selfridge-Field (1994b, 161).

The first column in this table is the name of the movement, the second is the time needed to manually engrave the music, the third column is the error correction stage, and the last is the total. All times in the table are in the hours:minutes:seconds format.

Movement	Stage 1	Stage 2	Total
Presto	1:55:00	4:24:00	06:19:00
Andante	1:20:00	2:52:00	04:12:00
Finale	1:05:00	2:04:00	03:09:00
Total	4:20:00	9:20:00	13:40:00

During her reflection of the results from the experiment, attention was given to the type of reproduction needed from a symbolic music file (e.g., whether it would be for listening, performance, or re-typesetting). Concerns arose because of the errors in the Breitkopf und Härtel edition of Haydn's first symphony. In this edition, there is an added natural sign in the second violin of measure 74, "which would be important to a performer" (Selfridge-Field 1994, 164). Another issue that arose during their evaluation was that "two pages of another Haydn symphony in the same meter strayed into the stack of pages representing the Finale. The scanning program was indifferent to this material, but the live data entry specialist found the abrupt change…" (Selfridge-Field 1994, 165). In closing statements, Selfridge-Field remarked that using humans was "well worth providing" because some problems cannot currently be solved with OMR (e.g., recognizing that a page was erroneously included from a different collection).

2.1.3 Reed and Parker (1996)

Reed and Parker (1996) described their Lemon OMR software implementation and compared it to two commercial OMR software (MIDISCAN and NoteScan).[2] Lemon was not designed to recognize scores with more than one voice per staff and the results are shown in Table 2-6 and Table 2-7.

[2] Links to the websites of MIDISCAN and NoteScan commercial software are available in Appendix 2.

Table 2-6 Compare the recognition accuracy of Lemon, MIDISCAN, and NoteScan; from Reed and Parker (1996, 807).

The name of the monophonic piece is in the first column, the total number of symbols counted is in the second, followed by the recognition rate for each software in columns 3–5.

Score	**Symbols**	**Recognition Rate**		
		Lemon	*MIDISCAN*	*NoteScan*
Magnificat	239	**99%**	84%	88%
Canon in D	206	**99%**	88%	73%
Concerto No. 1	247	95%	**98%**	94%
Polovetzian Dance	281	**98%**	93%	86%
The Entertainer	233	94%	**99%**	64%
Für Elise	534	**99%**	96%	83%
Moonlight Sonata	179	**96%**	**96%**	74%
Sonata in C Major	256	**99%**	95%	88%
The Four Seasons (Winter)	264	81%	**84%**	79%
Antiphonae Marianae	214	80%	87%	**90%**

Table 2-7 Compare the false identification of Lemon, MIDISCAN, and NoteScan; From Reed and Parker (1996, 807).

The names of the monophonic pieces are in the first column and the total number of symbols counted are in the second column. Columns 3–5 count the number of incorrectly classified symbols for each OMR software.

Score	Symbols	False Identifications		
		Lemon	*MIDISCAN*	*NoteScan*
Magnificat	239	**4**	39	24
Canon in D	206	**1**	2	5
Concerto No. 1	247	9	**5**	**5**
Polovetzian Dance	281	**6**	9	**6**
The Entertainer	233	**3**	**3**	7
Für Elise	534	**5**	30	79
Moonlight Sonata	179	**0**	3	23
Sonata in C Major	256	**2**	5	16
The Four Seasons (Winter)	264	24	**18**	25
Antiphonae Marianae	214	11	16	**10**

2.1.4 Ng and Jones (2003)

Ng and Jones (2003), at the 1st MUSICNETWORK workshop, announced that they had created an evaluation for Optical Music Recognition (OMR) software. The survey featured questions written by Ivan Bruno and Paolo Nesi, and the authors accumulated 60 images of scores to test OMR software. The scores were contributed by various publishers and permission was provided to use the scores as tests. The researchers anticipated the time-consuming work required to test 60 pages and instead proposed a "quick-test" comprising of a small selection of the original 60. This "quick-test" was composed of three pages of the most common musical symbols to evaluate OMR's capabilities. The contents of the pages in the "quick-test" did not contain musical examples; they only represented the most commonly found "musical features or fundamental musical symbols" (Ng and Jones 2003, 1). These pages were created to test the capabilities of OMR software.

The questionnaire, which is part of the survey prepared by Ivan Bruno and Paolo Nesi, is available through the internet archive.[3] There is an image collection of seven monophonic scores that were associated with this questionnaire. The first three images from the seven are most likely the images from the "quick-test" referred to earlier because of the wording in questions five, six, and seven. These three questions refer to only the first three images; thus, they are likely the three scores that form the "quick-test." Questions 5–7 from the questionnaire sought to rank OMR based on the amount of time OMR saved compared to manually engraving symbols in notation software. The other questions asked participants to rank the importance of symbols, the accuracy of the recognition of three OMR software, and others. All questions from the questionnaire are reproduced here, and the results from the questionnaire were presented in Bellini, Bruno, and Nesi (2007):

1. On the basis of your experience of what you expect to have from an optical music recognition please give a vote from 1 to 10 (mini to max) about the overall quality of reconstructed images by means of the three OMR applications with reference to the original images you found in the folder. In this assessment, please take into account only the results of the reconstruction and not the quality of printing resolution.
2. This part of the questionnaire has been set up for identifying the relevance of the basic music notation symbols in the context of typical OMR application and recognition phases. On the basis of your experience, what is the importance you give in the recognition of the set of basic symbols listed below? Please, give a vote (Vote column) from 1 to 10 where 10 means the highest relevance.
3. This part of the questionnaire has been set up for identifying the relevance of the complete and composed music notation symbols in the context of typical OMR application and recognition phases. On the basis of your experience, what is the importance you give in the recognition of complete music symbols and relationships among symbols listed below? Please, give a vote from 1 to 10 where 10 means the highest relevance.

[3] https://web.archive.org/web/20200601204514/http://www.disit.org/musicnetwork/wg_imaging/Omr_Assessment/index.html

4. On the basis of your experience as user of one of more music Editors we intend to estimate the typical work time to perform the data entry operations. They are the typical actions for polishing a result of an OMR processing.

   To this end, we ask you to evaluate in term of time (seconds) the editing operations listed below. Please, fill the *Time* field, in seconds, with the average time of each operation and fill the *music editor* field with the name of the software tools you are referring to.
5. On the basis of your experience as user of music Editors, how many minutes are needed to editing from scratch the 3 pages of music examples considered?
6. On the basis of your experience as user of music Editors, how many minutes are needed to verify the correctness of the 3 pages of music examples considered, that consists in marking the errors in the page?
7. On the basis of your experience as user of music Editors, how many minutes are needed to perform the corrections of the 3 pages of music examples considered, that consists in polishing the score to make it equal to the original.

2.1.5 Rossant and Bloch (2005)

The novel OMR software presented in this paper was manually compared against the commercial software SmartScore. The authors reported that SmartScore had difficulties differentiating between staccatos and note duration dots and had difficulty recognizing symbols in close proximity to each other. They manually evaluated the global recognition rate of SmartScore to be 92% and reported an accuracy of 98.55% on 100 pages for their own system. All pages were digitized at a resolution of 300 DPI (dots per inch). Figure 2-1 shows an example of the output of their system (indicated by the letter (a) and without errors) against SmartScore (indicated with the letter (b) and with red annotations indicating errors). The novel software (a) had no errors for these specific excepts. The researchers suggested that OMR output could be improved by using simple musical grammar (i.e., fuzzy modeling symbol classes for flexible labeling of ambiguous symbols and correction using the order of music symbol as rules).

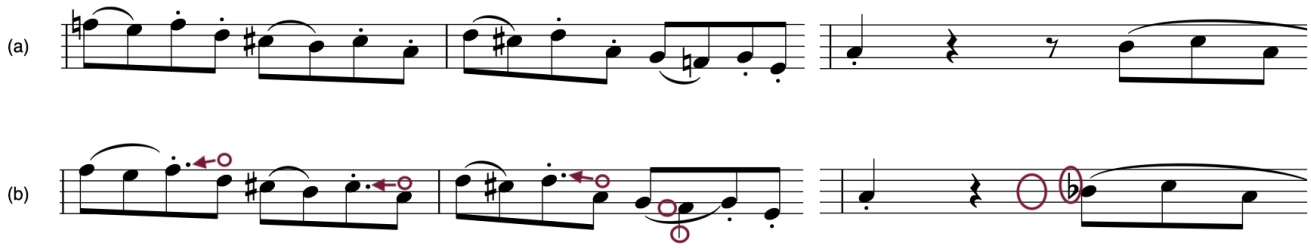


Figure 2-1 Recognition of novel software (a) and SmartScore (b); adapted from Rossant and Bloch (2005).

This redrawn figure reveals two of the errors examined by the authors Rossant and Bloch (2005). The first being the misattribution of connected components and the second being the incorrect identification of complete symbols. The slur in the second measure was not annotated as an error in the figure by the authors.

2.1.6 Toyama, Shoji, and Miyamichi (2006)

This paper compared their novel OMR software against the ScoreMaker commercial OMR software. The authors report that their software, which used template matching, had outperformed ScoreMaker. The testing corpus featured four pages of piano music with two levels of noise and two levels of note density (set A: low noise, set B: high noise, set 1: high note density, set 2: low note density). Scores were named according to the set they belong: A1, B1, A2, B2. The evaluation was on recognizing note primitives (e.g., stem, note head, beam, hook, natural, sharp, and flat). The results of the evaluation are available in Table 2-8 and Table 2-9. The recognition accuracy is calculated by subtracting the number of false positives from the correctly identified primitives and dividing the result by the total number of primitives.

Table 2-8 Recognition results of ScoreMaker; from Toyama et al. (2005, 4).

This table shows the results metric used to evaluate the ScoreMaker OMR software using symbol primitives (e.g., black noteheads, white noteheads, note stems, note beams, and accidentals).

score	black head	white head	stem	beam	hook	accidentals	# of false positive	recognition rate
A1	344/344	51/51	194/194	2/2	42/42	30/30	0	100 %
A2	515/517	3/3	359/361	67/67	0/0	123/126	0	99.35 %
B1	432/595	7/21	349/519	157/272	41/41	12/19	5	67.69 %
B2	448/478	5/5	421/441	238/276	3/7	10/13	12	91.23 %

Table 2-9 Recognition results of their proposed OMR software; from Toyama et al. (2005, 4).

This table shows the results metric used to evaluate the proposed OMR software.

score	black head	white head	stem	beam	hook	accidentals	# of false positive	recognition rate
A1	343/344	51/51	194/194	2/2	42/42	30/30	1	99.70 %
A2	516/517	3/3	361/361	67/67	0/0	126/126	1	99.81 %
B1	584/595	17/21	506/519	237/272	23/41	14/19	23	92.57 %
B2	469/478	5/5	423/441	236/276	3/7	4/13	7	92.87 %

2.1.7 Bellini, Bruno, and Nesi (2007)

Bellini et al. (2007) compared the accuracy of the author's software O^{3}MR, with those of two commercial software (i.e., SmartScore 2.0.3 and SharpEye 2.62). The testing corpus were the pages in the "quick-test" from Ng and Jones (2003). O^{3}MR (Object-Oriented Optical Music Recognition) is an OMR software from Bellini et al. (2004) and uses a neural network to identify basic symbols. For the evaluation, Bellini et al. (2007) created two test categories that mimic what they believed to be general types of people interested in the accuracy of OMR software: OMR builders and copyists. They explained that copyist would base their evaluations on the smallest of details (e.g., breath marks, fingerings, etc.), while OMR builders would evaluate on the most frequent symbols (e.g., notes, clefs, etc.). These two categoric sides are the pillars under which this paper evaluated OMR software. The result of these two categories, what they call basic symbols (i.e., symbol primitives to represent the needs of a copyist) and composite symbols (i.e., to represent the goals of OMR builders) are provided and explained further in this section.

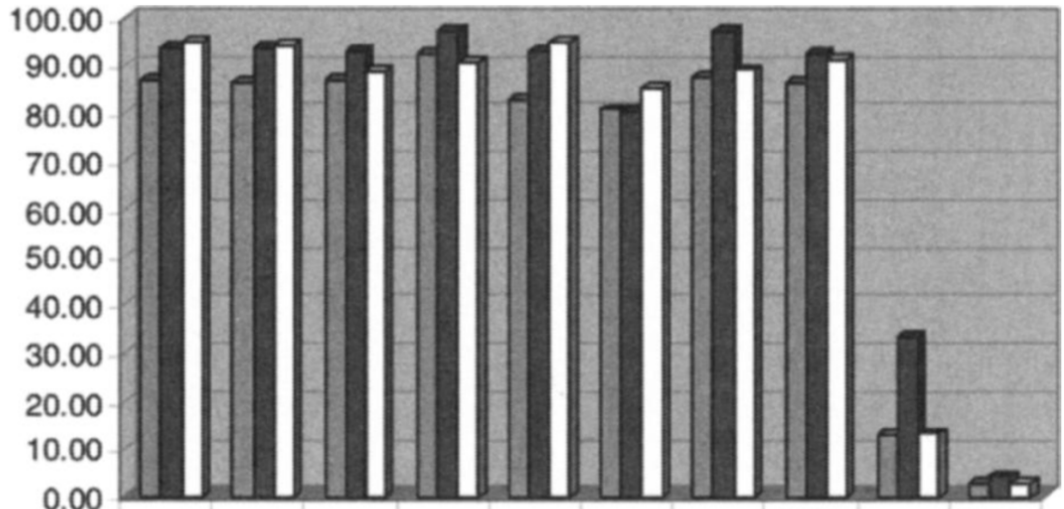
Basic symbols (e.g., noteheads, stems, etc.) were counted manually by "experts" (Bruno et al. 2007, 70). The exact number of experts is not given, however, "[t]his allowed collecting 15 scores for each test, thus a total of 105 scores" suggests seven questionnaires were filled (i.e.,

seven experts participated). The staff was not considered as an independent symbol in this evaluation. An individual symbol was only deemed recognized by an OMR when it was reproduced "in the output in its correct position with respect to the others." This means having a correct symbol but in the wrong position in the staff would still accumulate merit points in their accuracy metric. The metric based on basic symbols (i.e., primitives) was the one used to create the detailed evaluation and is similar to the expectations of OMR from a copyist category. Composite symbols were a higher-level category that groups multiple basic symbols together (e.g., rests and clefs). There was some overlap between basic symbols and composite symbols such as clefs that existed in both categories. For the composite symbols it is important to note that a correctly identified note head does not signify the correct identification of a note.

For their assessment, they limited the testing corpus to simple monophonic music, noting that the OMR result from polyphonic music was interesting but better recognition was identified with monophonic music. They did not evaluate the files created by the OMR, instead they counted errors in the previewing window of each software. The weight of their metric relied almost entirely on the recognition accuracy of the respective OMR software, by removing possible variations introduced by the ability of the software to translate the recognized output into any file format.

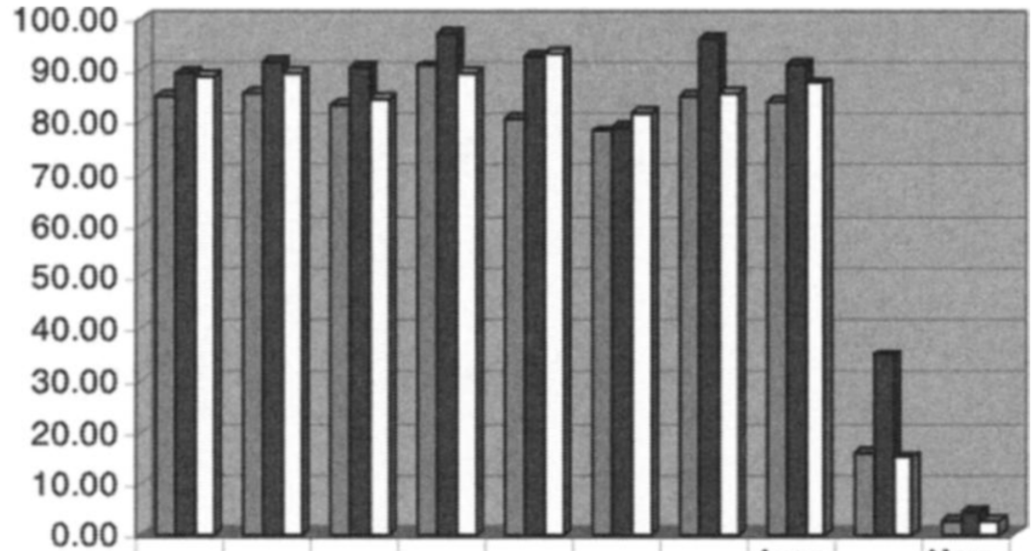
The equations for evaluating both the basic symbols and the composite symbols were the same and both evaluations count the number of expected, correct, wrong, not recognized, and extra symbols. The True Basic Symbol Recognition (TBSR) and the Total Composite Symbol Recognition (TCSR) were calculated by the number of correctly recognized symbols divided by the sum of correctly, wrong, and miss-identified symbols (as percentages). The Basic Symbol Error Rate (BSER) and the Composite Symbol Error Rate (CSER) are calculated the same way. The formula is the sum of all wrong, missed, and extra symbols divided by the sum of all symbols. The output of this equation is represented as a percentage of all wrong notes. These results are shown in Table 2-10 and Table 2-11 for the recognition rates and Table 2-12 and Table 2-13 for the error rates. TBSR and TCSR were used to calculate another metric, this new metric weighting symbols differently. Weights were assigned to individual symbols from results to the questionnaire of Ng and Jones (2003). WTBSR and WTCSR which are the weighted totals for both recognition scores (i.e., weighted TBSR and TCSR). The weighted scores are available in Table 2-14 and Table 2-15.

Table 2-10 True Basic Symbol Recognition (higher is better); from Bellini et al. (2007, 76).



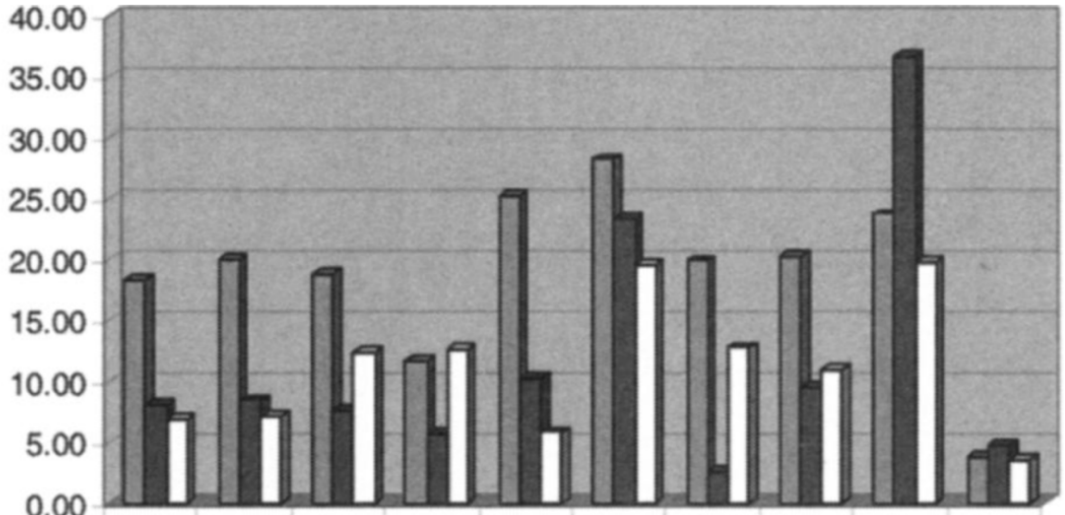
	1	2	3	4	5	6	7	Average	Var	Marg-Err
SmartScore	87.04	86.40	86.93	92.17	83.01	80.78	87.39	86.24	12.99	2.67
SharpEye 2	93.52	93.46	93.11	97.39	92.77	80.31	97.33	92.55	33.05	4.26
O3MR	94.58	93.97	88.69	90.43	94.82	85.16	89.17	90.98	13.23	2.69

Table 2-11 True Composite Symbol Recognition (higher is better); from Bellini et al. (2007, 77).



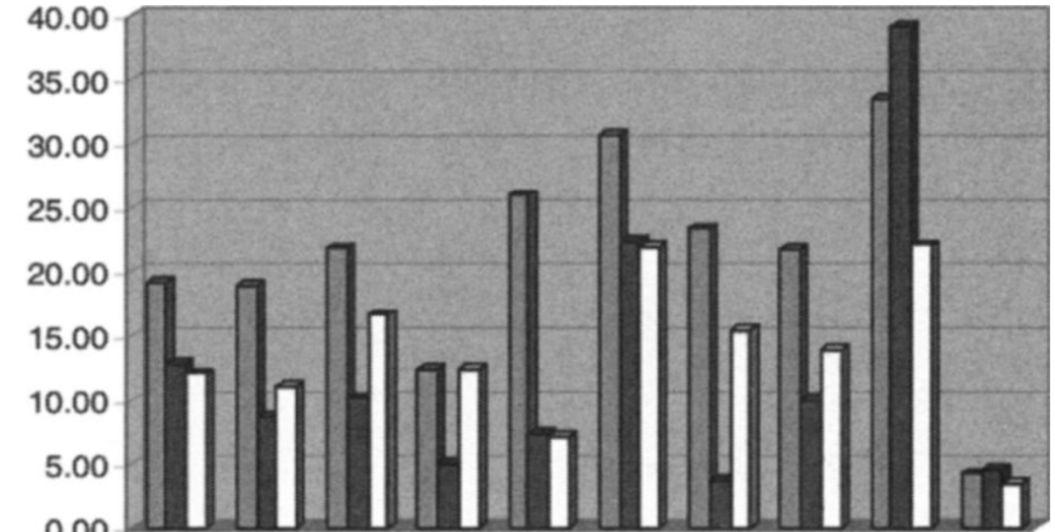
	1	2	3	4	5	6	7	Average	Var	Marg-Err
SmartScore	84.88	85.62	83.59	90.78	80.82	78.15	85.25	84.16	15.88	2.95
SharpEye 2	89.52	91.48	90.50	97.24	92.71	79.23	96.17	90.98	34.95	4.38
O3MR	88.99	89.48	84.45	89.40	93.35	81.69	85.44	87.54	15.21	2.89

Table 2-12 Basic Symbol Error Rate (lower is better); from Bellini et al. (2007, 76).



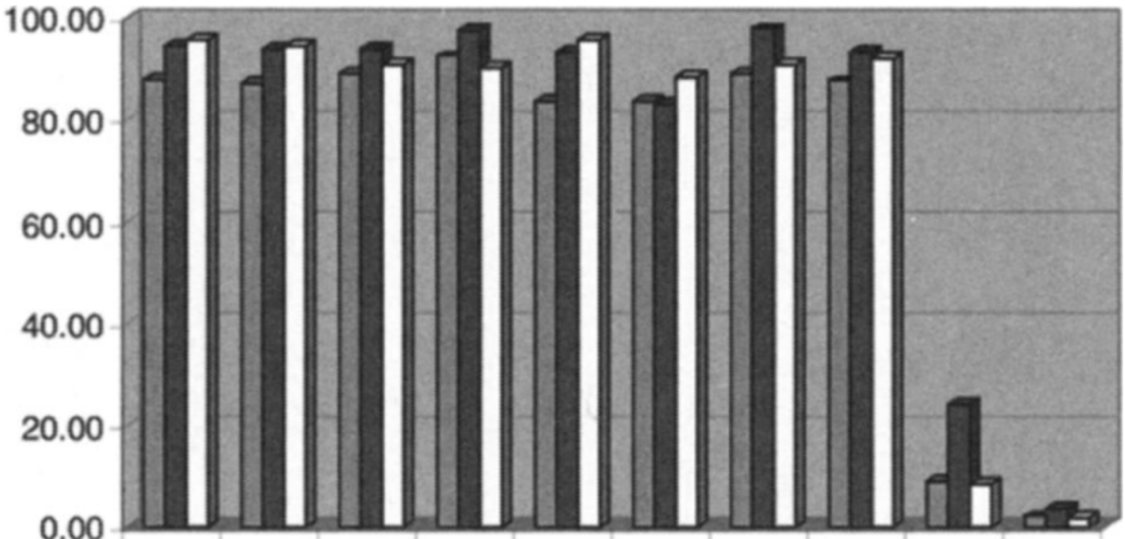
	1	2	3	4	5	6	7	Average	Var	Marg-Err
SmartScore	18.38	20.04	18.73	11.74	25.20	28.18	19.88	20.31	23.68	3.89
SharpEye 2	8.18	8.49	7.60	5.65	10.35	23.23	2.67	9.45	36.73	4.85
O3MR	7.01	7.16	12.37	12.61	5.86	19.55	12.76	11.05	19.68	3.55

Table 2-13 Composite Symbol Error Rate (lower is better); from Bellini et al. (2007, 77).



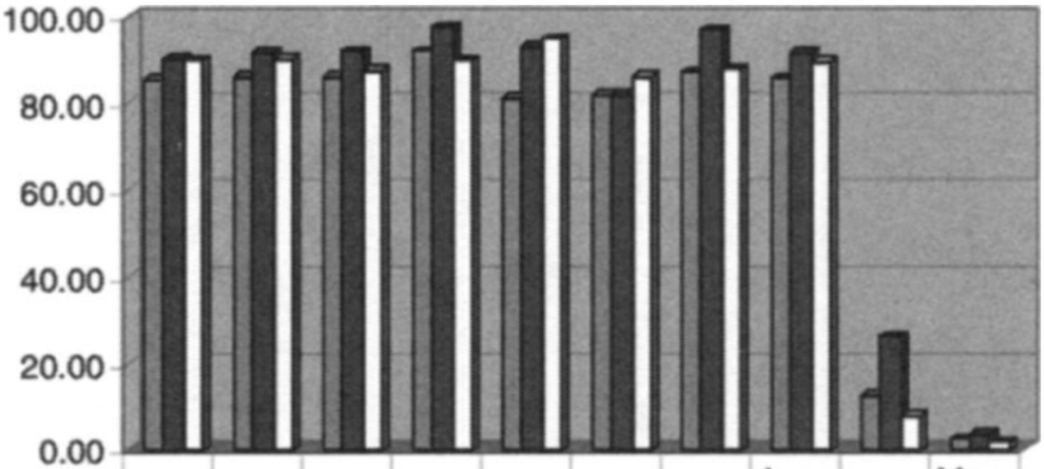
	1	2	3	4	5	6	7	Average	Var	Marg-Err
SmartScore	19.10	18.91	21.81	12.44	25.96	30.62	23.37	21.74	33.45	4.28
SharpEye 2	12.86	8.79	10.15	5.07	7.42	22.46	3.83	10.08	39.00	4.63
O3MR	12.07	11.19	16.63	12.44	7.29	22.00	15.52	13.88	22.04	3.48

Table 2-14 Weighted Basic Symbol Recognition; from Bellini et al. (2007, 78).



	1	2	3	4	5	6	7	Average	Var	Marg - Err
SmartScore	87.89	87.11	89.08	92.41	83.83	83.78	88.96	87.58	9.38	2.27
SharpEye 2	94.33	93.84	94.10	97.25	93.35	82.91	97.79	93.37	24.28	3.65
O3MR	95.44	94.66	91.02	90.14	95.72	88.30	90.93	92.32	8.55	2.17

Table 2-15 Weighted Composite Symbol Recognition, from Bellini et al. (2007, 78).



	1	2	3	4	5	6	7	Aver age	Var	Marg - Err
SmartScore	85.40	86.01	86.07	91.79	81.03	81.64	87.05	85.57	12.91	2.66
SharpEye 2	89.87	91.39	91.69	97.25	92.74	81.87	96.66	91.64	26.08	3.78
O3MR	89.78	90.12	87.13	89.76	94.47	85.71	87.62	89.23	8.04	2.10

2.1.8 Wel and Ullrich (2017)

The authors, Wel and Ullrich (2017), evaluated a sequence-to-sequence neural network against PhotoScore 8 and Capella-Scan 8 on monophonic music in printed Common Western Music Notation (CWMN) format. The proposed convolutional sequence-to-sequence model in this paper used a stack of two Recurrent Neural Networks (RNN), while a Convolutional Neural Networks (CNN) learned a feature representation of the input score. Instead of recognizing symbols, this neural network transformed the task into a translation problem. "[A]ny corpus of sheet music with corresponding digital notation could be used for training, opening up many new possibilities for data-driven OMR systems." (Wel and Ullright 2017, 1). The database used for this research was from the MuseScore database[4] where the authors filtered entries to allow only monophonic scores. They accumulated 17,000 MusicXML files and used 60% for training, 15% for validation, and 25% for evaluation.[5] All MusicXML files were split into 4-bar segments and converted to an image using the MuseScore notation editor. The images were programmatically distorted using Additive White Gausian Noise, Additive Perlin Noise, Small Scale Elastic Transformations, Large Scale Elastic Transformations, and all these techniques combined. The authors evaluated the accuracy of pitch (i.e., MIDI note number), duration, and note accuracy (i.e., a combination of pitch and duration). The results given by the authors are divided into augmented and non-augmented training and tested on augmented data and note accuracy is used for comparison. The scores for all three OMR software are available in Table 2-16.

[4] https://musescore.org/en
[5] https://github.com/EelcovdW/mono-MusicXML-dataset

Table 2-16 Recognition accuracy for clean and augmented scores; from Wel and Ullrich (2017, 5).

The first column indicates the OMR software, the "Clean" column shows the OMR results on images produced directly from MuseScore. The last column shows the OMR results using images that were programmatically distorted. The accuracy used for comparison is the note accuracy metric as a ratio between 0.0 and 1.0.

Model	Clean	Augmented
Capella Scan 8	0.53	0.14
Photoscore 8	0.61	0.09
CS2S	0.80	0.77

2.1.9 Summary

There were multiple evaluations in this section. One method used to evaluate OMR software is by sending questionnaires. Selfridge-Field (1994a) sent questions to developers, Ng and Jones (2003) proposed a set of questions, and Bellini, Bruno, and Nesi (2007) reported on the results of the questions by Ng and Jones (2003). Many problems could be learned from the research conducted by Selfridge-Field (1994a), where it was observed that few OMR developers would answer such a questionnaire, fewer would answer all questions in a similar fashion as the other participants, and not controlling for participant hardware selection makes performance metrics virtually unusable. Using questionnaires is less reproducible than other testing methodologies and, in my opinion, made these types of evolution less useful.

The second type of evaluation was to have a commercial OMR software and proposed OMR software recognize the same images of scores and compare the errors in the output of each file. Reed and Parker (1996), Rossant and Bloch (2005), Toyama, Shoji, and Miyamichi (2006), and Wel and Ullrich (2017) used the second type of evaluation to score their novel software. As a third method of evaluation, Selfridge-Field (1994b) evaluated the total time required to convert a Haydn symphony into a symbolic music file. The problems with this approach was that the definition of an error could be subjective and costly if done manually. Ideally, tests would occur on every version update of the commercial OMR software.

2.2 Automated Optical Music Recognition Evaluations

This section reviews papers that investigated how to automatically evaluate the accuracy of OMR software by parsing the output of OMR software.

2.2.1 Szwoch (2008)

Szwoch (2008) developed an automated evaluation methodology to test the accuracy of OMR. In this evaluation, a digital image that represents a score is used as input material for the OMR application. The MusicXML output of the OMR is then compared automatically with a manually engraved MusicXML file (ground truth) where errors and discrepancies are tallied, providing a rating for the accuracy of the score (see Figure 2-2). Ground truth symbolic files are usually created completely manually; however, they were made by correcting OMR output in this study. The OMR used in the research was Guido OMR (Szwoch 2007), developed at the Gdansk University of Technology. According to Szwoch (2007), this type of workflow would greatly benefit OMR's development by automatically tracking improvements in their pattern recognition algorithms.

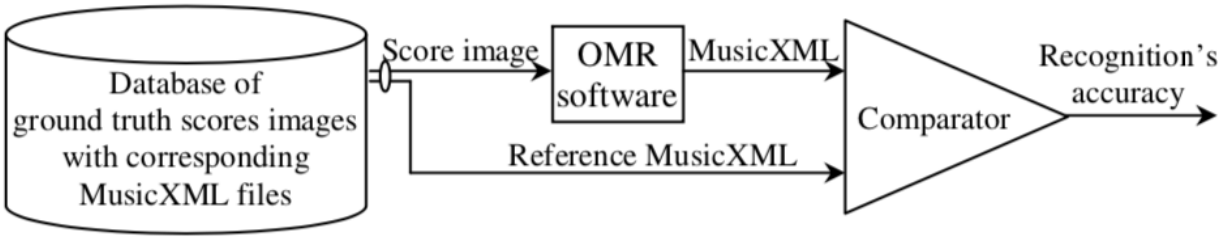


Figure 2-2 OMR development and evaluation workflow; from Szwoch (2008, 420).

Szwoch recognized the complexities of working directly with MusicXML files and decided on converting MusicXML files into a "musical notation syntax tree" for the comparison. This syntax tree ordered elements based on a musical hierarchy (e.g., score, part, voice, etc.) and, presumably, by order of appearance. The ordering of information alone is problematic because duration errors will propagate to every musical object that follows the error. The "default-x" tag in the MusicXML specification specifies how far horizontally the element should be from the left

measure line of the measure the element belongs. To get around this issue, the "default-x" tag[6] from MusicXML 1.0, which specifies the x-coordinate of a musical object within a measure, was used to align elements. The MusicXML comparator used the "default-x" tag with the notation syntax tree to order the musical objects in order of appearance from left-to-right, leaving open spaces for unrecognized musical objects (see Figure 2-3). The top staff shows the original score (the ground truth) and the bottom staff shows the OMR result. The missing music symbols, indicated by the red rectangles, will be appropriately skipped in the evaluation tool by using the default-x tag. Without the tag, musical objects will be aligned with the notes shown with arrows and count false-positive errors.

For the OMR evaluation experiment, ten score images were selected (the author provides no additional identifying information or description). MusicXML files were first produced automatically by Guido OMR. The ground truth files were produced by manually correcting the OMR output in Guido's notation editor afterwards.

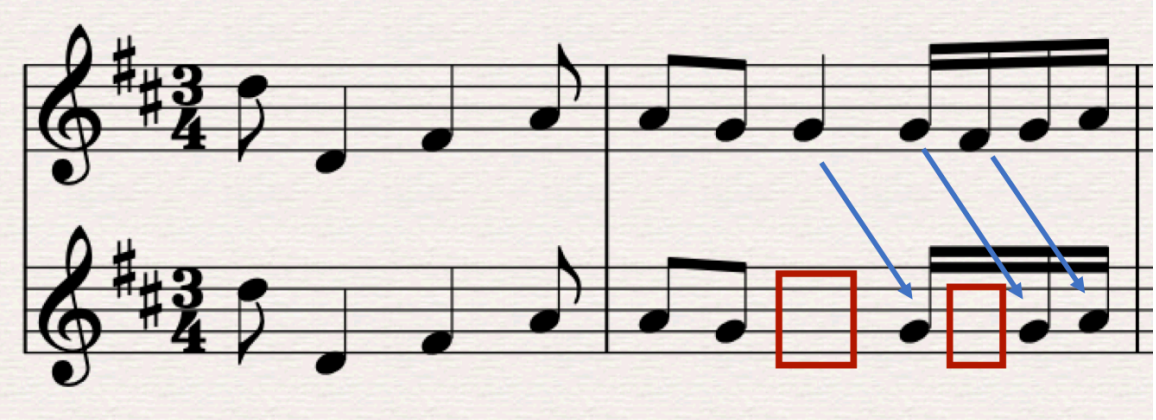


Figure 2-3 Temporal errors fixed using the default-x tag; adapted from Szwoch (2008).

The image shows the temporal advantage acquired from using the default-x metric to align MusicXML files. When an OMR software misses a note, marked by red squares, the added spacing provides additional information to correctly associate notes, marked with blue arrows, to the original score.

[6] Not all OMR software or music notation software will add the default-x tag in MusicXML. The notation software that currently adds the default-x tag include: Sibelius 7.5.1 and Finale v.25 and v.26 for Mac; those that do not add the tag include: Dorico 2.1.10.1080 and PhotoScore 8.8.2 among others.

A formula was devised to calculate an OMR's global accuracy on a document by counting the number of correct symbols, the number of extra symbols not in the ground truth, and the total number of expected symbols for each class of musical objects (see Figure 2-4). All comparisons of OMR and ground truth were evaluated manually by humans and automatically using Guido. Szwoch found similar results in the manual and automated evaluation when using a weight of 1 for all symbol classes. He emphasized that further improvements to the alignment, the weights, and the formula should be attempted in the future. The Guido OMR had a reported accuracy (A) of 92.16% using this evaluation method.

$$A = \max\left(0, \frac{\sum_{i=1}^{CN} \frac{TR_i - XR_i}{SN_i} \cdot \frac{SN_i}{SN} \cdot w_i}{\sum_{i=1}^{CN} \frac{SN_i}{SN} \cdot w_i}\right) = \max\left(0, \frac{\sum_{i=1}^{CN} (TR_i - XR_i) \cdot w_i}{\sum_{i=1}^{CN} SN_i \cdot w_i}\right)$$

CN = number of different symbol classes
SN = all symbols
TR = total correct symbols
XR = extra symbols
i = represents a specific symbol class
w_i = *weight for symbol class i*

Figure 2-4 Equation for calculating the OMR accuracy of a document; from Szwoch (2008).

2.2.2 Hajič, Novotný, Pecina, and Pokorný (2016)

Hajič et al. (2016) present an evaluation of OMR evaluation algorithms based on estimations by human participants. They created eight synthetic music scores and extracted portions of each to populate five categories: complex, full-fragment, multi-part, note-sequence, single note. The excerpts were created and exported using MuseScore. The author manually added errors based on OMR software problem areas reported by earlier research (Bellini et al. 2007, Byrd and Simonsen 2015, and Padilla et al. 2014). They created an 82-tests form with 18 control tests using 42 score images.[7] The error modifications included changes in pitch, modified accidentals, key signature errors, wrong beaming, swapping slurs for ties, and confusing noteheads in the place of staccato.

[7] https://github.com/ufal/omreval/tree/master/evaluations/test_case_corpus

Each of the 15 participants were asked which of the two "mangled" versions of the score would take less effort to correct to the ideal version of the score. The results were used to define weighted agreements according to the equation from Figure 2-5. This equation produced ground-truth edit-distance measures to be used to test automatic evaluation algorithms.

$$L_w(a,b) = \frac{1}{N}\sum_{c \in C} w^{(-a,b)}(c)\frac{|r_a(c) + r_b(c)|}{2}$$

where $w^{(-a,b)}$ is defined for an example c as:

$$w^{(-a,b)}(c) = \frac{1}{K-2}|\sum_{a' \in A \setminus a,b} r_{a'}(c)|$$

L = Simple agreement metric
c = Corpus
N = Number of examples
a = Annotators (a_1, a_2, ..., a_n)
b = Annotators (b_1, b_2, ..., b_n)
r = Ranking function
w = Weight

Figure 2-5 Equation for the Weighted Edit-Distance Agreement to correct errors; from Hajič et al. (2016).

The authors created four algorithms, which also explore different file formats to imitate the participant data found earlier. Those four algorithms are the Levenshtein distances on MusicXML files, Levenshtein distances on LilyPond files, Tree Edit Distance (TED), and Tree Edit Distance with note flattening (TEDn). To validate the results of the automated algorithms against participant data, the authors used nonparametric approaches from the field of Machine Translation.

The authors concluded that TEDn had the most agreement with participant results. Deleting a note is easy in a notation editor; however, TED calculated the edit-distance of deleting a note to be difficult because of the many properties of a note object. "[TEDn] was developed through analyzing the shortcomings of the TED metric on individual test cases" (Hajič et al. 2016, 162) and TEDn only encoded the "pitch, stem, voice, and type." They also reported that

agreement between participants appears to decrease when the experience of participants using music notation software increased.

2.2.3 Foscarin, Fournier-S'Niehotta, and Jacquemard (2019)

Foscarin et al. (2019) attempted to create a *diff* program for XML (Extended Markup Language)-based symbolic files (e.g., MusicXML and Music Encoding Initiative (MEI)) by using the Python library Music21. The Unix *diff* is a program designed to compare two files and list the changes necessary to convert one into the other. In Foscarin et al. (2019), the authors chose to work only with the MEI format because MEI has XML ids. XML ids are common attributes available to nearly all elements in MEI that serve as a unique identifier for a specific element. MusicXML does not have a comparable attribute available and XML ids are not part of the MusicXML specification. The software created by the authors is not publicly available.

The following summarizes how the program works according to the information in the paper. The authors created a sequence and two trees representing the contents of a MEI file parsed with Music21. The sequence is a collection of 2-tuples of the note pitch as a MIDI number and a note duration expressed as fractions of beats. This sequence of tuples is what the authors called a Lossy Linear Score Representation (LLSR). They then compare two LLSR from different files by applying the Levenshtein edit distance on tuple sequences. The authors created a second (i.e., beaming) and third (i.e., rhythmic) tree to include more classes for their *diff* tool, namely, note heads and rests types, note positions, beams, tuplets, ties, dots, and accidentals. Each instrumental part is a sequence of beaming and rhythmic tree pairs, with a single pair of those trees representing a single measure. Each measure is analogous to a line in a text file; therefore, a comparison considers the longest common subsequence to align measures from files, and an "inside-the-bars" comparison occurs after measures are aligned by using the size of the trees. The actual content of the trees was used for the "inside-the-bars" comparison. The comparison of used for the Beaming trees is novel. However, the comparison of Rhythmic trees was accomplished using the Zhang-Shasha equation from Zhang and Shasha (1989). The software was evaluated on 21 overtures of Jean-Philippe Rameau (i.e., evaluating OMR and manually created ground truth) and on synthetic sheet (i.e., creating pairs of symbolic files with predetermined errors). The OMR testing files were made using PhotoScore, and the ground truths were made manually. PhotoScore does not output files in the MEI format, for this task, the

authors used their own Gioqoso software to convert MusicXML files into MEI from earlier research in Foscarin et al (2018). Foscarin et al. (2019, 7) claimed that "[Gioqoso] could only handle correctly encoded scores, and that such faulty notations [from PhotoScore] will result in an error" preventing the evaluation of many of the overtures from Jean-Philippe Rameau.

2.2.4 Summary

Szwoch (2008) proposed software which aligned and evaluated MusicXML files, Foscarin, Fournier-S'Niehotta, and Jacquemard (2019) proposed software that does the same for MEI files and both software return a kind of edit distance. Hajič et al. (2016) compared edit distance algorithms by using human estimations as ground truth.

In my opinion, there are two problems with the work from Szwoch (2008) which hinder its utility in OMR research. The first problem in Szwoch (2008) is that most OMR software will not populate the "default-x" field in the MusicXML output which is required to align different MusicXML files together. The second problem, and perhaps more confusing, is that there exists two software related to music notation named GUIDO and Guido respectively. The first is music notation software with an associated file format and programming library of the same name. The second, the Guido software, is OMR software from Szwoch (2007). The only useful information gathered from Szwoch (2007)'s work is the model describing the evaluation of OMR software (e.g., keeping a database of images and symbolic files, and comparing the output of multiple file pairs to track improvements).

Unfortunately for Foscarin, Fournier-S'Niehotta, and Jacquemard (2019), the MEI format is not supported by commercial OMR products making any comparison dependent on translations from MusicXML to MEI unreliable even if the translation is made with the Music21 library. However, the argument saying that the "XML:id"s is required appears ill advised. The XML:id is a random UUID given to each element, while a random UUID could be generated as a MusicXML file is being parsed to include an "XML:id". In my opinion there are far better reasons to use MEI instead of MusicXML. However, being that MusicXML holds more information about the graphical representation of music over MIDI, and is the only other format which is currently ubiquitous among commercial OMR software, MusicXML remains the only viable format to evaluate the OMR output of commercial products.

2.3 Evaluation using Multiple Optical Music Recognition Systems

The papers presented in this section attempt to increase the accuracy of OMR by combining the output of multiple OMRs. Because of this task, they needed to create a method for automatically, or manually, compare their results. These works are separated from the works that evaluate OMR directly in the previous section.

2.3.1 Byrd and Schindele (2006)

Byrd and Schindele (2006) tested the feasibility of combining multiple OMR software into one output to create a more accurate OMR (called multiple-recognizer OMR or MROMR). By comparing the results of the OMR output of three different OMR software, they looked for "specific rules describing their strengths and weaknesses that MROMR system can exploit" (Byrd and Schindele 2006, 42). The commercial software included in the study were PhotoScore (Professional 3), SharpEye (2.63), and SmartScore (Pro 3). They processed 25 pages of scores scanned at 300 DPI and 600 DPI with 8-bits of grayscale. The pages were composed of "about five pages of 'artificial' examples, including the well-known 'OMR quick-test' (Ng & Jones, 2003)" and the remaining 20 pages were from "published editions" without any additional information.

They manually evaluated the OMR output of 11 pages for total errors, note duration errors, and note pitch errors. They also manually created 17 "rules" to potentially improve the results when combining the output of the OMR software[8] [9] and are reproduced here:

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. Resolution (300 vs. 600 DPI) affects PhotoScore far more than the others--though sometimes it does much better at high resolution, sometimes much worse.

[8] All of the rules were found in the Wayback Machine https://web.archive.org/web/20111119085331/http://www.informatics.indiana.edu/donbyrd/MROMRPap/MultipleClassifierOMRRules.txt

[9] Of these rules only the first four are listed in the paper and the URL for the remaining rules is no longer accessible except by the archive in the Wayback Machine.

5. PhotoScore at low resolution makes the most mistakes on notes; it also adds the most extraneous augmentation dots.
6. SmartScore is by far the best at hairpins; SharpEye is the worst.
7. [7 is missing]
8. SmartScore usually/always finds pedal downs; PhotoScore and SharpEye never find them.
9. SmartScore is the best at recognizing non-slur articulation marks; PhotoScore is the worst.
10. SmartScore is the only one that ever recognizes 1st and 2nd endings.
11. SharpEye is the worst at recognizing beams, i.e., the most likely to miss them completely.
12. PhotoScore is the worst (though it's still not bad) at recognizing diatonic pitch (line/space position), especially at low resolution.
13. PhotoScore is the only one that ever recognizes octave signs.
14. SmartScore is the only one that ever recognizes arpeggio signs.
15. SharpEye is the only one that ever recognizes trill markings.
16. SmartScore is the only one that ever recognizes cross-staff beams.
17. SharpEye is the only one that ever recognizes grace notes.

Byrd and Schindele (2006) also asked expert participants for a subjective "feel" of commercial OMR accuracy. All three commercial OMR were graded on a scale of 1 (very poor) to 7 (excellent) by eight expert participants composed of librarians and graduate students from a university music school. These subjective results mirrored some of the findings from the manual evaluation; however, the authors emphasized the need for a fully automated process, both in processing the commercial OMR and evaluating each OMR output to create these rules. Software which can automate the commercial OMR and automatically evaluate the OMR output is needed because of the potential changes in the pattern recognition strategies hidden in the future versions of commercial software.

2.3.2 Knopke and Byrd (2007)

In this paper, Knopke and Byrd (2007) describe their initial implementation of an incomplete music-diffing program (similar to the Unix *diff* program, which compares the

contents of two files and lists the changes necessary to convert one file into another). The program is incomplete in the sense that this software is the "engine" implementation of the MROMR project first introduced in Byrd and Schindele (2006). When the paper was released, MusicXML was already supported by Sibelius, Finale, and the OMR output of SharpEye in 2007 (Good 2013). Knopke and Byrd (2007) strongly suggested using MusicXML as the basis to create a music-diffing program because of "widespread support." This particular program was designed with only the features needed to create a MROMR system and would serve as a stepping stone towards a future "musicdiff" software, which would have many more musicological applications. It used PhotoScore, SharpEye, and SmartScore commercial OMR software.

They identified three difficulties while working with MusicXML files for a diffing program: 1) stylistic difference in MusicXML output by different software (e.g., time-wise versus part-wise XML files), 2) errors as a product of the OMR software (e.g., dynamic wedges such as slurs which span multiple measures can be placed at different locations in the file and still look the same in the rendering), and 3) errors in the interpretation of the MusicXML specification (e.g., writing pitches as notated rather than how they sound). Some of the stylistic differences in coding MusicXML come from the freedom given with optional elements and attributes in the MusicXML specification, while others come from using the time-wise or part-wise hierarchies for representing the score. These vastly different MusicXML files that could represent the same page correctly would not create similar tree structure without accommodating the structure of the MusicXML file. It should be to no surprise that these vastly different structures to the MusicXML also prevent string diffing via the Unix *diff* program.

In this study, they chose to normalize MusicXML files and align them based on measures. Their normalization step included "converting all durations to a common timebase," make "default values explicit,"[10] and put "simultaneous MusicXML elements like notes in a chord in a standard order." Note duration element in MusicXML are indicated by durations per quarter note and are relative to the time signature. Converting to a common "timebase" would mean converting everything to the same time signature and divisions elements to the same values. For example, if the divisions tag in the attributes for the measure is 256 (meaning 256

[10] No additional details are given surrounding this statement in the text. It could mean default default-x attributes; however, this is only a guess.

divisions per quarter note), a rest with duration tag of 1024 would be a whole note in length (i.e., 256 * 4 = 1024 or a duration of four quarter notes). Simultaneous elements in MusicXML, such as notes in a chord, are written the same way arpeggiated notes are with the exception that a <chord /> tag appears in each notes that are part of the chord (except the first). This means the notes of a chord can be in any order (from highest pitch to lowest and for a three note chord) "1,2,3", "1,3,2", "2,1,3", "2,3,1", "3,2,1", or "3,1,2."[11] Once the MusicXML files were normalized, they proceeded to an optional manual error checking phase.

The alignment algorithm in Knopke and Byrd (2007) worked by aligning measures. However, after reviewing the OMR output, it appeared that many bar lines were missing or misrepresented in the OMR output. To resolve this issue, they used a "dynamic programming algorithm," which was not explained. However, the authors did reference two books on DNA sequence alignment (Durbin et al. 1998; Gusfield 1997), suggesting the algorithm could have been the Needleman-Wunsch alignment algorithm (Needleman and Wunsch 1970). The alignment algorithm used was also not described, but the paper mentions that it used edit-distance and mention using a dynamic programming algorithm. The authors explained that they used a note pitch and note duration multiplied by an adjustment to align sequences together. The exact equations are reproduced in Figure 2-6.

$$p(i,j,k) = [avg(p_i + p_j + p_k) - min(p_i + p_j + p_k)] \quad (1)$$

$$d(i,j,k) = [avg(d_i + d_j + d_k) - min(d_i + d_j + d_k)] \quad (2)$$

$$l(i,j,k) = q \times p(i,j,k) + (1-q) \times d(i,j,k); \quad (3)$$

p = pitch
d = duration
q = adjustment for the weight of pitch and duration (higher q means higher p)
i, j, k = are different OMR software
l = local score matrix

Figure 2-6 Traceback equation to populate matrix; from Knopke and Byrd (2007).

[11] It should be noted that the music notation software at my disposal (MuseScore, Sibelius, GuitarPro) always declare notes in a chord from the highest pitch first in MusicXML. This easily could have been different 13 years ago.

The experiment found that increasing the weight, *q,* gave better alignment, which reduced the weight of note durations in their alignment algorithm (i.e., because 1-q times the duration means as q grows, d approaches zero). This meant that OMR software, from the experiment, were more likely to fail on note durations over note pitches because pitches were shown to give a better alignment according to Knopke and Byrd (2007).

Although the algorithm was able to align three MusicXML files for their MROMR software (which evaluated all MusicXML files simultaneously), each additional file (which creates an additional axis in their matrix) was said to require twice the amount of computations, making the addition of OMR software to the system problematic. A maximum of five MusicXML files was mentioned as a practical limit for the alignment algorithm. Every time one of the MusicXML files did not agree with one another after the alignment, the 17 rules from Byrd and Schindele (2006) were used to discern the musical object most likely to be correct in order to create a MusicXML that is more accurate than all of the others used to construct it. They observed that the announcement of the next version of PhotoScore[12] supported their theory that using multiple recognition engines improves accuracy. The authors reported that their multiple OMR system was finished when the paper was presented but that it was not tested thoroughly yet and did not report on the testing that was performed.

2.3.3 Bugge, Juncher, Mathiasen, and Simonsen (2011)

Bugge et al. (2011) attempted to combine the output of multiple OMR programs to improve the accuracy compared to each individual OMR (i.e., Multiple Recognizer OMR, or MROMR). The authors hypothesized that OMR programs could "fail dismally" (Bugge et al. 2011, 405) at some tasks, making a combined result (MROMR software output) even worse. They also believed a requirement should exist for a MROMR software, that "no single recognizer significantly outperforms the others" (Bugge et al. 2011, 405) because even after proper alignment, the less accurate OMR(s) may introduce noise. For this study, MusicXML files from the following commercial OMR software were used: Capella-Scan (6.1), PhotoScore (Ultimate 6), SharpEye (2),[13] and SmartScore X Pro (10.2.6).

[12] PhotoScore Ultimate 5 at the time, which announced the usage of SharpEye in addition to their own Liszt engine would be used for a combined recognition improvement in the PhotoScore software

[13] Although the paper only mentions SharpEye version 2, it was most likely 2.68 because SharpEye has not had an update since 2006, and this paper was written in 2011.

A diagram that describes the steps in their MROMR workflow for two OMR software is shown in Figure 2-7. The following explains the workflow: the MusicXML files were converted (normalized) to a custom filetype called MusicXiMpLe (which is still a valid MusicXML file, however, as in Knopke and Byrd (2007), allows them to normalize a MusicXML file for evaluation), sequences of musical objects are extracted from the MusicXiMpLe file, sequences originating from different files are aligned, a voter picks the musical objects based on the rules, and a sequencer converts the newly created sequence into a MusicXiMpLe file.

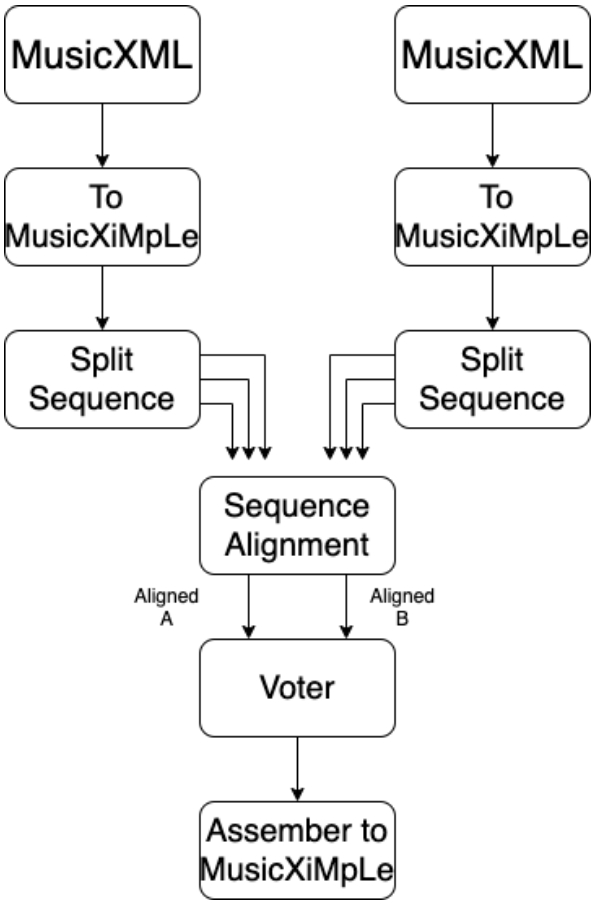


Figure 2-7 Workflow diagram for two inputs in the MROMR; adapted from (Bugge et al. 2011, figure 3).

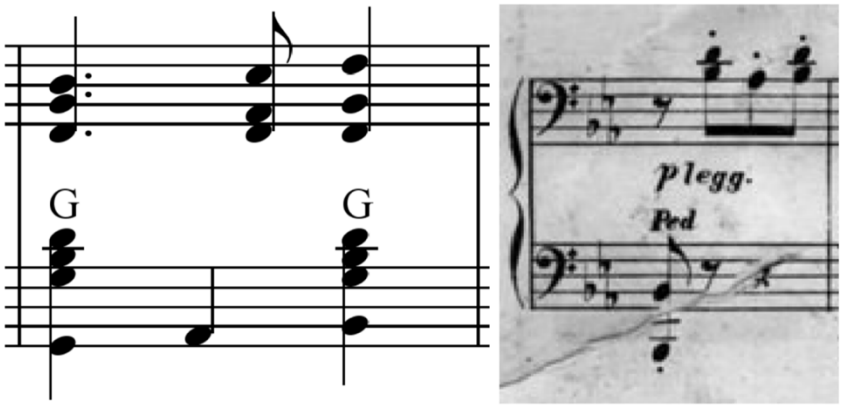


Figure 2-8 Score quality of testing corpus; from Bugge et al. (2011, 409).
Highest quality (5) left, lowest (1) right.

The test corpus featured 25 pages of CWMN divided into five categories representing their quality from bad to good and added 24 scanned pages from the Byrd and Schindele (2006) study for a total of 49 pages. An example of the score quality is provided in Figure 2-8.

When counting errors in OMR output, there can be many ways to interpret or weigh the cost of correcting errors. For this reason, the authors decided on a set of rules to decide on disambiguations of errors, and those are summarized here. If the OMR results produced notation that would sound the same as the ground truth, then the differences were not counted as errors. For example, if the OMR wrote two quarter-note rests instead of a whole note rests, it would not be regarded as an error. If the key signature or the clef were wrong, only one error would be counted for each instead of counting every pitch as wrong. Many of the disambiguations are explained in Figure 2-9. This study did not include articulations (dynamics, slurs, ornaments, arpeggiated chords, and other embellishments).

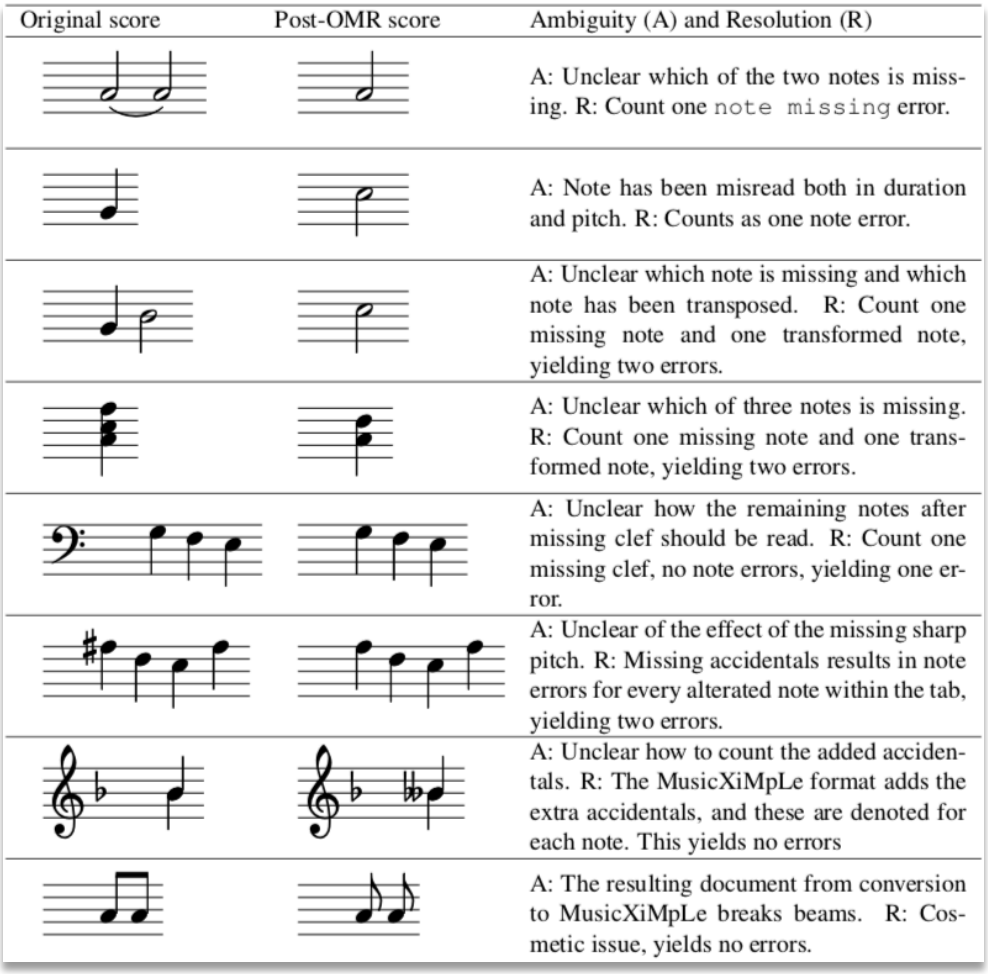


Figure 2-9 Disambiguations for errors in the OMR output; from Bugge et al. (2011, 409).

The study showed two problem areas with the alignment: aligning sequences containing chords and aligning sequences when the clef was incorrectly identified by a recognizer. The results of the study showed that using the Friedman test on a corpus of 49 scores, the combined recognizer was superior to the four commercial OMR studied. When comparing only the commercial software by themselves, they found that: Capella-Scan often made more errors than the others, SharpEye often made the least errors, and none of the OMR consistently outperformed each other. They found that OMR still requires humans for post-OMR correction.

2.3.4 Padilla, Marsden, McLean, and Ng (2014)

Padilla et al. (2014) also attempted to improve OMR by post-processing the output of multiple OMR software. The authors took the output of all available commercial OMRs and

created two different outputs (called S1 and S2) as part of their workflow. The commercial OMR used in this study were: Capella-Scan (8.0), PhotoScore (Ultimate 7), SharpEye (2.68), and SmartScore (X Pro).

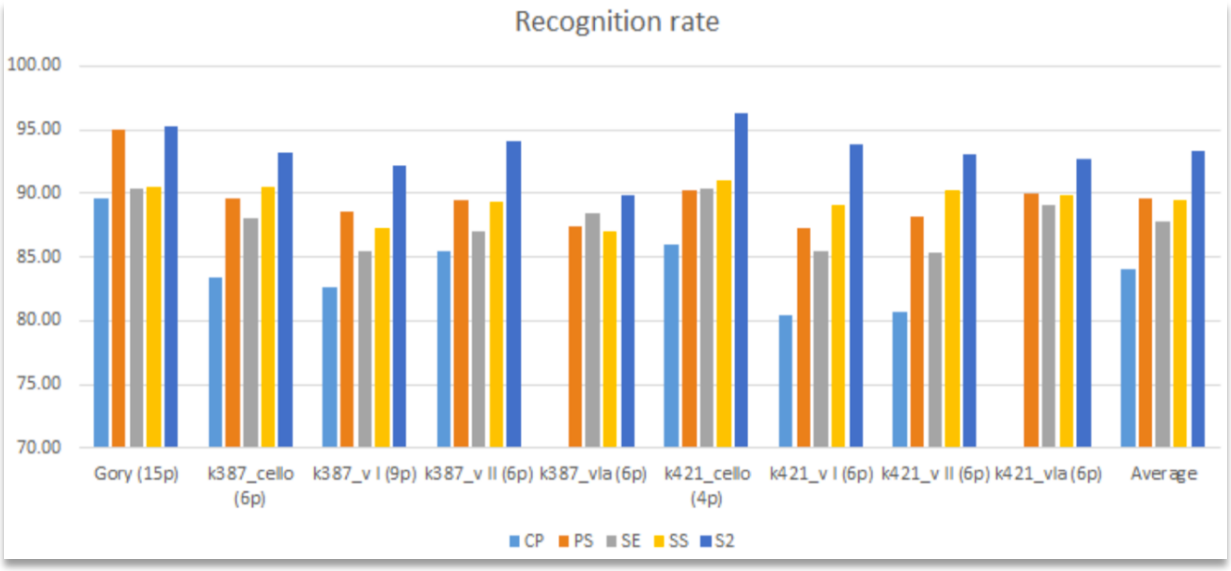
The first output (S1) in the workflow was created by automatically correcting measures based on metrical correctness. The authors based their method on work by Church and Cuthbert (2014) where rhythms were corrected in OMR output by exploiting the fact that often rhythmic patterns are repeated. Padilla et al. (2014) used Music21 to parse each the OMR output and aligned those outputs using the Needleman-Wunsch algorithm to align measures. In S1, entire measures were corrected by automatically finding the bars which had an incorrect number of beats (i.e., too many or too few musical objects for the time signature). The second output (S2) was created by using the Needleman-Wunsch sequence algorithm for alignment and voting (as in Knopke and Byrd (2007)).

In S2, as in S1, all durations were encoded using the hashing function from Music21 to align each OMR output for the Needleman-Wunsch algorithm. Measures are codified by a string of characters representing rhythms within the measure. The Needleman-Wunsch algorithm was used to align measures from two OMRs using these hashed strings to finds the shortest amounts of edits by adding gaps to create proper alignment. Then, four methods were explored to find incorrect measures.

The first method for detecting incorrect measures looked for irregular time signatures, ignoring possible anacrusis when an associated completing measure was found. The second rule took the average measure length of all measures in the score, flagging possible metric transitions based on the average number of 8^{th} notes in groups. The system then measures the average time signature in each time signature group to identify more false measures. The third rule used stylistic rules based on musical style. For example, if in a large section of 16^{th} notes lie an 8^{th} note, then it could be assumed that the duration of the 8^{th} note was incorrectly recognized. The fourth rule, also related to rhythm, attempted to discern when triplets are implied in the score but not written.

The ground truth for this study was created manually in a music notation software. They tested their evaluation system using Mozart's string quartet No. 14 in G major, K.387 and No.15 in D minor, K. 421. They reported that image resolution could affect the accuracy, with resolutions higher than 300 DPI giving unpredictable results (i.e., sometimes the results are

better and sometimes they are worse). On average, the best accuracy for a page was 89.5% (correct/total) from an OMR software and the average accuracy of the S2 output was 93.4%. The number of correct symbols include notes, chords, rests, barlines, key signatures, and time signatures. Figure 2-10 shows the comparisons between the commercial OMR and the output of S2. The elements counted in the accuracy are notes, chords, rests, bar lines, key signatures, and time signatures. Those objects combined create the total amount of objects, and the number of incorrect and extra objects of both reveal the number of incorrect objects.



CP = Capella Scan 8.0
PS = PhotoScore Ultimate 7
SE = SharpEye 2.68
SS = SmartScore X2 Pro
S2 = Authors proposed combined output (only the second stage)

Figure 2-10 Comparison the recognition of different OMR software; from Padilla et al. (2014).

2.3.5 Ng, McLean, and Marsden (2014)

This MROMR project was named Multi-OMR and had two methods of increasing the accuracy of OMR. With the first method, they would OMR multiple versions of the same score then aligned them for greater accuracy. With the second method they planned to use the same method discussed in Section 1.3.2: Knopke and Byrd (2007) to combine the strengths of multiple

commercial OMRs on the same score to produce a better recognition than any one OMR software on their own.

In their the first method of the project, they began with a corpus of Mozart string quartets taken from IMSLP,[14] NMA-Online,[15] MuseData,[16] Kern Scores,[17] and Mutopia.[18] It was hypothesized by the authors that multiple recognitions of the same score may improve OMR data. They planned to automatically recognize each image into the Audiveris OMR software, however, results were not reported.

2.3.6 Wen, Rebelo, Zhang, and Cardoso (2014)

The authors named this method of recognizing musical objects the Combined Neural Network (CbNN[19]). The CbNN uses majority voting on multiple neural network models simultaneously which is a type of Multiple-Recognizer OMR software. The testing and training corpus were created from an original 19 pages and augmented programmatically using several distortion methods "curvature, rotation, Kanungo and white speckles, etc." (Wen et al. 2014, 421) which created a total of 380 pages. They created two different CbNN's named CNNV1 and CNNV2. CNNV1 is composed of three Multi-Layer Perceptrons (MLP), each trained on a random selection of the database divided in the following manner: 25% training, 25% validation, and 50% test set. The three MLPs from the CNNV1 go through a decision tree, if all three MLPs agree then the decision is unanimous, if two of the MLPs agree then majority wins, and if all three have different answers then the MLP with the highest probability is chosen. They repeated CNNV1 four times with random selections of the database every time. CNNV2 was created with a majority voting on all 12 MLPs from running CNNV1 four times. The accuracy for all three systems is available in Table 2-17. They concluded that improvements were obtained by using majority voting on more CNNs.

[14] https://imslp.org/wiki/Main_Page
[15] http://dme.mozarteum.at/DME/nma/nmapub_srch.php?l=2
[16] https://musedata.org/list/
[17] http://kern.ccarh.org/
[18] https://www.mutopiaproject.org/
[19] In the paper CNN is used, however, CNN is more commonly used for Convolutional Neural Network. For this reason, I chose to use CbNN instead as CNN is used elsewhere in this chapter.

Table 2-17 Reported accuracy; from Wen et al. (2014, 422).

Classes	Accuracy of NN	Accuracy of CNNMV1	Accuracy of CNNMV2
Accent	82%	91%	97%
BassClef	92%	98%	98%
Beam	91%	96%	100%
Flat	82%	91%	100%
Natural	89%	96%	99%
Sharp	95%	100%	100%
TimeN	96%	100%	100%
TrebleClef	94%	99%	100%
TimeL	96%	98%	98%
AltoClef	87%	96%	100%
Note	78%	78%	93%
NoteFlag	86%	94%	95%
NoteOpen	81%	95%	99%
RestI	88%	96%	100%
RestII	87%	98%	100%
Relation	76%	84%	100%
Breve	89%	97%	97%
Semibreve	94%	100%	100%
Dots	82%	97%	99%
Barlines	90%	99%	100%
Average Accuracy	88.04 %	95.67%	98.82%

2.3.7 Ringwalt and Dannenberg (2015)

This project, named MS-OMR, attempted to compile many editions of the same score, running recognition on the same score through OMR, and combining those scans to improve accuracy. This project expands on the ideas in Multi-OMR from Ng et al. (2014) by attributing a quality to each score inputs, which has not been done before. Ringwalt and Dannenberg (2015) hypothesized that if there are several bad-quality versions of the same score and only one good quality, the Multi-OMR system would be expected to perform poorly. By weighting the better-quality scores more heavily or removing the bad-quality scores, the authors believed they can solve this problem. They used the Kanungo noise estimation (Kanungo et al. 1993) taken from

text documents to evaluate the quality of noise in musical scores. The lower quality scores according to Kanungo noise estimation would be removed from the MS-OMR.

The authors initially gathered score images of 32 Beethoven Sonatas for Piano from IMSLP (285 scores total). The ground truth files were generated by taking the LilyPond files from the Mutopia Project (free sheet music library similar to IMSLP but with PDFs and LilyPond files) and converting them into MIDI files. These MIDI files were used as ground truth in this study. All the MIDI files were separated by movement, therefore, all the scores fed to the OMR were separated by movement also. They then removed arrangements and other versions from the testing corpus. "We successfully generated and processed 95 single-movement scores for 16 works (single movements), belonging to 8 different sonatas" (Ringwalt and Dannenberg 2015, 18). Prior to feeding all pages to the OMR software, the scores were resized to normalize the staff distance in each score to a value of 8 pixels. The OMR software used was SharpEye on individual movements and exported to MIDI, to test against the ground truth MIDI files converted from LilyPond.

To align the output, they used the Needleman-Wunsch algorithm as "[i]t minimizes the sum of the distance between each aligned element of two sequences, plus a penalty for each inserted gap. The results showed that using the Kanungo noise estimator and the Needleman-Wunsch algorithm did not produce a high enough correlation to predict which source score would produce the most accurate results.

2.3.8 Summary

Byrd and Schindele (2006) proposed the first Multiple-Recognizer OMR (MROMR) software and a software implementation was created in Knopke and Byrd (2007), where the alignment algorithm is not named. Bugge et al. (2011) created another implementation of MROMR using the Needleman-Wunsch alignment algorithm. Padilla et al. (2014) created another MROMR software that also performed some post-processing to increase the accuracy. Ng, McLean, and Marsden (2014) aligned multiple sources of the same score whereas Ringwalt and Dannenberg attempted to automatically evaluate the quality of each score to add a weight to the multiple sources. Wen et al. (2014) used multiple trained neural networks instead of commercial OMR software.

While MROMR and MS-OMR can automatically produce more accurate OMR output in theory. At this time, however, both MROMR and MS-OMR require human intervention and preparation. In MROMR, each OMR needs to be evaluated manually to discover which elements are often mislabeled in a specific OMR and the process is repeated for every new version of each software. While MS-OMR is interesting, I believe it to serve little purpose until a reliable algorithm could be discovered to automatically discern the best quality digital image of a score among a list of scores. A more practical application of MS-OMR would be to employ a fuzzy match on a MusicXML file to an audibly correct MIDI file and make automatic corrections.

2.4 Large Database of Music Created with Optical Music Recognition

This final section highlights three large databases of Common Western Music Notation (CWMN) available in symbolic music files created using the OMR process. All three databases have used commercial OMR software to create their collection. Chapter 5 of this thesis will introduce a method to create, potentially, the largest collection of symbolic music files in the world.

2.4.1 KernScores

Sapp (2005) created a database called KernScores.[20] This is a collection of symbolic music files, digitized scores, and plots of musical keys. The author reported that the database was created with the help of the SharpEye commercial OMR software and contained over five million notes in 2005. This database houses the following file formats: Kern, PDFs (of the score), MIDI, and PNGs (which are generated key-plots). The Kern files in the KernScores database do not always represent entire scores. Instead, each Kern file "represents one movement in multi-movement pieces, or an entire composition for single movement works" (Sapp 2005, 664). This is noteworthy because a PDF file can contain multiple pages. In Chapter 5, it will be shown that a similar strategy was employed (i.e., one symbolic file per page in a multi-page score). Each entry in the KernScores database, which represents a specific score or movement,

[20] https://web.archive.org/web/20200514002400/http://kern.ccarh.org/

has bibliographic information and links to each file format available that is related to the same entry.

The database was constructed using SharpEye first to create MusicXML files, which were then converted into Kern files. The original MusicXML files are not available in KernScores. The author created a program (xml2hum) to translate the MusicXML files into Kern files. On the website that describes the usage of xml2hum,[21] the program was noted to have two difficulties. The first issue is that xml2hum will generate invalid Kern files when there are more than two voices in a staff, and second, sometimes the dynamic marking data is added incorrectly between notes.

The transcriptions aided by SharpEye were mostly for solo piano or string quartets, while using the commercial software for orchestral scores was still being evaluated; both at Stanford University and Ohio State University (Sapp 2005, 665). This means there are no orchestral scores in the KernScores database. No details were given about corrections made or frequencies of errors observed from SharpEye. However, the corrections were most likely performed manually. Today, the KernScores website shows closer to eight million notes with 108,703[22] files within their database. Copyrighted material is not publicly available from this database without being a member of the project.

2.4.2 PeachNote

According to Viro (2011), the PeachNote project was inspired by imitating for music scores what Google Books did for books. Searching for a musical score by its title was already available on the IMSLP database (Guo2014), however, the author wanted to make searching musical works by melodic fragments. This functionality imitates the Google Books Ngram Viewer[23] to search for words or phrases in various books. PeachNote implemented a web-based Ngram viewer, which searches for melodic fragments from OMRed scores in the IMSLP database.

The project's goal was to "help build up the foundation needed for computational musicology research by assembling a large corpus of symbolic music data" (Viro 2011, 359).

[21] https://web.archive.org/web/20070223223227/http://extras.humdrum.org/man/xml2hum/
[22] https://web.archive.org/web/20200514002400/http://kern.ccarh.org/
[23] https://web.archive.org/web/20200721171735/https://books.google.com/ngrams

PeachNote automatically transcribed over 45,000 PDFs from the IMSLP database using SmartScore, storing the generated symbolic files. IMSLP was used to avoid publishing copyright-infringing material and because "it was the easiest collection to work with" (Viro 2011, 359). The author implemented mouse and keyboard automation tools to control the Graphical User Interface (GUI) of commercial OMR software. The commercial software tested before creating this database were: Audiveris, SharpEye, CapellaScan, and SmartScore 10.3.2.[24] The commercial OMR used for most of the PeachNote database was SmartScore, because "it currently offers the best recognition rates among the four systems we tested" (Viro 2011, 360). There are no details describing how they tested the four OMR software. The symbolic files created by automating SmartScore are not directly available. Comma-Separated Value files (CSV) of Ngram information are available for download (the files are in the format of Ngram, Year, Frequency).[25] PeachNote also provides an API (Application Programming Interface) that allows queries by Ngrams.[26]

2.4.3 Bayerische Staatsbibliothek (Bavarian State Library)

Diet (2018) reported on the digitization effort of the Bayerische Staatsbibliothek, of which Google has been a collaborator since 2007. This project features three phases: a digitization phase (20,965 scores have been digitized), a recognition phase (40,000 pages have been recognized), and a melody search phase (discussed later in this section). This project digitizes specially selected copyright-free music. Their digitization effort started in 2010, concentrating on 16th- and 17th-century music, while their conversion effort from digital image to symbolic file started in 2016.

For the recognition phase, they evaluated Audiveris, Capella Scan, SharpEye, and SmartScore when choosing which OMR to use. To choose the OMR they evaluated which OMR produced the most output (because OMR will sometimes fail to produce files at all) and subjectively evaluated the accuracy of the outputs of OMR. They decided to use SmartScore for their OMR process. The OMR results, as in Viro (2011), were not corrected after the OMR process. The library also created a melody search application,[27] which is similar to Viro (2011)

[24] The versions of the other commercial OMR are not identified.
[25] https://web.archive.org/web/20181005074259/http://www.peachnote.com/datasets.html
[26] https://web.archive.org/web/20200709212753/https://www.peachnote.com/api.html
[27] https://web.archive.org/web/20191205132731/https://scoresearch.musiconn.de/ScoreSearch/

as the user input for both searches are note sequences devoid of note durations, articulations, key signatures, clefs, or time signatures. Unlike in Viro (2011), MusicXML files are available from the melody search.

2.4.4 Summary

There are three databases which were made using OMR software. KernScores which where the symbolic files are in the Kern format, PeachNote which does not make the symbolic files public but does provide searching tools, and Bayerische Staatsbibliothek which is searchable and provides MusicXML files. PeachNote has the highest number of symbolic files reported, followed by KernScores and finally the Bayerische Staatsbibliothek at the Bavarian Library.

Chapter 3 Mupix

Mupix is a software tool designed to automatically compare differences in symbolic music files, namely MusicXML files. The software was designed to compare accuracies of OMR output against human-generated ground truth. The program is available as a command-line tool after installing Mupix with the Python package managing system (pip) or installing it from the public repository on GitHub.[28] The Mupix software was written in the Python programming language, and all output from Mupix is valid JavaScript Object Notation (JSON) to make interoperability with other future software easier. This chapter describes how Mupix functions (3.1) and reports on an experiment of comparing OMR software using Mupix (3.2). The results of the experiment will be used in the next chapter.

3.1 How Mupix Works

The following explains how the software works and how a musical score is stored in a Mupix object. A Mupix object represents the content needed to visually represent a score. When Mupix parses a MusicXML file, it uses the Music21 library[29] to extract the contents of the file. Almost any file format supported by Music21 should work in Mupix, however, only MusicXML was used in testing. When instructed Music21 recognizes the file format and translates the content of a symbolic music file into Music21's internal representation. Mupix reorganizes this data from Music21 and groups it into specific categories in a MupixObject, which is specifically designed for comparing two MusicXML files.

As noted several times by various researchers in the past; for example, Knopke and Byrd (2007), Szwoch (2008), and Bugge et al. (2011), MusicXML is notoriously complicated to compare. This is mainly because of the hierarchical nature of XML files and the ordering and the placement of different musical objects are not strictly defined in the MusicXML specification. By using the Music21 library to separate and serialize different categories of musical elements, the task of comparing files can be greatly simplified. A similar approach of using Music 21 was

[28] https://github.com/deepio/mupix
[29] https://github.com/cuthbertLab/music21

also attempted by Padilla et al. (2014), but their goal was to combine different OMR software and not to compare the current goal of comparing different OMR software.

MupixObject is the main data structure in Mupix containing seven arrays, each representing one of seven musical objects: notes, rests, clefs, keys, spanners,[30] dynamics, and time signatures. A musical object, for example, a NoteObject, is similar to an associative array representing properties and the values of each object type. A diagram depicting the relationship between the MupixObject and a NoteObject is shown in Figure3-1.

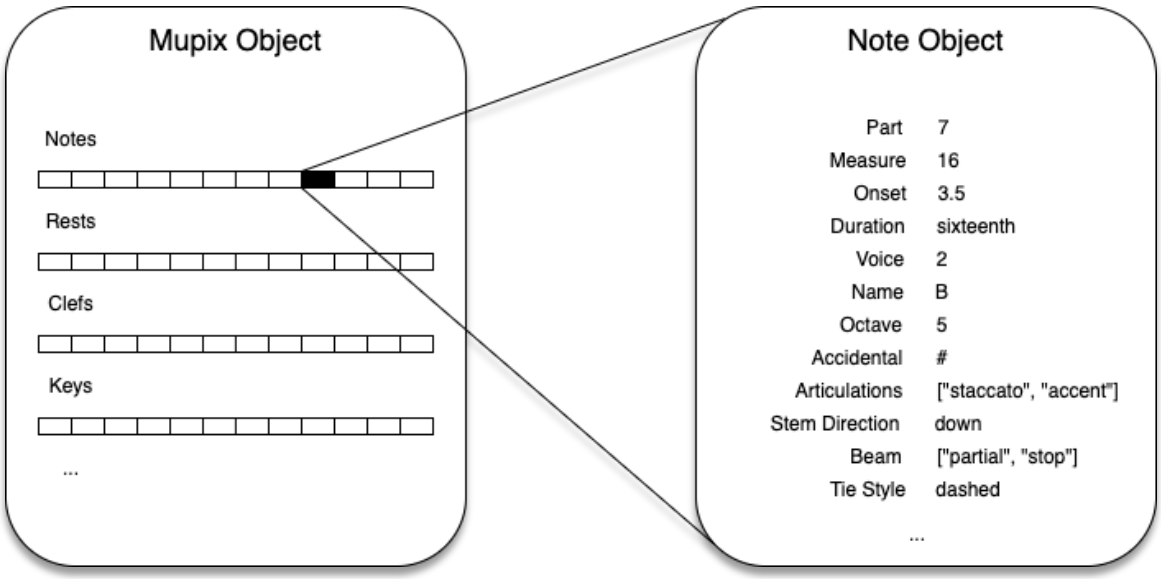


Figure3-1 Diagram of the relationship between a MupixObject, which represents a score, and a NoteObject.

Each musical object in each of the arrays of a MupixObjects have properties associated to facilitate the comparison of two MupixObjects. A NoteObject from the Notes array, for example, has a duration, a stem direction, a list of articulations, beam types, octave, and so on. Some properties are common to objects from all categories (e.g., onset and measure), and some properties are specific to a class (e.g., denominator and numerators for the objects in the time signatures array). Each property describes a possible value of a musical object in a score and those properties are counted as discrepancies or agreements when two MupixObject are compared.

[30] A spanner in Music21 is an object which is typically connected with another and can span multiple measures (e.g., slur, glissando, multi-measure rest, repeats).

3.1.1 Detecting differences with Mupix

In order to find the differences between two MusicXML files, two MupixObjects are created, each representing one file. Using Music21, each category of the MupixObjects is populated, such as notes, rests, keys, time signatures, spanners, dynamics, and clefs.

Each of these categories will be represented as an array of objects as explained above. When comparing two arrays of the same category (e.g. Notes), they may have different lengths, in which case the comparison will not be trivial. Therefore, when the number of elements in the two arrays are different, Mupix uses a sequence alignment algorithm so that they will have the same length.

The idea is to insert Null elements (i.e., empty elements) so that the two arrays will have the same length and has the least amount of mismatch. For this task, the Needleman-Wunsch algorithm is used to align two arrays by minimizing the cost of converting each musical object into another (also known as the edit distance) for the entire array. When the cost to convert one musical object into another is too high, a Null object is inserted to correct for missing elements.

Once the MupixObject categories have the same number of objects, a simple function counts and keep track of the number of differences between the two corresponding categories of MupixObjects (e.g., note beams, rest duration, spanner name, and time signature numerator).

The output of Mupix contains the total number of differences in each property of each of the seven categories of music objects in comparing two MupixObjects, which represents the two MusicXML files. For example, for a Notes array, the total discrepancies and agreements for each property of the Note object (e.g., note octave, note step, note voice, note duration, note stem direction, note beam) are reported as well as the total discrepancies for that entire category (i.e., how many NoteObjects are different).

3.2 Evaluating Mupix by Comparing OMR Software

Mupix was evaluated by comparing the errors caused by three different commercial OMR applications. First, the errors in the OMR results were counted manually by visual inspection. The results were also automatically compared with the ground truth data using Mupix to tally up the errors. This method was used to rank the three OMR software for their accuracy. The hypothesis being that the Mupix can be used as a reliable automated way to evaluate OMR

software if the rankings of both the manually counted errors and the errors counted by Mupix were similar.

The material featured pairs of pages from two sources: Beethoven symphonies (Beethoven 1976) and a Wagner opera (Wagner 1976). The pairs featured one page with the instrument names (e.g., at the beginning of a piece) and the next page the page immediately following the first without the full instrument names, see Figure 3-2 and Figure 3-3. Other page pairs were specifically selected because they had a different number of staves per system on the same page, see Figure 3-4 and Figure 3-5. These pages were included to challenge the OMR software because it would need correctly identify to which instrument a particular staff belonged. Every page was manually engraved using a notation editor and saved as the ground truth MusicXML files. The scanned score images were used as the input to the OMR software (OMR A, OMR B, and OMR C[31]). All three OMR output were manually compared against the original scores and ranked (i.e., 1st, 2nd, 3rd) by the number of errors. Each of the three MusicXML files generated by the OMR software was also compared against the ground truth MusicXML files using Mupix. The results for the Mupix were also ranked. The results are shown in Table 3-1 and Table 3-2.

Table 3-1 Manual evaluations of OMR software, marked B for Beethoven and W for Wagner.

MANUAL								
PAGE #	B60	B61	B33	B34	W346	W347	B292	B293
OMR A	2nd	1st	2nd	1st	1st	1st	1st	1st
OMR B	1st	2nd	1st	2nd	2nd	2nd	2nd	2nd
OMR C	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd

Table 3-2 Mupix evaluations of OMR software, marked B for Beethoven and W for Wagner.

MUPIX								
PAGE #	B60	B61	B33	B34	W346	W347	B292	B293
OMR A	2nd	1st	2nd	1st	2nd	1st	1st	2nd
OMR B	1st	2nd	1st	2nd	1st	2nd	2nd	1st
OMR C	3rd	3rd	3rd	3rd	3rd	3rd	3rd	3rd

[31] The actual names of the commercial software are not revealed in order to avoid possible legal complications.

The results show that in all but one case (W346) out of the eight pages evaluated, the rankings of the three OMR software applications were the same, demonstrating that Mupix seems to be able to automatically generate results comparable to human evaluation of OMR software. This is of great value because manual evaluation of OMR is an extremely time-consuming process.

As seen from the results, we found that one of the commercial OMR software consistently outperformed the other two software. We, therefore, decided to use OMR A, in order to determine whether the current state-of-the-art OMR software can compete, both in terms of time and accuracy, with human engraving process. This is investigated in the next chapter.

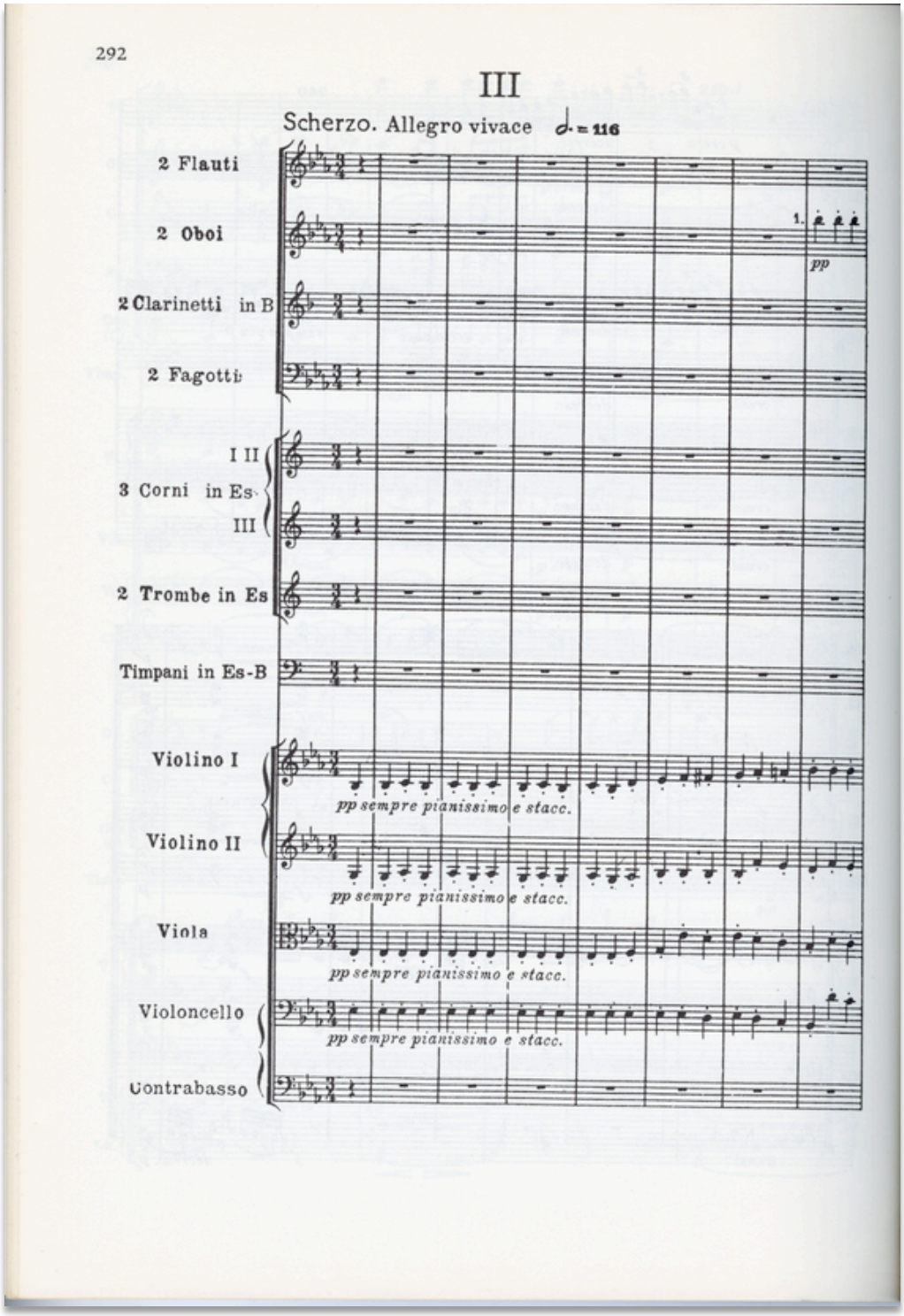


Figure 3-2 Third Symphony, Scherzo (Beethoven 1976, 292).

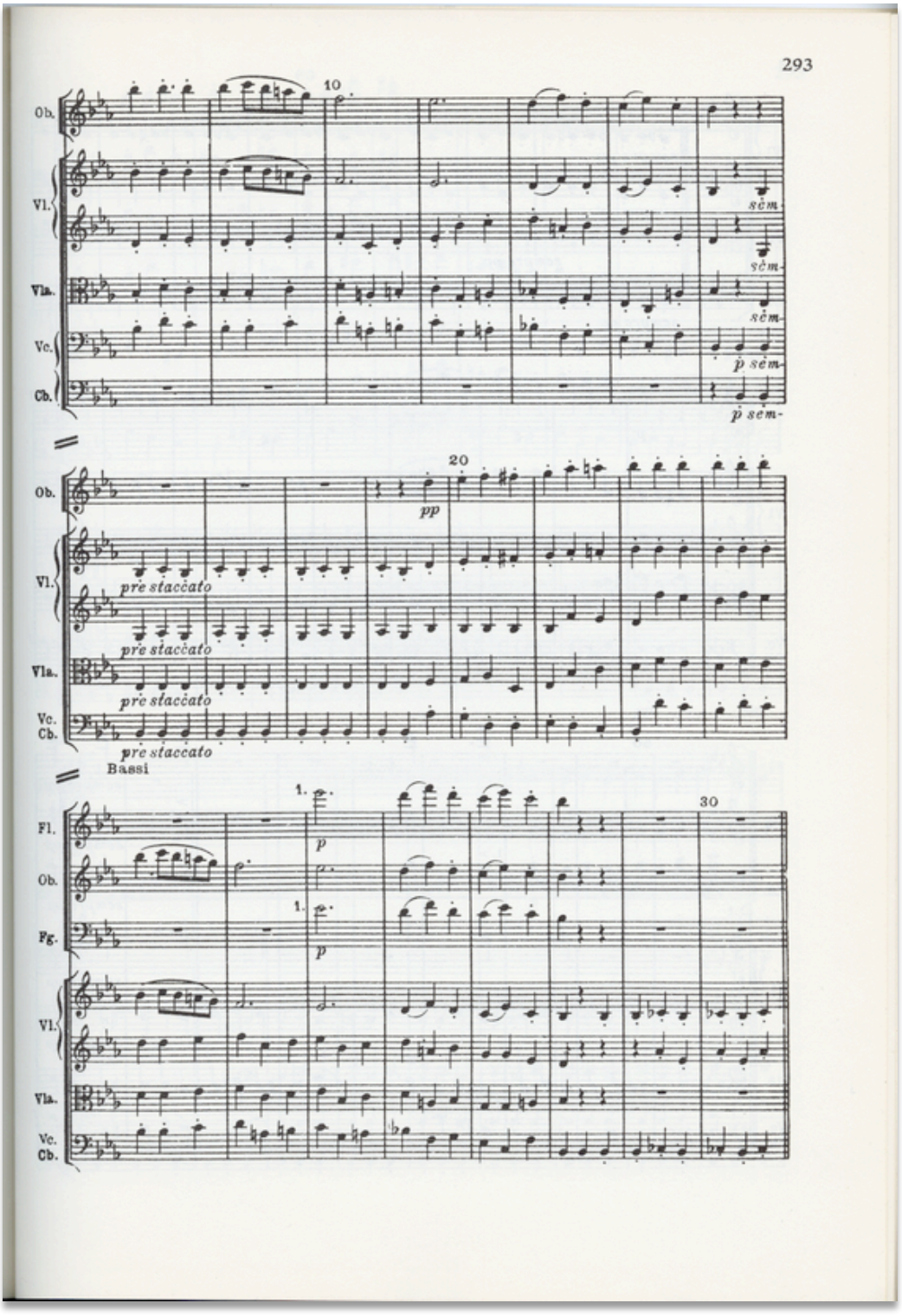


Figure 3-3 Third Symphony, Scherzo (Beethoven 1976, 293).

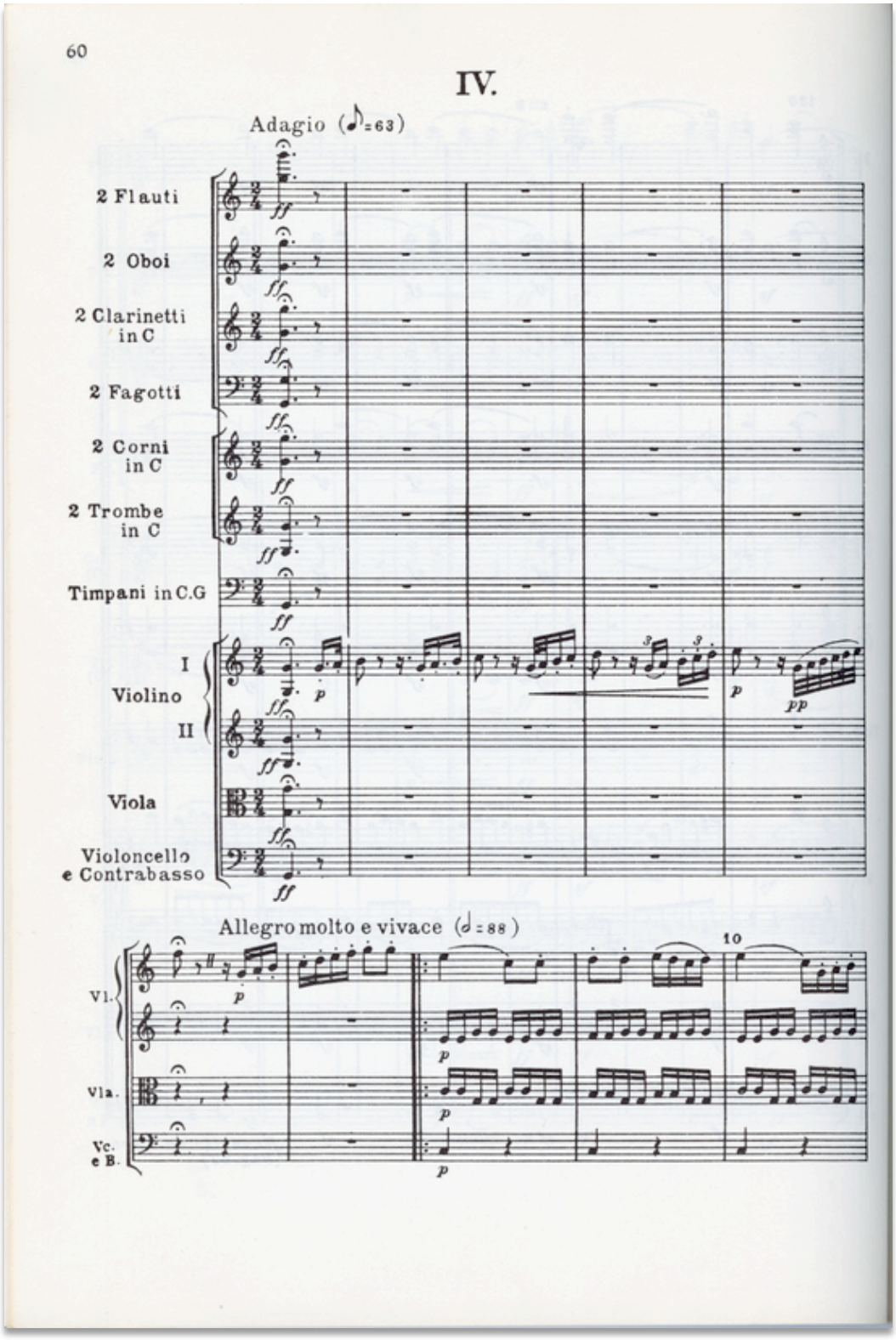


Figure 3-4 First symphony, Allegro molto e vivace (Beethoven 1976, 60).

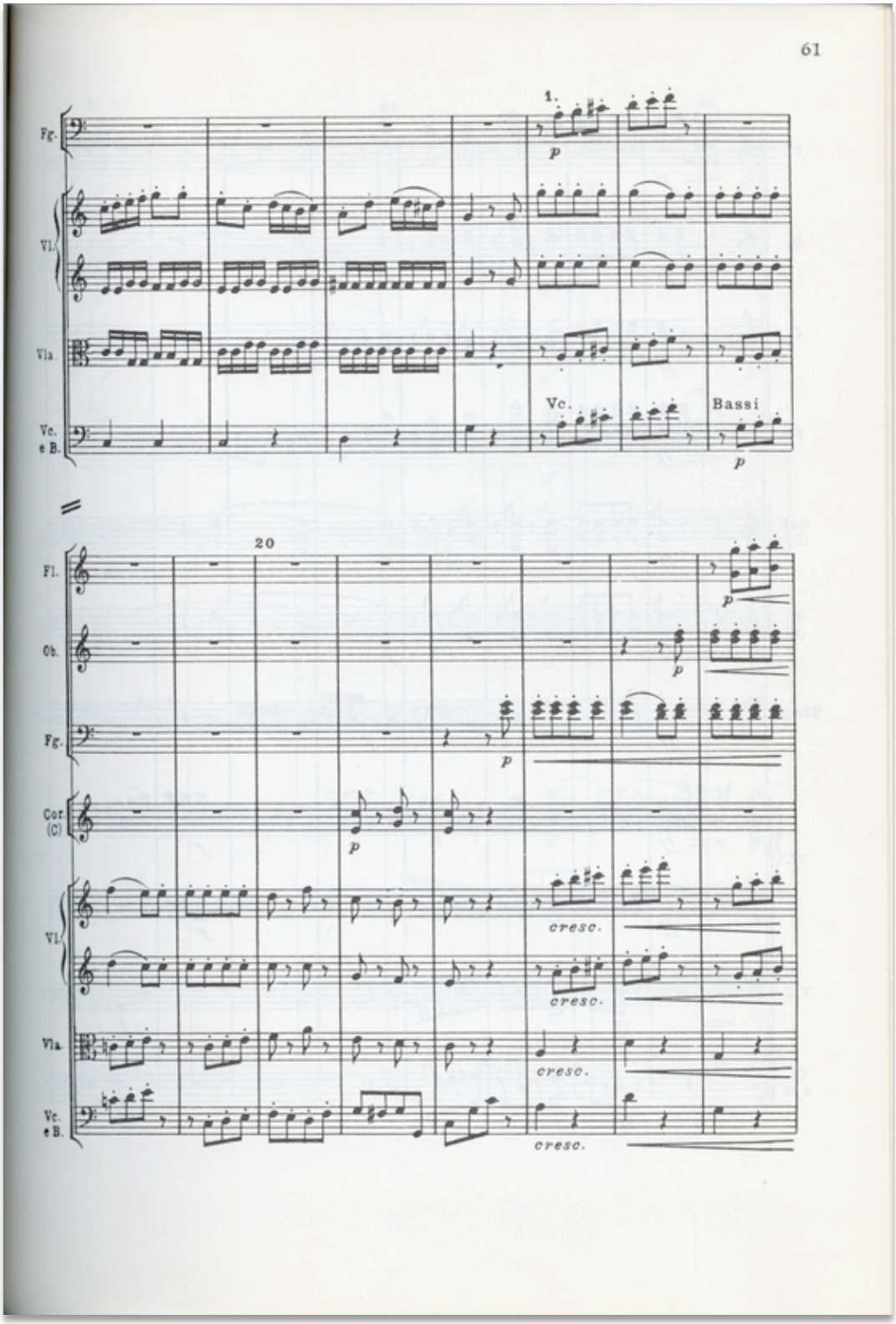


Figure 3-5 First symphony, Allegro molto e vivace (Beethoven 1976, 61).

Chapter 4 Music Engraving World Cup

This experiment, called the Music Engraving "World Cup" (MEWC), was a contest where human participants competed against each other to produce the fastest and most accurate engraving while participants unknowingly competed against a computer. This contest used orchestral scores and is quite similar to the research of Selfridge-Field (1994b) with a few changes. More specifically, the purpose of the experiment was to compare the MusicXML files produced manually by music engravers using their preferred music notation editor with the files produced by the same engravers aided by OMR software. The contest's primary motivation was to determine whether the manually engraved method of producing symbolic music files of orchestral scores is more cost-effective than the method aided by OMR software. It was hoped that this experiment's result might be useful for creating large amounts of scores in the Orchestration Analysis & Research Database (ORCHARD).[32] An ethics certificate was obtained for this research and is included in Appendix C
Ethics Certificate.

4.1 Experiment Methodology and Construction

The MEWC consisted of three stages: a qualifying stage, an engraving stage (i.e., conversion stage), and a correction stage. These three stages will be explored in detail in the following sections and how we selected and informed potential participants. The qualifying stage's goal was to identify the four "best" users of music notation software. In this experiment, a participant was considered "better" when this participant was faster and more accurate than another. The participant's score was calculated by multiplying the time in seconds with the number of errors in the MusicXML file produced. A lower score meant better. The participants were motivated to perform better than their peers with monetary incentives in each of the three stages. The Qualifying stage was the only stage that eliminated participants from the evaluation.

[32] https://www.actorproject.org/orchestration-tools-1

4.1.1 Participant Selection

To recruit potential engravers for the contest, applicants were informed about the contest through the student and faculty email lists affiliated with the Schulich School of Music at McGill University and the Center for Interdisciplinary Research in Music Media and Technology (CIRMMT). The email sent to the mailing lists explained the entire contest and had an application form as an attachment. The following is a copy of the email sent through both mailing lists.

> Are you the fastest music engraver (using Finale, MuseScore, Sibelius, etc.)? Let's put your skills to the test! We are holding a paid qualifying round to find the 4 best engravers, and these 4 will go head to head for an even bigger monetary prize and bragging rights. The competition is centered around orchestral scores; it may help if you keep a reference of instrument names in different languages nearby. This experiment is done online.
>
> **Elgibility**
> You must be at least 18 years of age. You should also have access to a computer and notation software for engraving.
>
> **The Qualifiers**
> Applications accepted to the qualifier round will be awarded a minimum of $20, and given bonuses based on the highest rankings of the evaluation committee. Participants will be asked to engrave a single page of Orchestral music that should take on average an hour. The top 3 participants in the qualifying round will be given a $35, $30, and $25 bonus respectively, and only the top 4 contestants will be given the chance to compete in the final. Contestants will be asked to do the best they can within one hour, they may only submit once. Engraving submissions will be evaluated in terms of speed and accuracy.
>
> **The Final**
> The final will occur a week after the qualifiers, giving the top 4 participants time to adjust their setup if they wish. In the first phase, two pages of orchestral music will be transcribed. In the second phase, the following week, the task will be a correction of 6 pages. Each phase should take 2 hours to complete. Monetary prizes for the final round will be as follows, $350, $200, $125, and $75.
>
> The deadline for applications is Friday 11 October 2019 at 23:59 (11:59 pm) EST, the qualifier round must be completed between Monday 14 October and Monday 21 October, and the final is from Sunday 2 November to Sunday 16 November. The winner shall be announced on December 2nd.
>
> If you are interested, please fill out and submit the attached application to: (my email address)

This research is supported and supervised by Professor Ichiro Fujinaga and Professor Stephen McAdams of McGill University.

There were 12 applicants that submitted their application within the deadline. Due to our budget, we needed to select 8 participants for the contest. The participants were first divided based on the music notation software they planned to use and within those categories, they were sorted by how many years of experience they had using the software. The notation software was represented in the following denominations: five users of Finale, three users of MuseScore, two users of Sibelius, one user of Dorico, and one applicant reported being a user of both Finale and Sibelius. Nine of the applicants reported the years of experience they had with music notation software with an average experience of 10.55 years. The years of experience varied between 3 years and 30 years of experience with music notation software. The two most proficient applicants from each notation software represented made the list of participants for the qualifying stage with one exception being Dorico as there was only one applicant. The eighth participant was chosen based on the years of experience among the remaining applicants and this participant used Finale. The distribution of users per software for this experiment was: three Finale, two Sibelius, two MuseScore, and one Dorico user.

The qualifying stage and each following stage for the evaluation were conducted online. The software on a server that controlled the contest was written in Python. The Application Programming Interface (API) was written with the Flask framework and templates for the website were created using JavaScript, HyperText Markup Language (HTML), and Cascading Style Sheet (CSS). The server was hosted from a virtual machine in the Distributed Digital Music Archives and Library laboratory with 20 Gigabytes (GB) of storage and 2 GB of RAM and 2 virtual cores on an Intel Xeon processor.

4.1.2 Qualifying Stage

The qualifying stage is the first stage where participants would compete against each other and the only stage where participants would be eliminated. This stage was implemented to find the best engravers, who were chosen based on combining the completion time and the number of errors found in their work. This stage featured eight participants and only the four best engravers would continue in the contest.

Participants were given strict instructions via an email with three links on a website specifically for this experiment at 12:01 AM (EST) on 14 October 2019. Both instructions and guidelines are explained here. Participants were given two links in the participation email, which contained additional details for the experiment (i.e., participation guidelines with examples and encoding guidelines) and the third link was a unique participation link. The instructions detailed how the scores needed to be encoded and how the files should be submitted (e.g., participation instructions and encoding instructions). On the website, multiple animated GIF (Graphics Interchange Format) files were used in tandem with text to explain to the participants how to start the experiment and what to expect from the experiment. All participants were repeatedly warned that they could only attempt the qualifiers once and to participate only when they were ready to allocate a minimum of one hour of work uninterrupted as it was estimated to take roughly an hour to complete. The participants were informed that all files submitted needed to be uncompressed MusicXML files and named a certain way (e.g., alex_1.xml, chris_1.xml, evan_1.xml). The participants were also given instructions on how the score was to be engraved in the music notation software. Those instructions were:

1. All parts must be separated into independent staves like in [replicated as Figure 4-1 here].
2. Slurs and Ties must be correctly engraved in the notation software.
3. Dynamics such as hairpin crescendos and other dynamic symbols must be appropriately attached to the proper staff, note, and bar line
4. Musical and graphical accuracy must be respected as much as possible (excluding after splitting instruments as in the first guideline from this list). All symbols must be engraved accurately (e.g., note beams, stem directions, key signatures, tempo markings, measures per staff must all copy the original score).

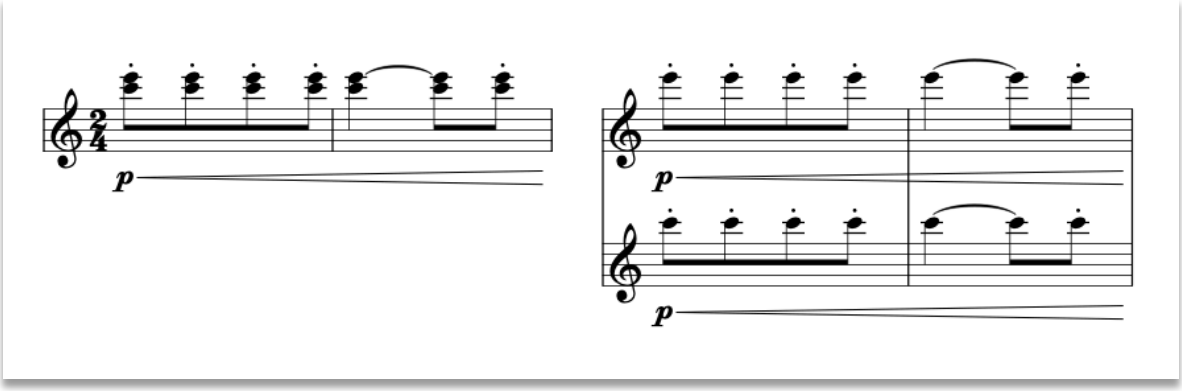


Figure 4-1 Part expansion rule for all MEWC engravings: First symphony, Allegro molto e vivace (Beethoven 1976, 61).

Once a participant was ready to download the score and clicked the "Download File" button (as shown in Figure 4-2), a timer was triggered by the click and automatically started the downloading process of the score. Each participant had a different uniform resource locator (URL) with a universally unique identifier (UUID). A log was kept on the server to accurately record when each participant downloaded the PDF file and uploaded the MusicXML file.

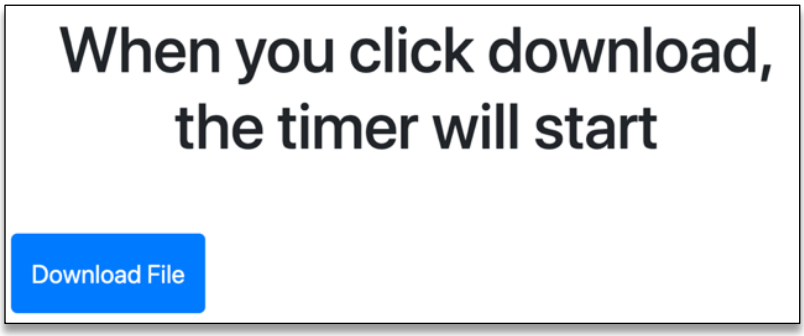


Figure 4-2 MEWC download page.

The deadline to complete the qualifying stage was at 12:00 PM (ET) on 21 October 2019, giving the participants one week to convert one page of orchestral music at their leisure using the website.

The eight chosen applicants were given a single page of orchestral music to transcribe for the qualification round. The page used for the qualification round was taken from Jean Sibelius's

"Phojola's Daughter, Op. 42" [33] as shown in Figure 4-3. Participants were given the title page with the instrument names for a reference, however, the title of the piece, all page numbers, file metadata, and other identifying marks were removed to avoid cheating by identifying the piece. The title page also featured a large red X to remind the participants that the title page was not to be converted. The results of the eight participants were evaluated and the top four participants were chosen to proceed to the Finals, which involved the engraving stage and the correction stage.

[33] https://web.archive.org/web/20200703014519/https://imslp.org/wiki/Pohjola's_Daughter%2C_Op.49_(Sibelius%2C_Jean)

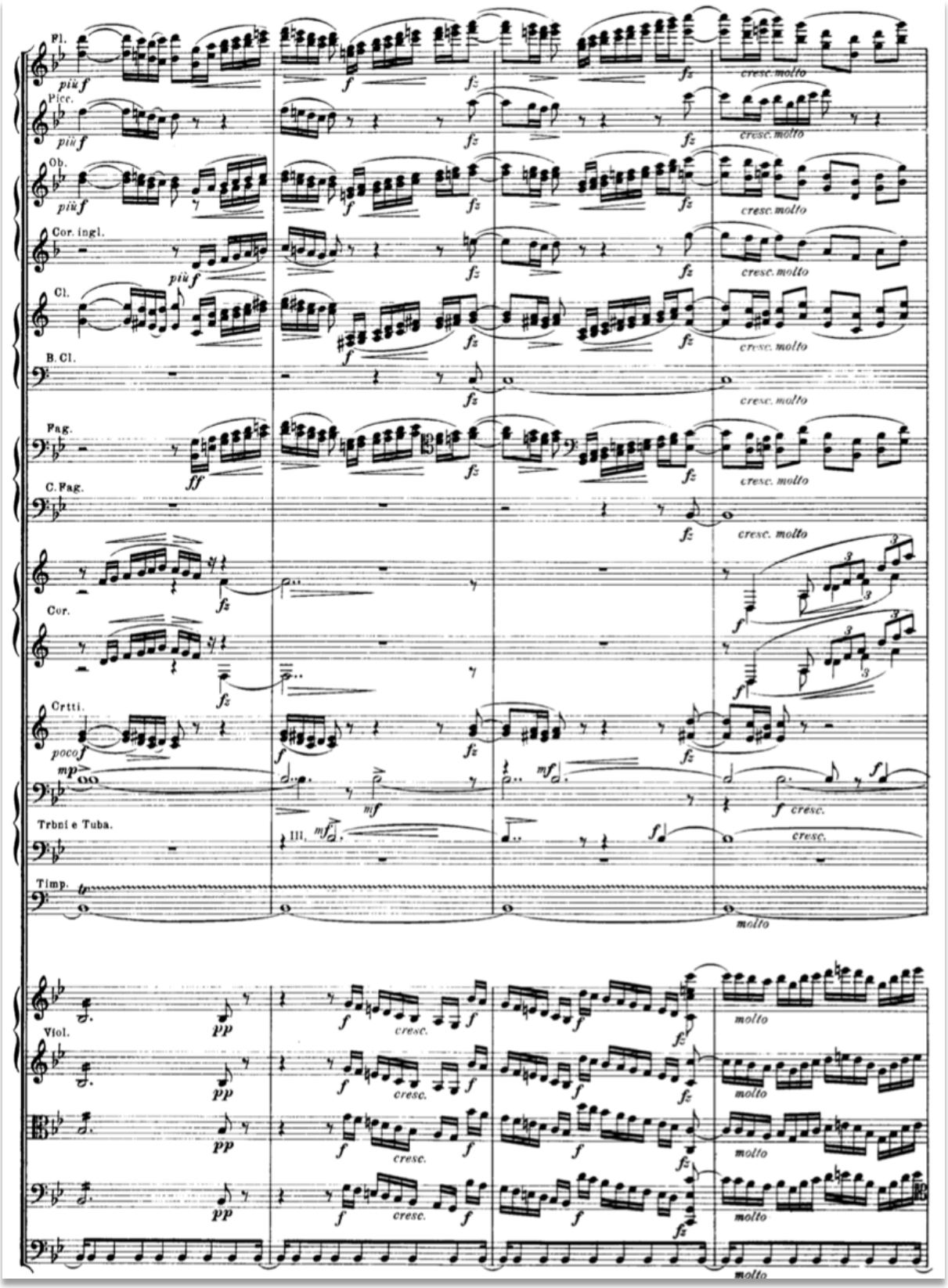


Figure 4-3 Test score for the Qualifying Stage: Phojola's Daughter, Op. 42 (Sibelius 1991, 196).

4.1.3 The Finals (Engraving Stage and Correction Stage)

The final two stages were administered two weeks apart and are referred to as the engraving stage and the correction stage (stage two). In the engraving stage, participants were asked to transcribe two pages of orchestral music form the classical era (Mozart): page A and page B, and the romantic era (Ravel): page C and page D. The task in the correction stage, was to correct the symbolic files (MusicXML) created in the engraving stage by others. Each individual page was presented to the participants using the same website as in the Qualifying stage with different UUIDs so the URLs could not be guessed ahead of the announcement. Both the encoding and the participation guidelines were the same as in the Qualifying stage. More details for both these tasks are explained below.

4.1.3.1 Engraving Stage

Participants were timed from when they downloaded an individual page to the moment they uploaded their engraving in MusicXML format. As each of the four human participants had two different pages to manually convert in notation software for the engraving stage, the participants were allowed to take a break of any length between engraving pages as long as both pages were completed by 19 November 2019 at noon. All participants were informed about the break in the participation email and a reminder was placed before starting the second page of the engraving stage.

A fifth participant (myself) representing the commercial OMR software was added to the list of participants in the engraving stage. The process of operating the OMR software including the scanning of the scores was timed and monitored by a PhD student and videotaped by another person. The timer began when I was physically handed a closed book containing the score with the page number and finished when a symbolic music file from the OMR software was saved to the desktop. I was unaware of the identity of the selected orchestral scores before the scanning. The same digitized images of the score were used for the participants in the engraving stage. The scanner used was an Epson Scanner 12000 XL connected to a Macmini7,1 on macOS 10.13.6.

In the engraving stage, a time limit was imposed on all the participants to finish each page within one hour and thirty minutes. Participants were warned that going over the time limit would disqualify the submitted file and that submitting an incomplete file is preferable. How the pages were distributed to the participants is shown in Table 4-1. The table shows that each of the four human participants engraved two pages and each page was engraved twice by two different

participants. All of the pages were scanned and engraved automatically by a commercial OMR software (Software A from Chapter 3).

Table 4-1 Distribution of pages in the Engraving stage of MEWC.

Engraving participants	*Engraving 1*	*Engraving 2*		
Participant 1	A	B		
Participant 2	B	A		
Participant 3	C	D		
Participant 4	D	C		
Participant 5 (OMR)	A	B	C	D

4.1.3.2 Correction Stage

In the correction stage, the process was similar as the process from the engraving stage (i.e., one webpage and a timer per file), and the differences are described here. The correction stage was different in that participants would download a zip file with both a MusicXML file and a PDF file of the original score and the participant needed to correct the MusicXML file in their preferred notation software. In this stage, each participant had six files to correct which were created by the participants in the previous stage. Six files meant six parts and five opportunities to take a break at any time between those parts (i.e., as in the engraving stage).

Participants were told that incomplete pages are acceptable, however, going above 30 minutes would be counted against them as the file would be disqualified just as in the engraving stage. The distribution of pages was different for this stage. If a participant had the classical-era pieces (page A and page B) in the engraving stage, then the participant would be presented with the romantic-era pieces (page C and page D) in the correction stage (and vice versa) as seen in Table 4-2. The pieces selected for the two final stages were two pages from Mozart's Magic Flute Overture and the orchestral versions of Ravel's Aldorado del Gracioso. The pages were selected by Kit Soden, a PhD student in composition at McGill University, and are reproduced here: Figure 4-4, Figure 4-5, Figure 4-6, and Figure 4-7, as Page A, B, C, and D, respectively..

Table 4-2 Distribution of pages to participants in the Correction stage.

"Pn" represents the participant from the previous stage who encoded the MusicXML. A, B, C, and D refer to the pages.

Correction stage participants	*1st*	*2nd*	*3rd*	*4th*	*5th*	*6th*
Participant 1	P5-C	P3-C	P4-C	P4-D	P3-D	P5-D
Participant 2	P4-D	P5-D	P3-D	P3-C	P5-C	P4-C
Participant 3	P1-A	P2-A	P5-A	P5-B	P2-B	P1-B
Participant 4	P2-B	P5-B	P1-B	P1-A	P5-A	P2-A

Each participant corrected three versions of the same score. The different versions were as follows, one OMR output and two other versions were taken from the engraving stage by two other participants who engraved the pieces in the other style. The time limit for the correction stage was thirty minutes with the same engraving and participation guidelines as the previous stages (i.e., qualifying and engraving stages). The participants received the pieces in random order through the website with the ability to take breaks at the discretion of the participants. For both the qualifying stage and the final round, the errors were counted manually after rendering the MusicXML symbolic files in both Sibelius and MuseScore notation editors.

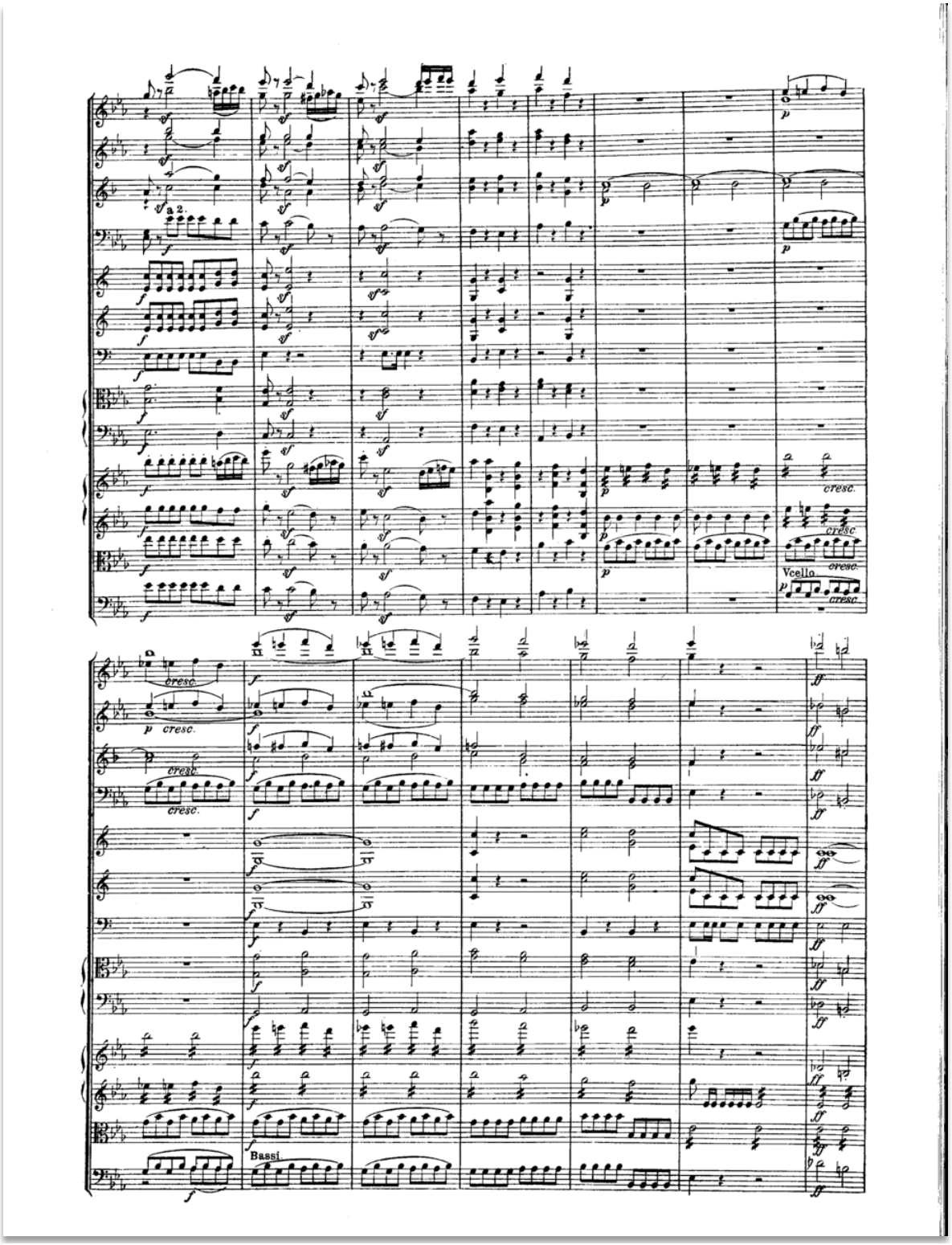


Figure 4-4 Test score for the Final: Overture, The Magic Flute (Mozart 1985, 15) (Page A).

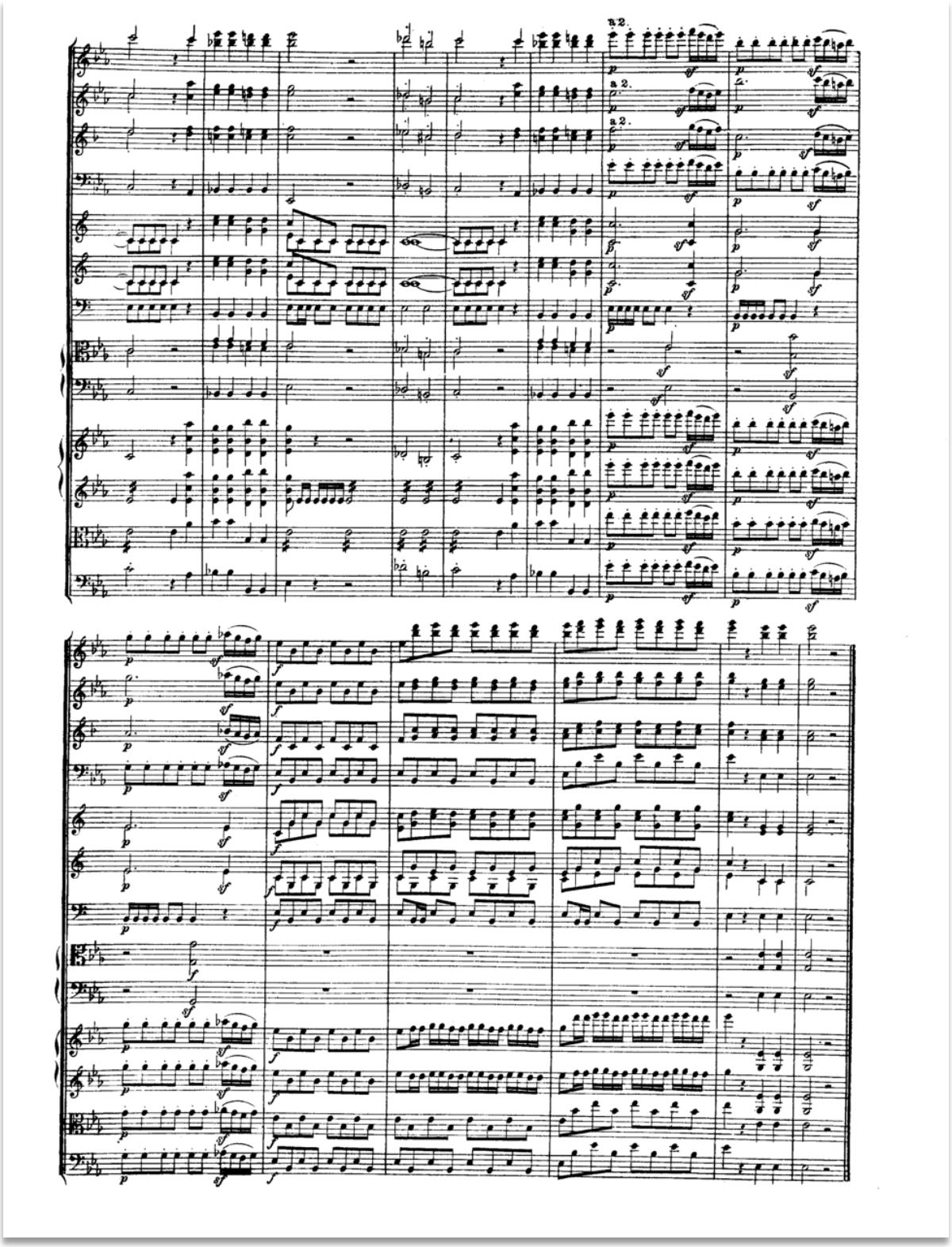


Figure 4-5 Test score for the Final: Overture, The Magic Flute (Mozart 1985, 16) (Page B).

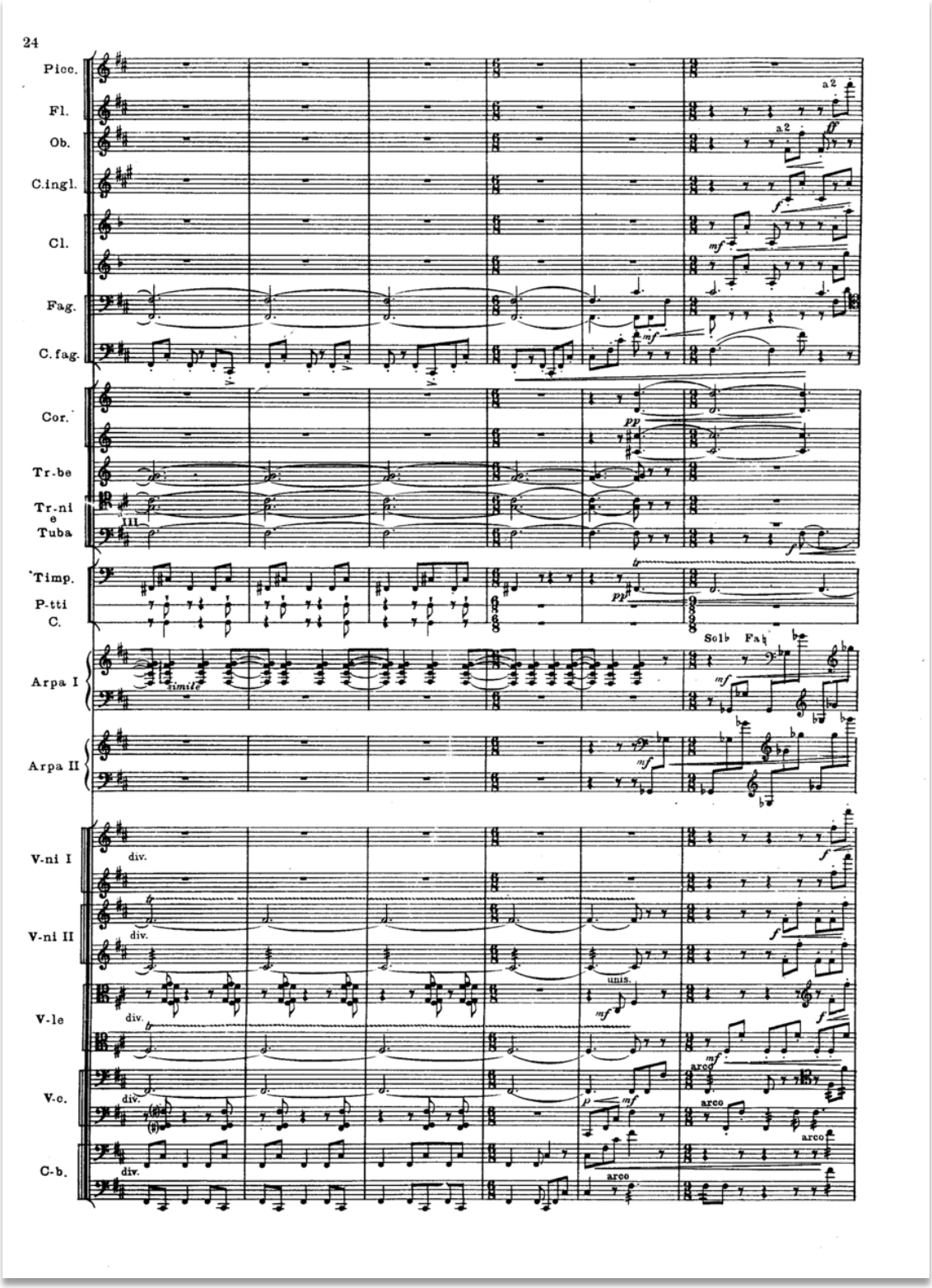


Figure 4-6 Test score for the Final: Aldorada del Gracioso, Miroirs (Ravel 1970, 24) (Page C).

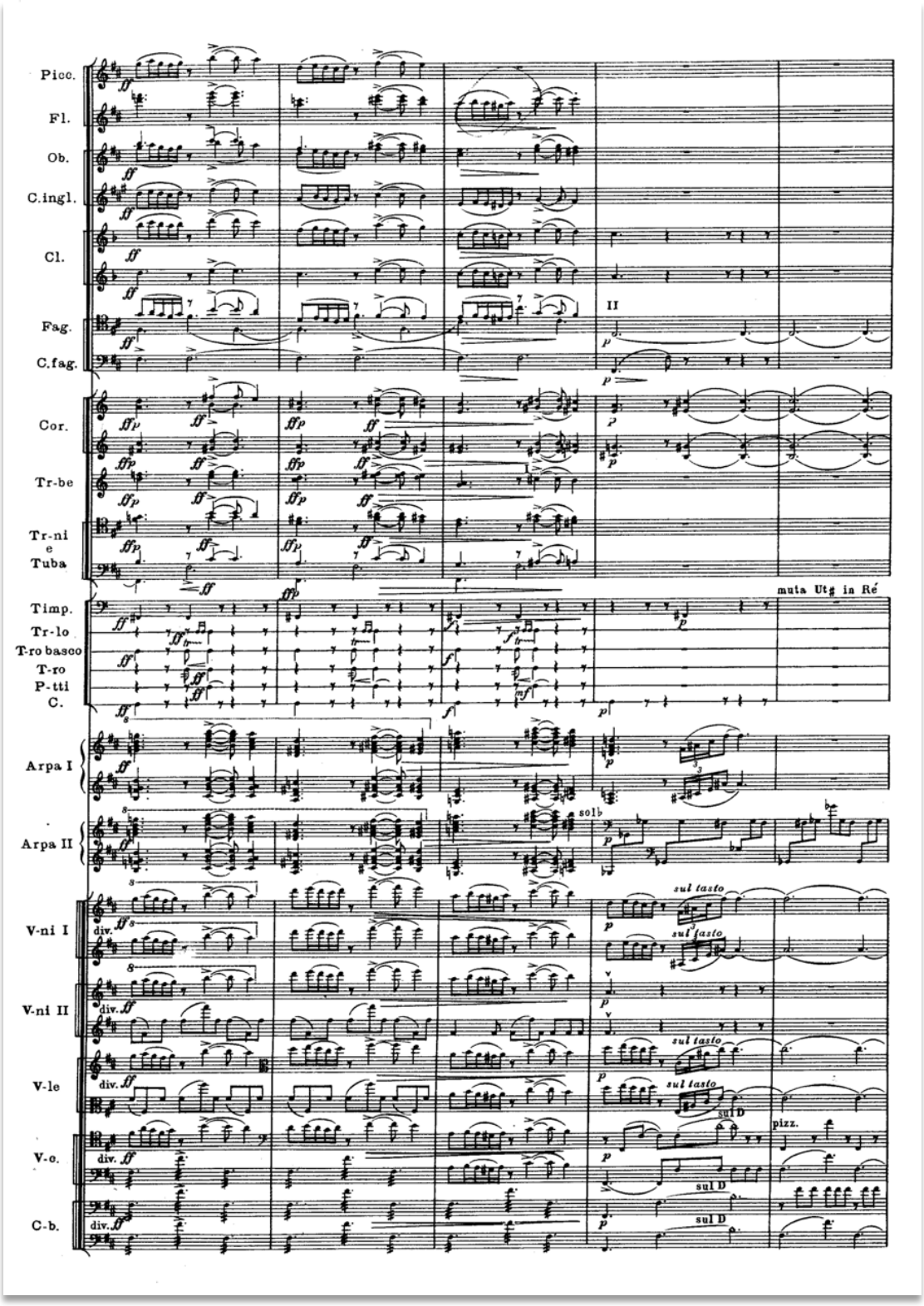


Figure 4-7 Test score for the Final: Aldorada del Gracioso, Miroirs (Ravel 1970, 25) (Page D).

4.2 Results of the Contest

4.2.1 Qualifying Stage results

As in all stages, the participants were asked to engrave as quickly and as accurately as possible. Although the qualifying stage did not have a time limit for the participants to complete the engraving, their timings were recorded. The time taken by the participants is shown in Table 4-3, along with the number of correct notes the participant engraved in that time. The number of correct notes were counted manually. The four fastest participants finished the page within an hour and thirty minutes. Some participants did submit partially completed scores around the one-hour mark (i.e., Participant one and eight).

Table 4-3 Qualifying stage participant performance.

The "Total Time" is in the hours:minutes:seconds format, and it is converted to seconds in the "Time in Seconds" column. The score takes the number of "Correct Notes" and divides this number by the number in the corresponding "Time in Seconds" column. Finally, a rank is given to each participant.

	Total Time	*Time in Seconds*	*Correct Notes*	*Score as Notes per Second (Higher is better)*	*Rank*
Participant 1	1:02:00	3720	526	0.141	4
Participant 2	0:54:33	3273	750	0.229	1
Participant 3	1:35:52	5752	747	0.130	5
Participant 4	2:17:09	8229	643	0.078	7
Participant 5	1:40:28	6028	659	0.109	6
Participant 6	1:21:55	4915	736	0.150	3
Participant 7	1:19:52	4792	739	0.154	2
Participant 8	1:04:00	3840	195	0.051	8

To choose the top four finalists, a score based on the number of correct notes engraved per second was used (Notes per Second). The four best participants, based on Notes per Second, were Participants 1, 2, 6, and 7, who went on to compete in the two-stage final round: the Engraving stage and the Correction Stage.

4.2.2 Engraving Stage results

Table 4-4 shows the time it took the participants to engrave each of the two pages they were assigned. The Mozart and Ravel pieces are not equivalent with respect to manual engraving. The total number of symbols differs in each of the four pages. The "Score as Notes per Second" was used to quantify the performances of the participants to find a winner. The average times to engrave each of the four pages by the two participants who engraved them are also shown. The last row ("OMR") shows the total time it took to process each of the pages using the commercial OMR software.

Table 4-4 The engrave times the four test pages by the four participants and the OMR software.

The time is in hours:minutes:seconds format. The time to engrave OMR includes digitizing the score using a scanner and troubleshooting when the commercial OMR failed to recognize a page which did occur. The timer stopped when a MusicXML file was saved to the desktop. Oowashi from Chapter 5 was not used to automate this process.

	Time to Engrave Page A	*Time to Engrave Page B*	*Time to Engrave Page C*	*Time to Engrave Page D*
Participant 1	1:24:01	1:27:00		
Participant 2			1:10:06	0:34:56
Participant 6			1:24:42	1:26:30
Participant 7	1:11:07	0:51:18		
Average time	**1:17:34**	**1:09:09**	**1:17:24**	**1:00:43**
OMR	0:03:04	0:06:16	0:11:40	0:06:17

4.2.3 Correction Stage results

In order to determine whether the use of OMR is more efficient than purely manual engraving process, the correction time will be added to the engraving time. At the beginning of this stage, each of the four test pages (Pages A, B, C, and D) had been encoded into three MusicXML files: two files by two different human engravers (Engravings 1 and 2) and another file by the OMR software. Each of these files were corrected by two participants (Corrector 1 and 2).

The elapsed times taken to correct two engraved versions of the four pages are shown in Table 4-5. The elapsed times taken to correct the output of the OMR software are shown in Table 4-6. As expected, the correction times for the OMR output were longer than the correction times of the human engraved music scores.

To properly compare the human-only process and the OMR-aided process for transcriptions of orchestral scores, additional time was added to manually correct the last remaining errors in the MusicXML files. The total timings for the creation of finished transcriptions are shown in Tables 4-7 and 4-8 for the human-only process and the OMR-aided process, respectively. The results are also show as a graph in Figure 4-8. The results clearly show that the OMR-aided process is consistently faster than the human-only process.

Table 4-5 Human correction time.

Humans	*Corrector 1: Engraving 1*	*Corrector 2: Engraving 1*	*Corrector 1: Engraving 2*	*Corrector 2: Engraving 2*	***Average Correction Time***
Page A	0:22:20	0:10:04	0:26:45	0:20:18	**0:19.52**
Page B	0:29:12	0:29:54	0:23:34	0:11:13	**0:23:28**
Page C	0:26:28	0:07:09	0:29:46	0:10:18	**0:18:10**
Page D	0:29:29	0:12:37	0:29:22	0:16:51	**0:22:05**

Table 4-6 OMR correction time.

OMR	*Corrector 1*	*Corrector 2*	***Average Correction Time***
Page A	0:27:18	0:28:35	**0:31:00**
Page B	0:29:26	0:28:46	**0:35:22**
Page C	0:15:58	0:29:58	**0:34:38**
Page D	0:30:37	0:29:19	**0:36:15**

Table 4-7 Human finishing time including additional correction time.

The average Engraving times are taken from Table 4-4 and the average Correction times are taken from Table 4-6.

Humans	*Engraving*	*Correction*	*Additional*	***Total***
Page A	1:17:34	0:19:52	0:01:21	**1:38:47**
Page B	1:09:09	0:23:28	0:02:51	**1:35:28**
Page C	1:17:24	0:18:10	0:04:51	**1:40:25**
Page D	1:00:43	0:22:05	0:12:42	**1:35:30**

Table 4-8 OMR finishing time including additional correction time.

The average Correction times are taken from Table 4-5.

OMR	*Recognition*	*Correction*	*Additional*	***Total***
Page A	0:03:04	0:31:00	0:14:52	**0:45:52**
Page B	0:06:16	0:35:22	0:15:10	**0:50:32**
Page C	0:11:40	0:34:38	0:25:50	**1:00:28**
Page D	0:06:17	0:36:15	0:40:00	**1:16:15**

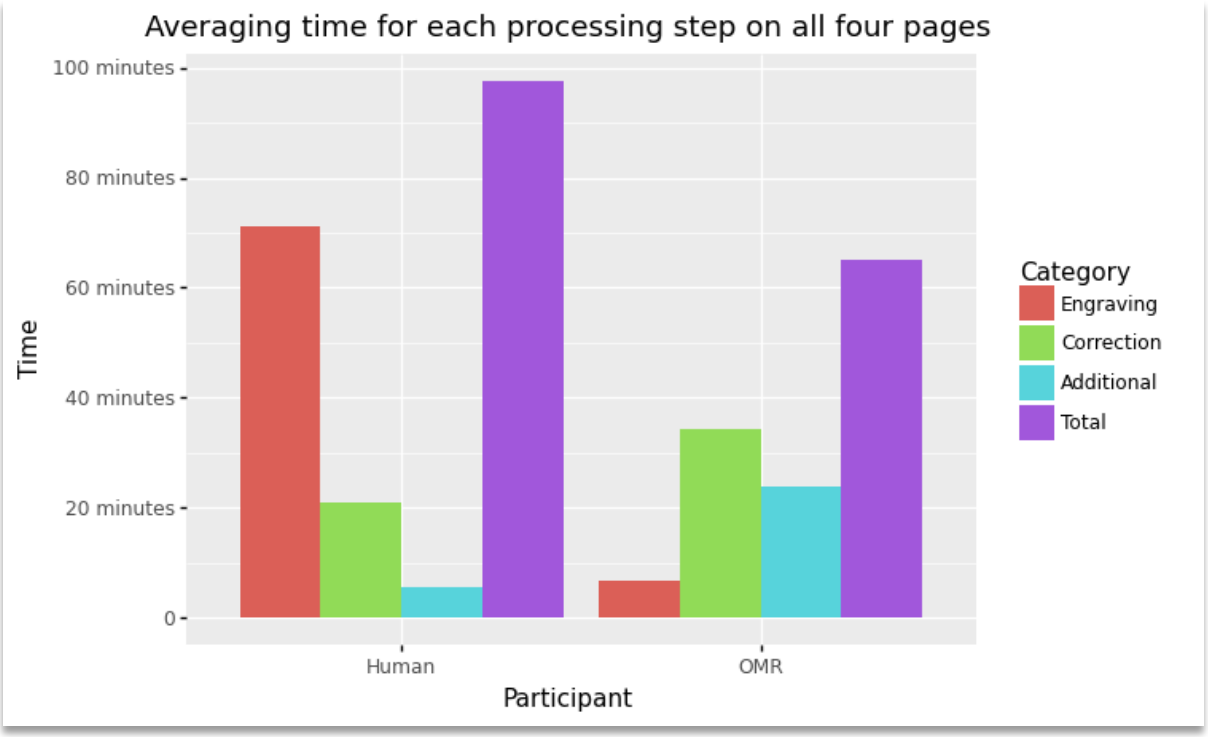


Figure 4-8 Comparison between Human and OMR times in MEWC.

Chapter 5 The Oowashi Project

The goal of the Oowashi[34] project is to automate the task of converting images of scores to symbolic music files on a large-scale. Oowashi will take a database of scores and systematically create symbolic files for each page in each of the scores within the database. The database of scores used was the International Music Sheet Library Project (IMSLP) database (Guo 2014). A large section of the IMSLP database has already been transcribed automatically by Viro (2011). Oowashi was inspired by Viro (2011), and the software libraries used in Hankinson (2014) influenced the libraries used in the construction of Oowashi. Both Oowashi software and the symbolic files created with it will be publicly accessible. In the next section, a background on the IMSLP database is presented.

5.1 Background on the IMSLP Database

The International Music Sheet Library Project (IMSLP) is a website that stores digital images of musical scores uploaded by users. The website began "in February 2006 as a project to make public domain scores freely accessible on the internet." (Guo 2014, 267). Edward W. Guo is the IMSLP project creator and lead programmer. Guo (2014) had always maintained two firm positions in regard to all entries on the IMSLP server. First, all files in IMSLP must always be stored on IMSLP servers as opposed to links. Second, all entries in IMSLP, if they are digital scores, must be in PDF format. Any users of IMSLP are allowed to upload digitized scores to the database and guidelines on how to upload material are available on the IMSLP website.[35] They are not made publicly available, however, until moderators have verified copyright information. Special users with moderation privileges manually review new uploads before making them available on the website to avoid copyright infringements. This is because IMSLP has had legal cease and desist letters sent to them in the past.[36] All digitizes scores available on IMSLP must

[34] Oowashi is named after a Kaiju from the Godzilla franchise and is also known as Ookonduru https://web.archive.org/web/20200712191148/https://godzilla.fandom.com/wiki/Giant_Condor
[35] https://imslp.org/wiki/IMSLP:Score_submission_guide
[36] https://web.archive.org/web/20071023090807/http://imslpforums.org/Second%20U-E%20Cease%20and%20Desist%20Letter.pdf

be out of copyright to be viewable in the country where a user is accessing the website from to conform to local copyright laws.

Users of IMSLP include representatives from University libraries as they are digitizing their scores and uploading them to IMSLP. One such collaboration is the Sibley Mirroring Project from the University of Rochester.[37] Many scores have been digitized and users of IMSLP may manually add metadata needed to include the score in the IMSLP database.[38] Other users of IMSLP are composers who want to upload their own compositions.

5.2 Overview of Oowashi

Oowashi is divided into two components, the Command and Control Server (Controller) and the Workers. The Controller controls and supervises a swarm of Workers that run on individual computers. Each Worker is contained inside a virtual machine so that several Workers can coexist independently on a host computer. A diagram describing this interaction between Controller and Workers is shown in Figure 5-1.

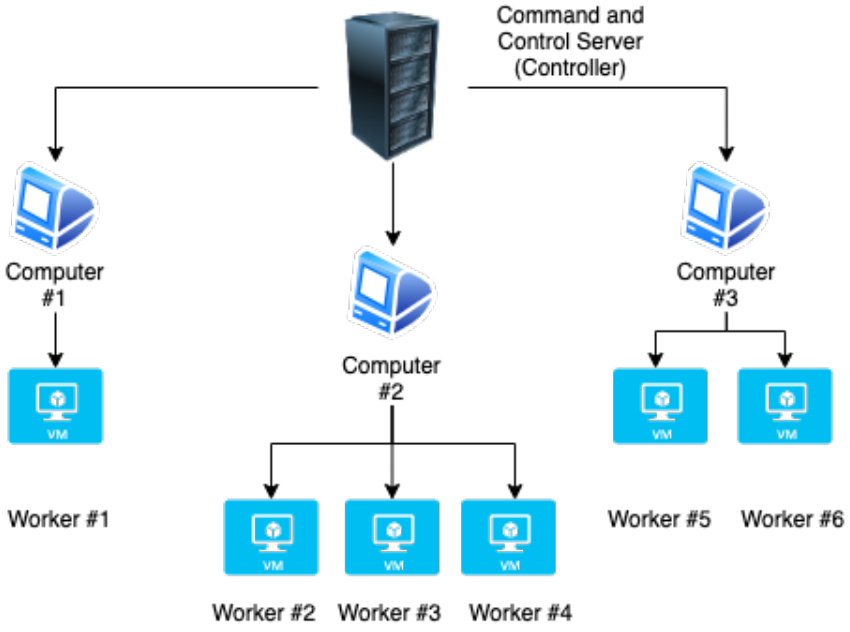


Figure 5-1 A representation of an Oowashi project.

[37] https://imslp.org/wiki/IMSLP:Sibley_Mirroring_Project

[38] https://web.archive.org/web/20200815163406/https://imslp.org/wiki/IMSLP:Sibley_Mirroring_Project:Walkthrough

The Controller is a web application running on a server (a website), which also stores the resulting files that the Workers uploads. The input files for the OMR process is stored on a separate file server. Multiple Oowashi Workers, typically running on desktop computers, execute the tasks given to them by the Controller. A task for a Worker is to download a file, process the file using an OMR software running on the local computer, then uploading the resulting files to the Controller.

The Controller maintains a list of tasks. It tracks how many tasks have been completed by the Workers and how many tasks remain to be processed. A Worker requests the Controller for a new task when the Worker finishes a previous task. A task is marked as completed when a Worker uploads the finished files to the Controller. The Controller also monitors the health of its Workers and can remotely restart a Worker if it takes too long to process a task. The specific functions of the Controller and the Workers will be described in the next two sections.

5.3 Description of Oowashi Command and Control Server

The Oowashi Command-and-Control Server (Controller) commands and controls multiple Workers.[39] A Controller is deployed on a server enabling all Workers to communicate with it. The Controller features tools and scripts that control Workers.

The Controller can start and stop, any Worker and once a Worker starts, it will request the Controller for a task. In response, the Controller sends the location (URL) of one of the unprocessed PDF files from the server hosting the IMSLP files to be processed by the Worker. When the Worker finishes processing the file, by performing OMR, and sends back the resultant files, the Controller will acknowledge the receipt of the files and flags the PDF file as processed. The Controller will then wait for the next request from a Worker.

The Controller also keeps track of the time since a Worker has last requested a new file, recording the elapsed time for a specific task. This allows the Controller to restart a Worker if a task is taking too long to complete—a sign that the Worker has stopped working.

The Controller is also responsible for making backups of processed data, which it does twice a day to one of two off-site locations. In case of a power outage of the Controller, it can

[39] A Command-and-Control server (i.e., C2, or C&C) is a collective term used to refer to a computer that controls a network of computers when talking about malware and botnets. With good or bad intentions, a computer that distributes a large-scale process over multiple computers to complete the task faster is the same as a C2.

automatically restore files from the backups. The Controller can also automatically update the Worker's software by remotely restarting a Worker.

5.4 Description of Oowashi Worker

The main job of Oowashi Worker (or Worker) is to run the OMR software. Given a PDF file fetched from the Controller, it automatically processes the file, unattended, to produce MusicXML files, and sends the recognized files to the Controller for long-term storage.

All automation for OMR workflows follows the same structure. The communication sequence for downloading a file is in Figure 5-2, while the flowchart of the automation is shown Figure 5-3. The automation proceeds as follows: The Worker makes a request to the Controller for a file to process, it downloads a PDF, which may contain several pages, converts each page into a single page TIFF file, and runs the commercial OMR on each TIFF files to produce a MusicXML file (i.e., one MusicXML file per TIFF file). After completion of the conversion, it creates a Zip archive file that contains all TIFF and MusicXML files, uploads the Zip file to the Controller, and deletes the Zip file upon successful upload to the Controller. It then starts the next iteration of the automation loop by making a request to the Controller for another PDF file to process.

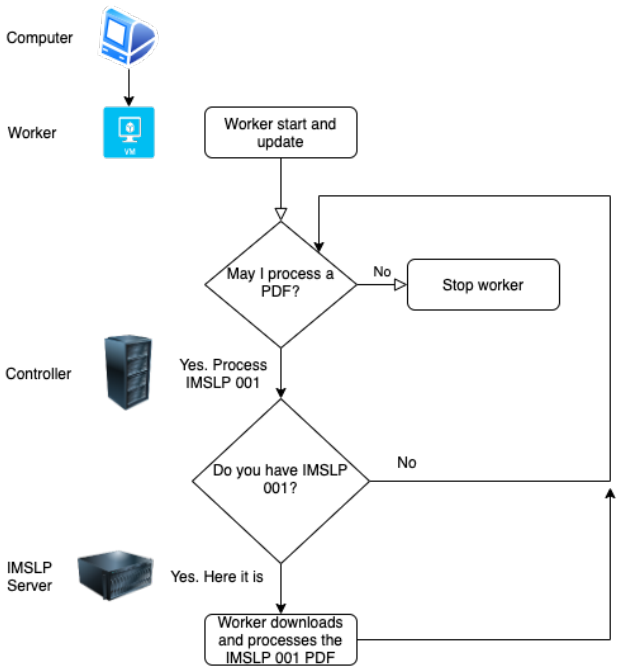


Figure 5-2 An example Worker communication with the Controller and the IMSLP server.

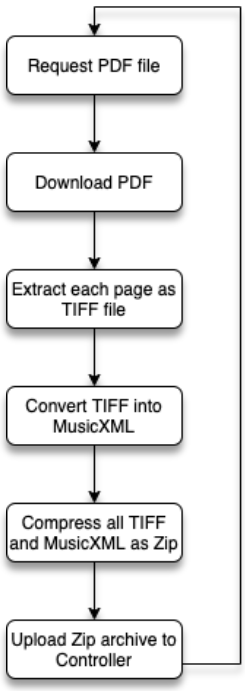


Figure 5-3 Common Worker process for all Oowashi Workers.

The Worker is built on a collection of scripts that automate commercial OMR software applications, which are all based on interacting with the Graphic User Interfaces (GUI). The scripts, therefore, are designed to control the GUI without human intervention in order to process thousands of score images automatically. During the conversion process (OMR), a Worker controls the commercial software by triggering a series of preprogrammed sequences of mouse and keyboard actions (e.g., mouse clicks, mouse movements, keyboard typing, keyboard combinations, mouse dragging, mouse dropping).

The following is a simple explanation of how the GUI automation works: At different stages in the process, the Worker programmatically takes a screenshot of what is visible on a computer screen and searches for the coordinates of a template image (e.g., of a button in the OMR software), which has been previously provided, in that screenshot. The Worker then moves the mouse to the coordinates of the image found in the screenshot and interacts with the object found at that location (e.g., by clicking, dragging, or typing). When that action is completed, the Worker then looks for the next template image in a new screenshot.

In the preprocessing stage, the individual TIFF files to be recognized are placed in the "to_scan" folder. After recognizing a TIFF file from the "to_scan" folder, the file is moved by

the Worker into the “from_scan" folder along with the resulting OMR output, which is in the MusicXML file format. Once the “to_scan” folder is empty, the Worker creates a Zip archive file of all the files in the “from_scan” folder, and upload the archive to the Controller. This completes the task and the whole process is started again by the Worker requesting the next PDF file from the Controller.

Workers are also designed to be resilient to unexpected scenarios, such as power outages. A Worker is programmed to automatically start itself with the startup routine of the operating system. When the operating system is up and running, the Worker will attempt to update its own software code before doing anything else. It will then check to see if there are any local files before it went down. If there are no local files, the Worker will request a new PDF file from the Controller—meaning that it had completed the previous task—or, if there are some local files to be processed, such as the PDF file or the Zip file, it will resume from the processing step before the computer was restarted.

5.5 The Oowashi Experiment

5.5.1 Experiment Setup

The goal of the experiment was to convert as many music scores image files available at IMSLP into MusicXML files. This was accomplished using the Oowashi framework, which was explained in previous sections of this Chapter. Several computers were used for this experiment. The Oowashi Controller was deployed to a cloud-based virtual machine, running an Intel Xeon Skylake processor at Computer Canada. Four virtual cores were allocated to the Oowashi Controller using a virtual machine, which was allotted 5 Terabytes of storage and 6 Gigabytes of RAM. The Workers were distributed among computers in the Distributed Digital Music Archives and Libraries Laboratory and the Music Perception and Cognition Laboratory at McGill University. The machines currently in use are listed in Table 5-1. Note that each virtual machine (VM) runs a single Oowashi Worker. Each VM uses 2 Gigabytes of RAM and 27 Gigabytes of storage. In addition, another VM was deployed at Compute Canada which mirrors the IMSLP database. The VM was allocated one compute core (also an Intel Xeon Skylake processor) with 1.5 Gigabytes of RAM and 4 Terabytes of storage.

Table 5-1 Oowashi Worker instances with hardware specifications.

	Workers	*Operating System*	*Hardware Revision*	*CPU*
Computer 1	3	Mac OS X 10.15.4	Macmini8,1	6-Core Intel Core i5
Computer 2	2	Mac OS X 10.13.6	Macmini6,2	Intel Core i7
Computer 3	3	Mac OS X 10.14.6	Macmini8,1	6-Core Intel Core i5
Computer 4	3	Mac OS X 10.14.6	Macmini8,1	6-Core Intel Core i5
Computer 5	3	Mac OS X 10.14.6	Macmini8,1	6-Core Intel Core i5
Computer 6	3	Mac OS X 10.11.6	MacPro4,1	Quad-Core Intel Xeon
Computer 7	2	Mac OS X 10.11.6	Macmini6,2	Intel Core i7
Total # of Workers	19			

5.5.2 Preparation of IMSLP Mirror File Server

The IMSLP mirror file server was used to locally replicate all the files from the main IMSLP website (https://imslp.org). It took over a week to download the 2.5 Terabytes of data from the main website to the local mirror file server and files are constantly being downloaded as more files are being added to the main website. As of writing on 23 August 2020, there are 10,240,659 pages in 490,373 PDF files in the IMSLP mirror. This means the average number of pages in a PDF is 20.88 pages. The largest PDF in IMSLP contains 5,356 pages.

5.5.3 Preparation of the Controller Database

The server that runs the Controller also stores all of the output files (TIFF files and MusicXML files) created by the Workers. In order to keep track of which PDF files were used to generate the output files, the folder structure in the Controller is made identical to the folder structure in the IMSLP file server. To create the database for the Controller, a script was written to walk the directory structure of the IMSLP file server and populate the Controller's database. The database keeps track of which files are unprocessed, in process, or finished processing.

5.6 Experimental Results

This section is a report on the current progress of the large-scale recognition project named Oowashi. Currently, the IMSLP database is being converted from digitized images into MusicXML files and the following are some statistics.

The earliest version of the Controller and the Workers were put online on 3 April 2020 to begin processing the PDF files from IMSLP marking the official start date of the large-scale recognition experiment.

As of 23 August 2020, 40,090 PDF files have been processed. The progress of the experiment is plotted in Figure 5-4. The sudden increase in the number of Zip files on 4 May was caused by an unexpected bug in the OMR software, causing a few Workers to produce over 5,000 invalid Zip files in a day. A fix to deal with this bug was installed in the Worker software the same day. The flat line at the end of June was caused by a stoppage of all systems due to a power outage and because of the Covid-19 pandemic, during which the lab was inaccessible for a few days. After this incident, all the Mac desktop computers were configured so that even in the event of power failures, the machines will start up automatically. In the graph, there are three different slopes (the rate of processing): Period 1: 3 April to 4 May, Period 2: 5 May to 19 June, and Period 3: 23 June to 23 August. These reflect software updates and the number of Workers involved, which changed slightly over time. From Period 1 to Period 2, the Worker software was updated to be more efficient along with the bug fix. From Period 2 to Period 3, the number of Workers was reduced from 25 to the current 19, due to two computers becoming unavailable to be used for this experiment.

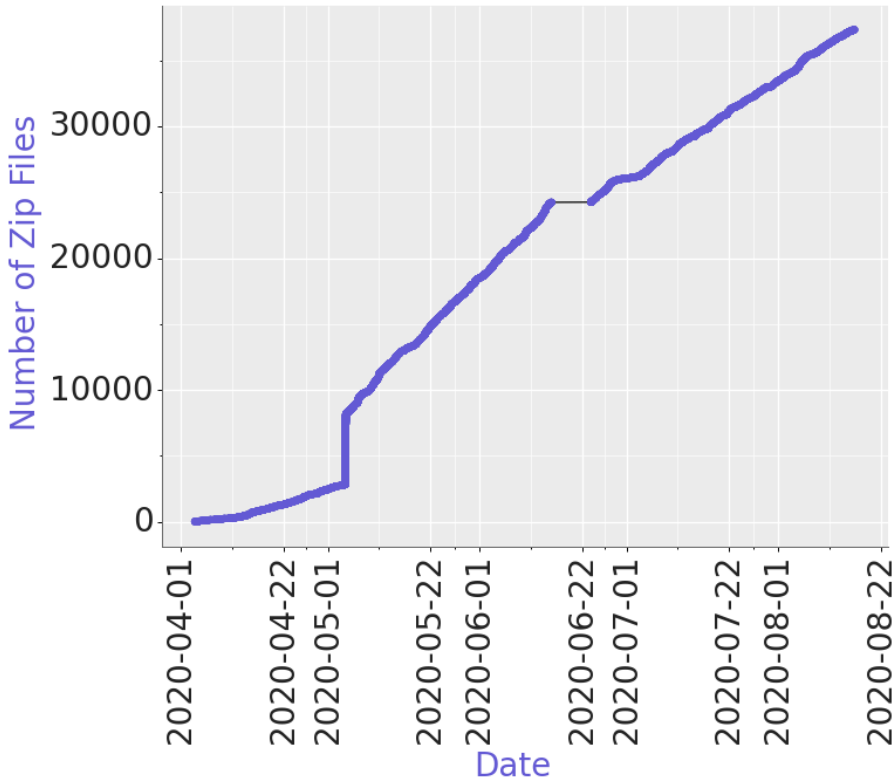


Figure 5-4 Growth of the Controller's database.

5.7 Discussion of the Results

Of the total of 40,090 Zip files uploaded by the Workers, 5,192 files were basically empty caused by the incident on 4 May. By unzipping the remaining 34,898 Zip files, we find that the Workers have created 534,226 TIFF files and 438,276 MusicXML files. This means that the Worker generated about 3,065 MusicXML files (each representing a page of music) per day in the 143 days. Therefore, the rate at which the OMR software successfully produced a MusicXML file is 82%.

The main cause of missing MusicXML files is likely due to the fact that the OMR software is designed to recognize printed Common Western Music Notation (CWMN) but IMSLP contains a significant number of music in non-CWMN, and some handwritten music scores can have unexpected results.

Because there are currently about 500,000 PDF files on the main IMSLP website, at the current processing rate of 244 (34,898/143) PDF files per day, the project should finish in about five years (465000/244/365 = 5.22) using the current number of computers. This estimate also

ignores improving the efficiency of Workers. But the IMSLP is still growing at the rate of about 100 PDF files per day (see Figure 5-5), so it would take a few more years beyond the five years to complete.

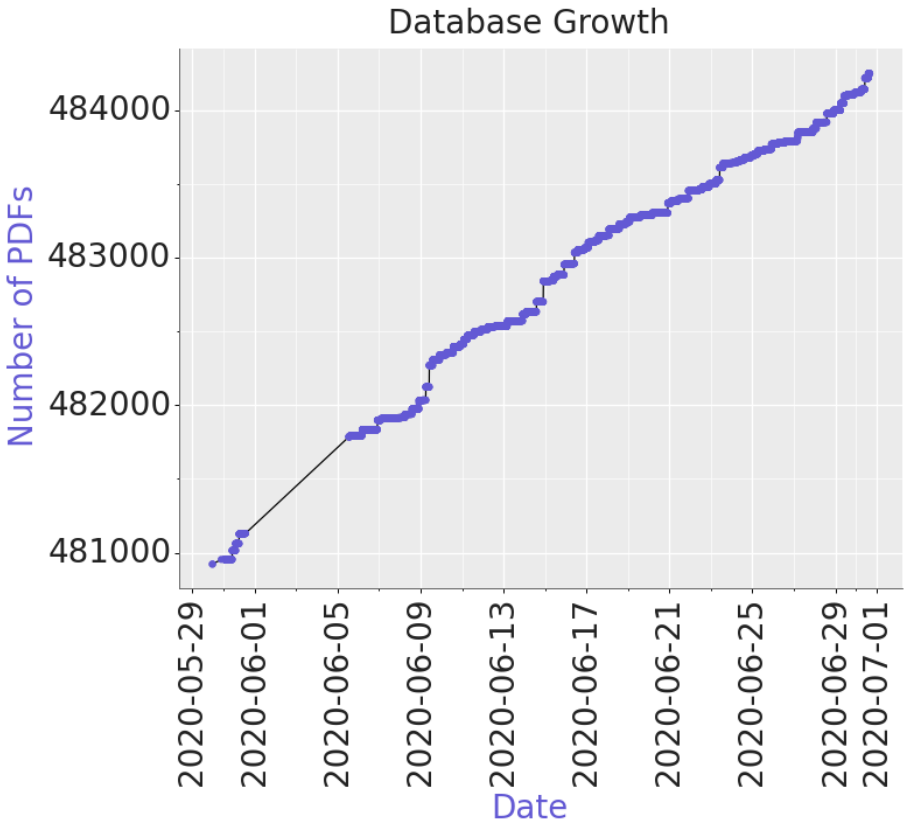


Figure 5-5 IMSLP database growth over June 2020.

5.8 Conclusions and Future Works

The Oowashi Project is running successfully and there are plans to expand on its efficiency, public visibility, and support for other operating systems. Additional scripts will be written to automate more OMR software, along with efficiency updates to the existing scripts.

Currently, the Oowashi Worker can only execute the GUI automation on the macOS operating system. The plan is for a Worker to be able to automate tasks on the Windows operating system using the same calls to the Oowashi library for the same actions as in the

macOS. This is desirable because some commercial OMR applications are only available on the Windows operating system (e.g., ScoreMaker and SharpEye).

There are several potential avenues of research using the results of the Oowashi Project. When the IMSLP dataset is processed by several different OMR software, the use of Multiple-Recognizer OMR (MROMR), devised by Byrd and Schindele (2006) and others, may be revisited to ameliorate the OMR output. Also, because IMSLP often contains different editions of the same music, the strategy used by Ng, McLean, Marsden (2014) and Ringwalt and Dannenberg (2015) can be further investigated to minimize the errors. Finally, it is hoped that a crowed-based platform can be developed to allow users to submit corrections to these files to improve the quality of results of the Oowashi Project.

Chapter 6 Conclusions

This chapter summarizes the results of the three investigations in this thesis, which attempted to evaluate various aspects of Optical Music Recognition (OMR) processes. In Chapter 3, a software application, called Mupix, was created that automatically enumerate the differences between two or more MusicXML files. The difficulties in comparing MusicXML files were surveyed in Chapter 2.

By using a music analysis software library, Music21, Mupix first converts different musical elements found in a music symbolic files, specifically MusicXML files, into associative arrays of musical objects. These arrays represent a sequence of notes, rests, key signatures, etc. In order to compare two MusicXML files, these arrays are aligned using the Needleman-Wunsch dynamic programming algorithm. Once aligned, the software counts the differences between the arrays, revealing differences between MusicXML files.

As one application of the Mupix software, three different commercial OMR software results were compared against a set of ground truth MusicXML files. The best performing software from this experiment was selected to be used for the two remaining experiments of this thesis.

The Music Engraving "World Cup" (MEWC), detailed in Chapter 4, was a contest between humans and computers. The task being the creation of symbolic music (MusicXML) files. The combination of the processing time and the correction time was used as a metric to select the winner. After a qualifying stage, four top-performing human music engravers were chosen to compete again a commercial OMR software. Four pages of relatively complex orchestral scores were used as the test data. The humans used a notation editor of their choice to manually engrave scores. Also, the four pages of scores were scanned and processed by a computer. The resulting MusicXML files were corrected for errors by two human correctors. The time it took to correct the errors were added to the engraving time by humans or the recognition time by the computer and tallied. The winner was the computer. Thus, engraving orchestral music appears to be more cost-effective when OMR software is used.

A project of automatic creation of an extensive database of publicly available symbolic music files was presented in Chapter 5. To create this database, thousands of PDF files from the IMSLP database is being processed by a commercial OMR software. In this last experiment, a

special set of software tools were developed to manage a swarm of computers to process several music scores files simultaneously increasing the speed at which these files were converted from the PDF files to TIFF image files and processed by the OMR software to generate MusicXML files, which were then uploaded to a server for public access; all automatically and unattended.

6.1 Future Work

The tools developed in this thesis can be improved in the future. The Mupix software should be further tested using a wide variety of musical scores generated by different OMR software. Also, more sequence alignment algorithms should be implemented and tested.

The participation website created for the MEWC experiment could be improved by automatically parsing the log files to generate statistics such as the engraving time and correction times. If this contest were to be repeated, the most significant improvement, however, would be to have a program such as Mupix perform the error analysis as manually counting errors is time-intensive and prone to mistakes.

As mentioned in Section 5.8, there are several opportunities for improvements and potential applications for the Oowashi project. It is hoped that the database created by the Oowashi project, along with the tools developed in this thesis can motivate and enhance various new research opportunities in our study of music.

Appendix A

Symbolic Music File Formats

DARMS (Digital Alternative Representation of Music Scores)[40]

This file format was first under development in 1961 by Stefan Bauer-Mengelberg and Melvin Ferentz. The project grew to include Raymond F. Erickson, David M. Gomberg, and Anthony B. Wolff in different capacities as it quickly became apparent that creating a syntax to represent music notation was not simple. More information about the specific roles of each person involved is available in Erickson (1975).

DARMS is also known as the Ford-Columbia file format. The DARMS language tried to include all musical objects. The language would not encode musical objects as musical terms because the authors believed that the interpretation should be left to the human interpreter and not the encoding language. Erickson (1975) provides an example: rather than interpreting a key signature with a single sharp as G major or E minor, the language should instead encode a sharp on a specific line with neither assumption nor presumption of the key or tonality. Grande and Belkin (1996) underscored a weakness with DARMS, that it "is incapable of representing graphical information with precision," when they began working on a new file format which will be discussed later.

Resource Interchange File Format (RIFF)

Resource Interchange File Format, or RIFF, is a file format that can enclose other files. RIFF provides a specification for writing "future-proof" file formats with metadata. Microsoft, in collaboration with IBM, released the first version of the RIFF specification in August 1991.[41] Since its creation, the RIFF file format has informed many digital-multimedia file formats such

[40] Although many people believe DARMS stands for "Digital Alternative Representation of Music Scores", Erickson (1975, 291) says that DARMS was also named "in honor of a benefactor of the project" as it was when it was named the Ford-Columbia format.

[41] https://web.archive.org/web/20060604055015/http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf

as: video (e.g., AVI), audio (e.g., WAVE), images (e.g., DIB), and symbolic music (e.g., RIFF MIDI, which are MIDI files enclosed in a RIFF container) among others.

Notation Interchange File Format (NIFF)

The goal of the NIFF file format was for all software developers using Common Western Music Notation (CWMN) to support one common file format. This idea was not new because other file formats tried the same (e.g., Standard Music Description Language or SMDL). According to Grande and Belkin (1996), how NIFF differs from other formats is its inclusion of graphical information. The vendors targeted would include developers of notation software, OMR software, Digital Audio Workstations (DAWs), and software for musicological research.

The growth of the internet was a contributing factor to convincing commercial software vendors to support a single format (Grande and Belkin, 1996). The companies involved in creating the NIFF standard included: Passport Designs (Encore notation software), Coda Music Technology (now MakeMusic Inc. and makers of Finale), San Andreas Press (Score notation software), Musitek (SmartScore OMR), Cindy Grande (NoteScan OMR), Nicholas Carter (SightReader OMR). Code Music Technology then withdrew from the NIFF project to work on their own standard format (Enigma Transportable Format, or ETF). After the absence of Coda, other companies (Mark of the Unicorn (MOTU), Opcode Systems (no longer a company), Twelve Tone Systems (now Cakewalk, inc.)) among others joined the NIFF effort. The NIFF specification followed features from RIFF that accommodate future extensions to the format. This means a file that was created using a newer version of the NIFF specification would still work in software that was written with an older version of the NIFF specification because the new features could simply be ignored.

Enigma Transportable Format (ETF)

This file format was created by Coda Music Technology (now MakeMusic, inc.) with the goal of being the *de-facto* standard file format for transporting CWMN information between music software. A document that explains an early version of the ETF file format is available,[42]

[42] https://web.archive.org/web/20010614204853/http://www.cs.ruu.nl/~hanwen/lily-devel/etfspec.pdf

and more information is contained in the Finale 2000 plug-in development kit.[43] The format was not widely accepted, and was gradually phased out in favor of MusicXML.[44]

MuseData

This file format is made to contain notation information and sound information to then create either a SCORE file or a MIDI file (Selfridge-Field et al. 2001). This was accomplished by including a specific data type in their format called a "suggestion" for printing and sound reproductions that can be ignored when in a specific use-case.

The construction of a MuseData file is a set of variable-length records. Each record can store different information such as: the work, the encoder of the work, the source of the work, etc. Many individual MuseData files (representing each part of a full score) are linked together in one MuseData file that represents a full score (with links to each part).[45] [46]

Standard Music Description Language (SMDL)

In 1991, SMDL was another language with a design goal of being a file format which could be shared between commercial and open-source applications. This format stores "visual" (score), "gestural" (performance), "analytical" (music theory analysis), and "logical" (houses all common information of the 3 other) domains (Newcomb, 1991). It accomplishes this by referencing different files, or sections of files (Selfridge-Field 1997, 487–88). For file formats which are well established, SMDL can point to music encoded in DARMS, MUSTRAN, SCORE, MIDI, and others.

---

[43] https://web.archive.org/web/20010417195922/http://www.codamusic.com/coda/Fin2000_pdk_download.asp
[44] The oldest version of Finale I could find was 2009, which could not produce ETF files anymore. However, this version could still import ETF files.
[45] https://web.archive.org/web/20200624160056/https://wiki.ccarh.org/wiki/MuseData_file_specification
[46] https://web.archive.org/web/20200506161351/http://www.ccarh.org/publications/books/beyondmidi/online/musedata/

Kern (Humdrum file format)

This file format was intended to be used within a software toolkit (Humdrum[47]) for various musicological applications. Humdrum itself can help create new representations of music, even of non-western traditions. This software is developed at the Center for Computer Assisted Research in the Humanities.[48]

Individual parts and instruments of a score are stored in a score horizontally (into spines), while the chronology of a score represented in kern grows vertically. The content of the format are readable American Standard Code for Information Interchange (ASCII) characters, using the International Standards Organization (ISO) representation for pitches.

[47] https://web.archive.org/web/20200625164030/https://www.humdrum.org/
[48] https://web.archive.org/web/20200513092425/http://www.ccarh.org/

Appendix B

Commercial Optical Music Recognition Software

Multiple commercial OMR systems have been introduced and disappeared since 1992, therefore, we included links from the Internet Archive to hopefully avoid the problem of disappearing websites in the future.[49] Luckily, there are also some websites of commercial OMR in the Archive which have not been available on the World Wide Web for over a decade. Here is a list of commercial OMR software mentioned in OMR literature or found on the Internet with URL links:

- Capella-Scan[50] and Capella Score Reader[51] by Capella Software
- Forte Scan[52] by Forte Notation
- iSeeNotes[53] by Gear Up AB
- Notation Scanner - Sheet Music[54] by Song Zhang
- MIDI Connections Scan[55] by cas
- Music Publisher (with MP Scan)[56] by Braeburn Software (now a product of Lauriso Software)
- Notate Me[57] and PhotoScore[58] by Neuratron (now a product of Avid)
- NoteScan[59] by Nightingale

[49] Internet Archive links are provided when possible.
[50] https://web.archive.org/web/20200520032618/https://www.capella-software.com/us/index.cfm/products/capella-scan/info-capella-scan/
[51] https://apps.apple.com/us/app/capella-score-reader/id1449570362
[52] https://web.archive.org/web/20200520042544/https://www.fortenotation.com/en/2016/11/comes-forte-scan/
[53] https://web.archive.org/web/20200520034954/http://www.iseenotes.com/
[54] https://apps.apple.com/us/app/notation-scanner-sheet-music/id1260311003
[55] https://web.archive.org/web/20200520032639/http://www.midi-connections.com/download_demos_e.htm
[56] https://web.archive.org/web/20020306002432/http://www.braeburn.co.uk/mpsinfo.htm
[57] https://web.archive.org/web/20200520041348/https://neuratron.com/notateme.html
[58] https://web.archive.org/web/20200520033046/https://www.neuratron.com/photoscore.htm
[59] https://web.archive.org/web/20200520032652/http://www.ngale.com/index_02.html

- Optical Music easy Reader[60] and PDFtoMusic[61] are by Myrad
- PlayScore[62] by Dolphin Computing
- ScoreMaker[63] (Platinum, Standard, Elements) by Kawai
- ScorScan[64] by npcimaging
- SharpEye[65] by visiv
- ScanScore[66] and Sheet Music Scanner[67] by scan-score
- Score Reader[68] by Yamaha
- MIDISCAN[69], PianoScan[70], SmartScore[71] (64, X2 Pro, X2 Songbook, X2 Piano, X2 Guitar, X2 MIDI, NoteReader[72]) by Musitek

[60] https://web.archive.org/web/20200520032851/http://www.myriad-online.com/en/products/omer.htm
[61] https://web.archive.org/web/20200520033902/https://www.myriad-online.com/en/products/pdftomusic.htm
[62] https://web.archive.org/web/20200520033739/https://www.playscore.co/playscore-2-for-ios/
[63] https://web.archive.org/web/20200526151716/https://cm.kawai.jp/products/sm/
[64] https://web.archive.org/web/20191019105038/http://www.npcimaging.com/scscinfo/scscinfo.html
[65] https://web.archive.org/web/20200520034311/http://www.visiv.co.uk/dload.htm
[66] https://web.archive.org/web/20200520033223/https://scan-score.com/en/
[67] https://web.archive.org/web/20200520040257/https://sheetmusicscanner.com/
[68] https://web.archive.org/web/19980630160554/http://www.yamaha.co.jp/xg/products/scor.html
[69] https://web.archive.org/web/19980430072720/http://musitek.com/midiscan.html
[70] https://web.archive.org/web/19990209104554/http://musitek.com/piano.html
[71] https://web.archive.org/web/20200520033359/https://www.musitek.com/smartscore-pro.html
[72] https://web.archive.org/web/20200520041615/https://play.google.com/store/apps/details?id=com.musitek.notereader&hl=en_CA

Appendix C

Ethics Certificate

McGill University
Research Ethics Board Office (REB-1, 2, 3, 4)
RENEWAL REQUEST/STUDY CLOSURE FORM

This form must be completed to request **ethics renewal approval** or to **close a study**. A current ethics approval is required for ongoing research. To avoid expired approvals and, in the case of funded projects, the suspension of funds, this form should be returned 1-2 weeks before the current approval expires. **No research activities including recruitment and data collection may take place after ethics approval has expired.**

REB File #: 156-0107
Principal Investigator: Stephen McAdams
Project Title: **Behavioural and psychophysiological correlates of response to music**
Email: stephen.mcadams@mcgill.ca
Faculty Supervisor (if PI is a student):

1. Any modifications to the study or forms must be approved by the REB **prior** to implementation. **Are there any modifications to be made that have not already been approved by the REB?** ____YES __X__NO
If yes, complete an amendment form indicating these changes and attach to this form.

2. The REB must be notified of any findings that may have ethical implications or may affect the decision of the REB. The REB must be promptly notified of any new information that may affect the welfare or consent of participants.
Are there any ethical concerns that arose during the course of this research? ___ YES _X__ NO
If yes, please describe.

3. Unanticipated issues that may increase the risk level to participants or that may have other ethical implications must be promptly reported to the REB. **Have any participants experienced any unanticipated issues or adverse events in connection with this research project that have not already been reported to the REB?** ___ YES _X__ NO
If yes, please describe.

4. Is this a funded study? ___ NO
__X__YES. If yes, **indicate the agency name and project title** and the Principal Investigator of the award if not yourself. This information is necessary to ensure compliance with agency requirements and avoid interruption of funding.

FRQSC, "Orchestration and perception: Research, theory and technological applications"
SSHRC, "Analysis, creation and teaching of orchestration (ACTOR)"

Principal Investigator Signature: _____ Date: __2019-07-28_____

Faculty Supervisor Signature: _____ Date:_____
(if PI is a student)

____ Check here if the **study is to be closed** and continuing ethics approval is no longer required. A study can be closed when all data collection has been completed and there will be no further contact with participants. Studies involving secondary use of data no longer need ethics approval when all secondary data has been received.

__X__ Check here if this is a **request for renewal** of ethics approval.

For Administrative Use

Signature of REB Chair or designate: _____ Date: August 15, 2019

Approval Renewal Period: August 15, 2019 to August 14, 2020

The researcher is responsible for ensuring that all other applicable approvals/renewals from other organizations are obtained before continuing the research.

Submit by email to lynda.mcneil@mcgill.ca. tel: 514-398-6831/6193; www.mcgill.ca/research/research/compliance/human
(UpdatedApril-17-2019)

References

Beethoven, Ludwig Van. 1976. *First, Second, and Third Symphonies: in Full Orchestral Score*. New York: Dover.

Bellini, Pierfrancesco, Ivan Bruno, and Paolo Nesi. 2004. "An Off-Line Optical Music Sheet Recognition." In *Visual Perception of Music Notation: Online and Offline Recognition*, edited by S. George. Hershey, PA: IRM Press.

Bellini, Pierfrancesco, Ivan Bruno, and Paolo Nesi. 2007. "Assessing Optical Music Recognition Tools." *Computer Music Journal* 31 (1): 68–93.

Bugge, Esben Paul, Kim Lundsteen Juncher, Brian Søborg Mathiasen, and Jakob Grue Simonsen. 2011. "Using Sequence Alignment and Voting to Improve Optical Music Recognition from Multiple Recognizers." In *Proceedings of the 12th International Society for Music Information Retrieval Conference*.

Byrd, Donald, and Megan Schindele. 2006. "Prospects for Improving OMR with Multiple Recognizers." In *Proceedings of the 7th International Conference on Music Information Retrieval,* 41–6.

Byrd, Donald, and Jakob Grue Simonsen. 2015. "Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images." *Journal of New Music Research* 44 (3): 169–95. https://doi.org/10.1080/09298215.2015.1045424.

Calvo-Zaragoza, Jorge, Jan Hajič Jr., and Alexander Pacha. 2020. "Understanding Optical Music Recognition." *ACM Computing Surveys* 53 (4). https://doi.org/10.1145/3397499.

Church, Maura, and Michael Scott Cuthbert. 2014. "Improving Rhythmic Transcriptions via Probability Models Applied Post-OMR." In *Proceedings of the 15th International Society for Music Information Retrieval,* 643–48.

Cuthbert, Michael Scott, and Christopher Ariza. 2010. "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." In *Proceedings of the 11th International Society for Music Information Retrieval Conference,* 637–42.

Diet, Jürgen. 2018. "Innovative MIR Applications at the Bayerische Staatsbibliothek." In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*

Durbin, Richard, Sean Eddy, Anders Krogh, Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabalistic Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press.

Erickson, Raymond F. 1975. "'The Darms Project': A Status Report." *Computers and the Humanities* 9 (6): 291–8. https://doi.org/10.1007/BF02396292.

Foscarin, Francesco, David Fiala, Florent Jacquemard, Philippe Rigaux, and Virginie Thion. 2018. "Gioqoso, an Online Quality Assessment Tool for Music Notation." *Proceedings of the 4th International Conference on Technologies for Music Notation and Representation*.

Foscarin, Francesco, Florent Jacquemard, and Raphaël Fournier-S'niehotta. 2019. "A Diff Procedure for Music Score Files." In *Proceedings of the 6th International Conference on Digital Libraries for Musicology*, 58–64. The Hague, Netherlands: ACM Press. https://doi.org/10.1145/3358664.3358671.

Good, Michael D. 2013. "MusicXML: The First Decade." In *Structuring Music through Markup Language: Designs and Architectures*, edited by Jacques Steyn, 187–92. IGI Global. https://doi.org/10.4018/978-1-4666-2497-9.ch009.

Grande, Cindy, and Alan Belkin. 1996. "The Development of the Notation Interchange File Format." *Computer Music Journal* 20 (4): 33. https://doi.org/10.2307/3680416.

Guo, Edward W. 2014. "A Librarian's Guide to IMSLP." *International Association of Music Libraries, Archives, and Documentation Centres (IAML)*, Fontes Artis Musicae, 61 (3): 267–74.

Gusfield, Dan. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York: Cambridge University Press.

Hajič Jr., Jan, Jiri Novotny, Pavel Pecina, and Jaroslav Pokorny. 2016. "Further Steps Towards a Standard Testbed for Optical Music Recognition." In *Proceedings of the 17th International Society for Music Information Retrieval Conference*.

Hankinson, Andrew Noah. 2014. "Optical Music Recognition Infrastructure for Large-Scale Music Document Analysis." Ph.D. Dissertation, McGill University.

Jones, Graham, Bee Ong, Ivan Bruno, and Kia Ng. 2008. "Optical Music Imaging: Music Document Digitisation, Recognition, Evaluation, and Restoration." In *Music Document Digitisation, Recognition, Evaluation, and Restoration,* 50–79. IGI Global.

Kanungo, Tapas, Robert M. Haralick, and Ihsin Phillips. 1993. "Global and local document degradation models." In *Proceedings of the 2nd International Conference on Document Analysis and Recognition*, 730–34.

Knopke, Ian, and Donald Byrd. 2007. "Towards MusicDiff: A Foundation for Improved Optical Music Recognition Using Multiple Recognizers." In *Proceedings of the 8th International Conference on Music Information Retrieva*l, 121–24. Vienna, Austria.

Mozart, Wolfgang Amadeus. 1985. *The Magic Flute (Die Zauberflöte): in Full Score*. New York: Dover. Reprint of edition published in 1791 by C. F. Peters (Leipzig).

Nápoles, Néstor, Gabriel Vigliensoni, and Ichiro Fujinaga. 2018. "Encoding Matters." In *Proceedings of the 5th International Conference on Digital Libraries for Musicology*. Paris, France. http://cloud.simssa.ca/index.php/s/lofhdmzzF9VKQ4m.

Needleman, Saul B. and Christian D. Wunsch. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Journal of Molecular Biology* 48 (3): 443–453.

Newcomb, S.R. 1991. "Standards-Standard Music Description Language Complies with Hypermedia Standard." *Computer* 24 (7): 76–79. https://doi.org/10.1109/2.84842.

Ng, Kia, Alex McLean, and Alan Marsden. 2014. "Big Data Optical Music Recognition with Multi Images and Multi Recognisers." In *Proceedings of the Electronic Visualisation and the Arts*, 215–18. BCS. https://eprints.lancs.ac.uk/id/eprint/75614/1/eva2014.pdf.

Ng, Kia, and Andrew Jones. 2003. "A Quick-Test for Optical Music Recognition Systems." In *Proceedings of the 2nd MUSICNETWORK Workshop.*

Padilla, Victor, Alan Marsden, Alex McLean, and Kia Ng. 2014. "Improving OMR for Digital Music Libraries with Multiple Recognisers and Multiple Sources." In *Proceedings of the 1st International Workshop on Digital Libraries for Musicology*, 1–8. London, UK: ACM Press.

Prerau, David Stewart. 1970. "Computer Pattern Recognition of Standard Engraved Music Notation." Ph.D. Dissertation, Massachusetts Institute of Technology.

Pruslin, D.H. 1966. "Automatic Recognition of Sheet Music." Ph.D. Dissertation, Massachusetts Institute of Technology.

Ravel, Maurice. 1970. *Alborada del Gracioso: pour Orchestre*. Paris: Editions Max Eschig.

Rebelo, Ana, G. Capela, and Jaime Cardoso. 2010. "Optical Recognition of Music Symbols: A Comparative Study." In *International Journal on Document Analysis and Recognition* 13 (1): 19–31.

Rebelo, Ana, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. 2012. "Optical Music Recognition: State-of-the-Art and Open Issues." *International Journal of Multimedia Information Retrieval* 1 (3): 173–90.

Reed, Todd K., and James Parker. 1996. "Automatic Computer Recognition of Printed Music." In *Proceedings of the 13th International Conference on Pattern Recognition*, 3: 803–7.

Ringwalt, Dan, and Roger B Dannenberg. 2015. "Image Quality Estimation for Multi-Score OMR." In *Proceedings of the 16th International Conference on Music Information Retrieval,* 17–23.

Rossant, F., and I. Bloch. 2005. "Optical Music Recognition Based on a Fuzzy Modeling of Symbol Classes and Music Writing Rules." In *IEEE International Conference on Image Processing 2005*, II–538. Genova, Italy: IEEE. https://doi.org/10.1109/ICIP.2005.1530111.

Sapp, Craig Stuart. 2005. "Online Database of Scores in the HumDrum File Format." In *Proceedings of the 6th International Conference on Music Information Retrieval*, 664–65.

Selfridge-Field, Eleanor. 1994a. "Optical Recognition of Music Notation: A Survey of Current Work." Computing in Musicology 9, 109–145

Selfridge-Field, Eleanor. 1994b. "How Practical is Optical Music Recognition as an Input Method." *Computing in Musicology* 9, 159–166.

Selfridge-Field, Eleanor. 1997. *Beyond Midi: The Handbook of Musical Codes*. Cambridge, MA: MIT Press.

Selfridge-Field, E., W.B. Hewlett, and C.S. Sapp. 2001. "Data Models for Virtual Distribution of Musical Scores." In *Proceedings First International Conference on WEB Delivering of*

Music. WEDELMUSIC 2001, 62–70. Florence, Italy: IEEE Computer Society. https://doi.org/10.1109/WDM.2001.990159.

Sibelius, Jean. 1991. *Pohjola's Daughter*. Mineola: Dover.

Szwoch, Mariusz. 2007. "Guido: A Musical Score Recognition System." In *9th International Conference on Document Analysis and Recognition,* 809–13. Parana, Brazil: IEEE.

Szwoch, Mariusz. 2008. "Using MusicXML to Evaluate Accuracy of OMR Systems." In *Diagrammatic Representation and Inference*, 5223: 419–22. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer.

Toyama, Fubito, Kenji Shoji, and Juichi Miyamichi. 2006. "Symbol Recognition of Printed Piano Scores with Touching Symbols." In *18th International Conference on Pattern Recognition*, 480–83. Hong Kong, China: IEEE. https://doi.org/10.1109/ICPR.2006.1099.

van Der Wel, Eelco, and Karen Ullrich. 2017. "Optical Music Recognition with Convolutional Sequence-to-Sequence Models." In *Proceedings of the 18th International Society for Music Information Retrieval*. Suzhou, China. http://arxiv.org/abs/1707.04877.

Viro, Vladimir. 2011. "Peachnote: Music Score Search and Analysis Platform." In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 359–62.

Wagner, Richard. 1976. *Die Meistersinger von Nürnberg: Complete Vocal and Orchestral Score*. New York: Dover.

Wen, Cuihong, Ana Rebelo, Jing Zhang, and Jaime Cardoso. 2014. "Classification of Optical Music Symbols Based on Combined Neural Network." In *Proceedings of the 2014 International Conference on Mechatronics and Control (ICMC)*, 419–23. Jinzhou, China: IEEE. https://doi.org/10.1109/ICMC.2014.7231590.

Zhang, Kaizhong, and Dennis Shasha. 1989. "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems." In *Proceedings of the Society for Industrial and Applied Mathematics Journal on Computing* 18 (6): 1245–62. https://doi.org/10.1137/0218082.